Chapter 8
Mobile Context-Aware Support for Public Transportation Users

Esben von Buchwald, Jakob Eg Larsen, and Roderick Murray-Smith

Abstract We present a fully functional location-aware application prototype named Relevant Service Suggestion System, which runs on an off-the-shelf Nokia 6210 Navigator. The system allows the user to point and probe to find the services needed, using the awareness of the user's location and the measured compass bearing, in addition to a distance range chosen by a physical gesture. The main application tested in this paper is a system to support public transport users in Copenhagen. Users can point at any bus-stop or train station and be given timetables, next departure times, and buy a ticket via SMS direct to their phone. The requirements for text entry are minimal, due to the use of location, bearing and pose sensing.

8.1 Introduction

In this paper we demonstrate how currently available technologies can be combined to make a novel ubiquitous information system for public transport users. We present a method of making public transportation information available by means of a context-aware mobile application on modern off-the-shelf mobile phones, rather than investing in expensive hardware at each bus stop or station. The study was carried out in the greater Copenhagen area in Denmark, where there is a well developed public transportation system including bus, train and metro. Information systems for acquiring public transportation information including bus, train and metro schedules and buying online tickets are available. These systems have also been adapted for mobile devices, but the daily use has been limited, due to the need to provide information about where you are going from and to via laborious text-entry.

E. von Buchwald (✉)
DTU Informatics, Technical University of Denmark, Copenhagen, Denmark

J. Eg Larsen
DTU Informatics, Cognitive Systems Section, Technical University of Denmark, Copenhagen, Denmark

R. Murray-Smith
School of Computing Science, University of Glasgow, Glasgow, Scotland, UK

In this paper we discuss the potential of supporting public transportation use by providing easier access to information (including schedules) and better integration with existing systems for buying tickets through the use of context-aware mobile applications. We present a context-aware mobile application prototype system. The client-server architecture of the system consists of a server aggregating public transportation information from multiple sources and a mobile application utilizing multiple embedded sensors on a mobile phone, presently the Nokia 6210 Navigator. The mobile user interface is a point-at-and-gesture system based on detection of current location (GPS), direction (compass bearing), and allowing gesture-based input (accelerometers). The pilot system has been developed and tested in Copenhagen, utilizing the existing systems for accessing bus, train and metro schedules, location of stops and terminals, as well as the SMS-based system for buying travel tickets. It demonstrates the possibilities of promoting public transportation with relatively small resources and investment in development, implementation and deployment compared to traditional public transportation information systems, such as information displays at a bus stop.

8.2 Related Work

Early experiments in location-aware services tended to use PDAs and external sensors to test the basic concepts of providing appropriate information as a function of location. Tourism services were a common focus, with an early example [2]. Recent work in mobile spatial interaction is outlined in [4]. Non-visual feedback in location-aware systems has included audio feedback, e.g. [5, 6, 10]. Vibrotactile navigation systems include [4]. Many applications involved predominantly visual feedback and touch or keyboard navigation, but some alternatives included interfaces where the pitch control could allow you to project further along the current bearing, as used in [10–12]. This was used in [8, 9] for various explorations of augmented information in the environment. Gestures for playful interaction with game characters is used in the REXplorer system [1].

The latest mobile phones have all the sensors necessary for such interactions, since the Nokia 6210 Navigator, Nokia N8, iPhone 4 and HTC Sensation smartphones, and we have seen a series of new applications and papers relating to this. Mobile augmented reality examples include the Layar [7] project for Android, which allows users to browse the local environment using the camera viewfinder, and it places layers of geotagged data and weblinks on the screen. Much of the recent output has been less visible in the academic literature, but has been disseminated via e.g. the Apple AppStore in the form of applications. The Wikitude API project for Android allowed developers to add their own content to their applications based on the Wikitude API [13]. Many cities' public transport systems have shown rapid improvements in their web sites for supporting users trip-planning, and these have in part been transferred to mobile applications, such as the "London Bus, Tube and Rail Journey Planner", or the one-bus-away [3] app for the iPhone, although these tend to require you to enter which stop you are at, which might not always be clearly marked, or which might require you to walk to the place to find out. The “Nearest Tube” app from Crossair is a prototype mobile augmented reality system which overlays information about how to get to the nearest train station on the iPhone screen as you focus the camera on different areas.

8.3 Usage Scenarios

We sketch two example usage scenarios, which illustrate the ways such a system could be used in practice.

1. Peter is a tourist from Germany coming to Copenhagen for a few days. He is happy to use public transportation, but doesn’t speak Danish, and is unfamiliar with Copenhagen. When he lands at the airport he also has no Danish Kroner, but only Euros. On arrival, he points his phone at the railway station in the airport, and enters his destination. The phone tells him when the next train is, how long the trip will take, how to connect to a bus that will take him to his hotel, and how much the ticket would cost. He decides that this is a good trade-off compared to the cost of a taxi, and he buys a ticket on his phone using the Copenhagen traffic system’s SMS ticket option. While on the train, he is alerted by his phone a few minutes before arriving at the transfer station, and on coming out into the large main station he easily finds his bus stop using the built in compass. The phone tells him the bus is due in 2 minutes, so he jogs to the stop to ensure he catches the bus. He shows the phone’s SMS ticket to the bus driver and boards the bus.
2. Søren lives in Copenhagen and he has to go to a job interview in a part of town he rarely visits. He thinks he could walk there, but after a few minutes he realizes he is running late, and notices a bus stop. Unsure about the bus routes, he points the phone at the bus stop and it tells him which routes go from there. He selects the route which will take him to his destination, and the phone tells him a bus will be arriving in 4 minutes, and will get him there 13 minutes earlier than expected walking speed. This makes sense, so he buys the ticket and waits for the bus.

   The system described in this paper is a first, functioning step to creating such a mobile transport aid, and has the station finding, timetabling and ticket purchasing working on an off-the-shelf mobile phone, using the existing Copenhagen traffic system.

In Fig. 8.1, the flow of a typical usage situation is illustrated, with screenshots from the actual prototype, and shows a user who wants to arrange a trip by train from a station “Peter Bangs Vej”.

8.4 System Overview

Our system consists of a mobile client and a server back-end. The mobile application primarily acts as an assistant to find the services relevant for the user in its current context (i.e. nearby bus/trains). It also lets the user interact with the selected services

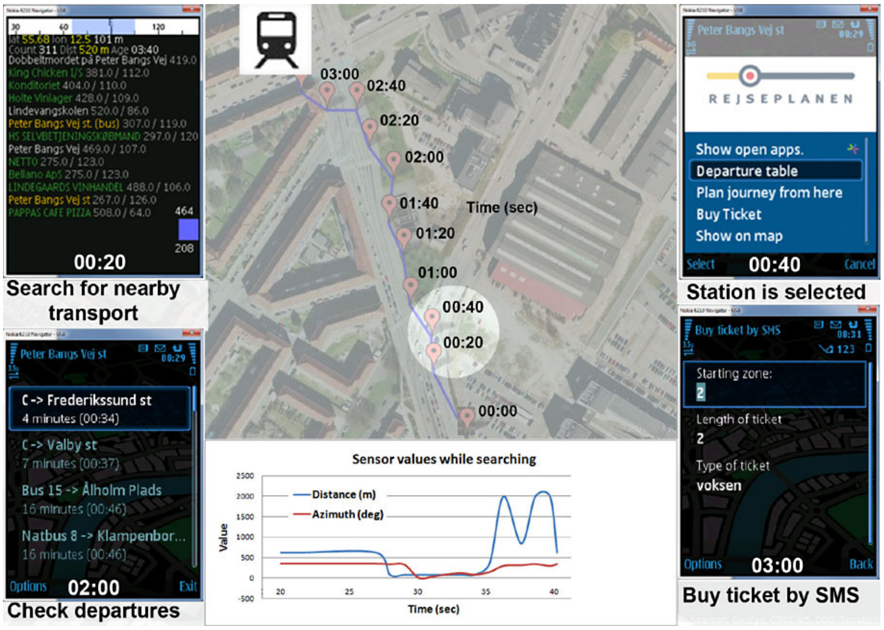

Fig. 8.1 The flow of actions when a user walks towards a train station. In the time sequence 00:20–00:40 seconds the mobile application prototype is used to find the station, through the gesture-based interaction. When the station has been chosen the user checks the departure table, finds the next departure, and buy a ticket through the integrated SMS service

afterwards, enabling the user to see live departure tables from the chosen station, plan a journey from the station to a chosen destination, and buy an SMS ticket for the trip, all through the built in features of the mobile application. To find the relevant services, it couples the known context data acquired from the sensors, with the service data received from the server application, and shows a list of matching services. The presentation of data and the interaction options for a given service is based on the service-specific data provided by the server application, which means that the mobile application is totally independent on specific service providers etc. The mobile application is running on a Nokia 6210 Navigator, and is written in Python for S60.

The server application provides data to the mobile application upon request. It basically acts as an interface between the mobile application and the provider modules, with a database to be used by the provider modules, and data import scripts to import service data from external service providers. Each module has its own customized table of data in the database, and its own API to provide a list of services, interaction options for its services, or call service-specific functions. For the 'Rejseplanen' module used in this paper, there are custom functions for getting live departure tables, train and bus station information (IDs, names, price zones, and geographic positions) and purchasing SMS tickets. The server application is written in PHP and the data about services is stored in a MySQL database.

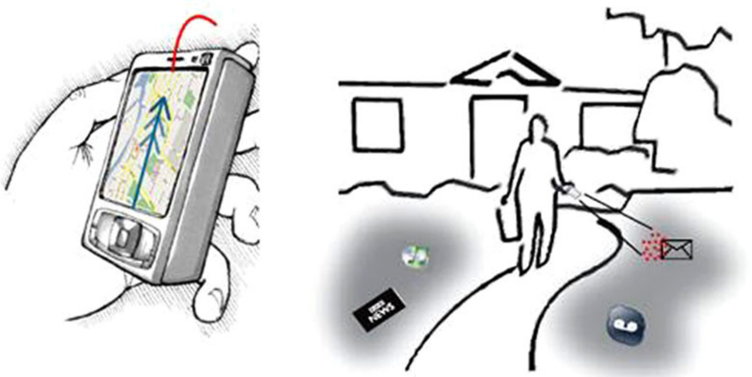

Fig. 8.2 (**a**) A user tilts the phone to indicate distance, (**b**) A user is pointing in the direction of different services

8.5 User Interaction

The system enables the user to select a point of interest (i.e. a bus terminal or train station) from the pool of available services, by simply pointing the phone in the direction of it, and selecting it when suggested by the application, as illustrated in Fig. 8.2. The user can also choose the interval of distance to the desired service, by holding the phone closer to the face or further away from the body. This movement results in a tilting of the phone around the x-axis, which is measured by the accelerometer. When tilted towards the body, it will be vertical, giving a large acceleration value for the y-axis, and when the user tilts the device away from the body, the phone will be placed horizontally, giving only small amounts of acceleration on the y-axis.

8.5.1 Location

We use a window of 1×1 degree to make 2D-plane approximation of the spherical earth surface, which results in a small but insignificant difference of about 0.1 % when calculating the distance to objects within 500 meters, in Denmark. By using this approximation, the calculations for distance and bearing between the current position and relevant services can be simplified. Using this method, we get the length of one degree, called d_{lat} and d_{lon}. The selected area in which to get services, is determined by first calculating some exact point, that the user is pointing at. With the knowledge of position (lon_0, lat_0), bearing ϕ and distance d, a new point (lon_1, lat_1), can be calculated:

$$\Delta x = \cos(\phi) \cdot d$$
$$\Delta y = \sin(\phi) \cdot d$$

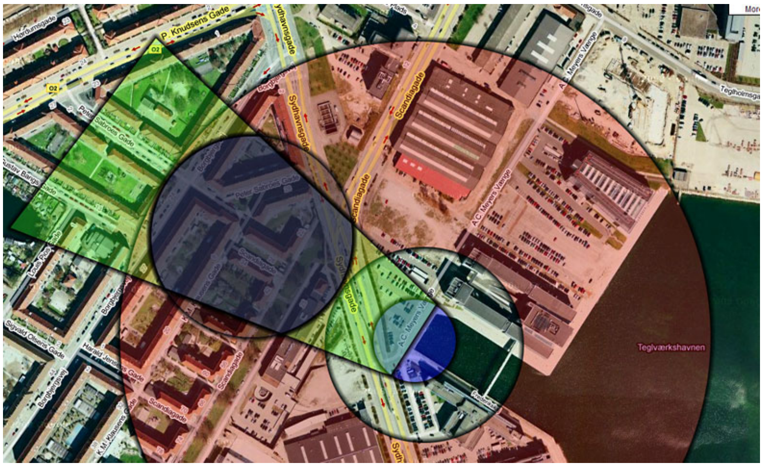

Fig. 8.3 Distance inaccuracy is *larger* than the bearing inaccuracy

$$lon_1 = \frac{lon_0 + \Delta_x}{d_{lon}}$$
$$lat_1 = \frac{lat_0 + \Delta_y}{d_{lat}}$$

Then, a circle is drawn around the calculated position, with a radius based on the reported inaccuracy for either the compass sensor or the chosen distance. If the width of the angle made by compass inaccuracy, at the chosen distance, is smaller than the distance inaccuracy, the circle is limited to only cover the angle scope made by the compass inaccuracy (see Fig. 8.3).

Otherwise, the circle is limited to only cover the area within the chosen distance interval (see Fig. 8.4).

It can be determined whether a given service is within the covered area, by simply calculating its distance to the chosen point, and afterwards filtering out the services outside distance interval or angle scope.

8.5.2 Sensor Data

The system is based on the data acquired from the GPS receiver, accelerometer and digital compass (magnetometer sensor), available in the Nokia 6210 Navigator. The magnetic north azimuth is derived from the data acquired by the 3-axis magnetometer and the accelerometer. When the phone is held in a hand, this results in a significant amount of noise. The majority of the recorded values are within a 2 degree range from the mean, which means that a filtering threshold of at least ±2 degrees needs to be applied to avoid constant flickering of the obtained value. This also affects the inaccuracy, which will increase from the approximately 20 degrees to

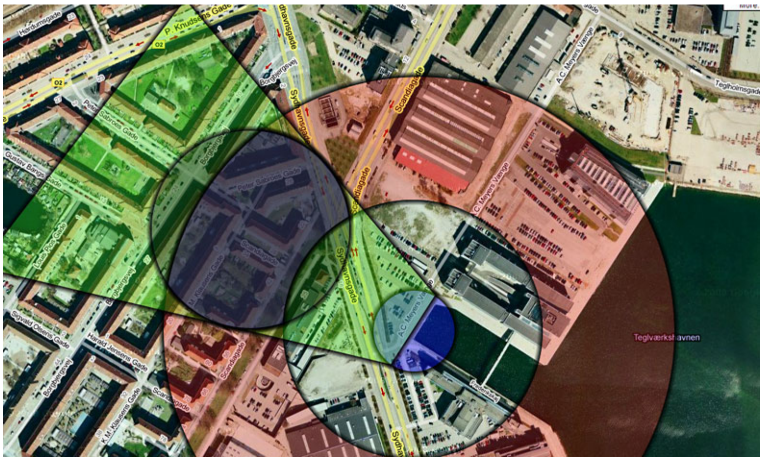

Fig. 8.4 Distance inaccuracy is *smaller* than the bearing inaccuracy

about 24 degrees. In addition, the inaccuracy caused by the magnetic declination is about 0–2 degrees in Denmark, but neglected since it is insignificant compared to the compass inaccuracy itself.

The accelerometer reports an 8-bit signed value representing $\pm 2g$ acceleration for each axis. Also a significant amount of noise is produced when the phone is handheld. Applying a low pass filter makes it stable enough for handheld usage. The Nokia 6210 Navigator offers an integrated GPS receiver with support for network assistance (A-GPS). Sensor data is logged in the mobile application to analyze user behavior and enable usability tests. When the user closes the application, the log files are uploaded to the server, and the user movements can be plotted on a map, or used for simulation.

To demonstrate the system logging feature, Fig. 8.1 also includes a graphical representation of logged data, when the user is walking, searching for services and ends up by pointing at a nearby train station.

8.5.3 Performance and Resource Consumption

The server application can easily handle multiple users simultaneously and returns hundreds of available services. The mobile application works stable and responsive, and delivers the expected results, with exception from failures caused by the phone itself (i.e. unstable network connection, unavailable sensors etc.). When searching (pointing) among 311 services loaded from the server, it provides screen updates at a frequency of 10–12 times per second. Currently the main bottleneck is getting a position fix from the GPS, which may take from seconds to several minutes.

Also, downloading and decoding a large list of services from the server for the neighboring 2 km, takes 15–20 seconds. Such delays reduce the perceived potential for spontaneous use, and optimisation of such issues is an important area. Pre-caching data, and use of newer, more advanced phones with faster network access and processors, is expected to decrease this impact. The power usage when the application is searching among the loaded services (i.e. a user is pointing around) is about 240 mA, and the CPU load is about 85–90 %. When the application is "paused" i.e. the motion sensors and display are off, but GPS is running, it consumes around 90 mA, which allows for a couple of hours usage without fully draining the 950 mA battery of the Nokia 6210 Navigator. The Python interpreter itself takes up about 3 MB of memory, where this application takes up another 2 MB. A list of about 300 services consumes another MB of phone RAM.

8.6 User Feedback

The current user interface is suitable for demonstration of the concept. Thus the system has been informally tested using two persons with no significant technical skills, but who were both experienced mobile phone users. They have been doing tasks, and provided feedback, ending up in an overall user experience discussion. Furthermore, the system has been tested at an internal Nokia workshop where eight participants could define their own tasks and experiment with the system.

In our pilot experiments the test subjects needed some instructions before being able to use the interface, but after the first few minutes, they found the interaction method to work very well, and were able to find the desired services. The current text-based list of services is not optimal for a large numbers of services, and another issue is that the services found are ordered by their distance to the center point of the selected area. This means that the order of the found services currently changes rapidly when the parameters change, i.e. the user slightly moves the hand around. During the workshop, the participants also expressed that it seems more comfortable and less awkward to use a system like this, than the recently publicized camera-assisted mobile augmented reality systems. The main issue with these is that users often feel uncomfortable walking around and "shooting" with their phone, and people on the street might think that photos are taken.

8.7 Discussion

To improve the user interface and user experience, a map-based search method, as used in [3], could augment our current system. It should show a small section of a map, similar to the area covered by the calculation method, and the services could be plotted on this map. The small changes in sensor values would then just result in the map sliding a few pixels left/right, instead of changing the contents of the present text list (see the screenshots in Fig. 8.1). The system could be expanded to a more context-specific approach of processing and presenting the available service data. It could provide everyday commuters with quick access to data about their normal route, and any unusual circumstances that day, or to notify the user to get off at the right bus stop, as implemented in Koozyt [7]. It could help tourists by guiding the user from the current location to the desired destination, by leading the user to the nearest bus/train and showing the suggested journey plan, in addition to offer an electronic ticket for the trip, to the user. The system already knows the location of the user, and its home address, and with a "*Take me home, now!*" system, could provide the necessary instructions to get home as soon as possible.

8.8 Conclusions

We have presented a location-aware application allowing the user to point and probe to find surrounding services, by means of the user's location, the measured compass bearing, and a distance range chosen by a physical gesture. The present version is a fully functioning system supporting public transport users in Copenhagen by allowing users to point at any bus-stop or train station to retrieve timetables, live next departure times, and to buy a ticket via SMS directly from the mobile phone.

Our experiments with the current prototype implementation have provided valuable insights and demonstrated the feasibility and potential of the concept as the application runs on standard off-the-shelf hardware. The generic architecture of our prototype system enables further research and rapid prototyping to be carried out. The results from our experiments carried out in Copenhagen will apply in any other city which makes basic public transportation information and infrastructure available. The system only needs station/stop locations, timetables or live arrival data. We believe that such technology can be developed and deployed with significantly lower resources and provide easy access to context-dependant information and services requiring minimal text entry.

Acknowledgements We would like to thank the participants who took part in our experiments at the workshop and the user experience tests.

References

1. Ballagas, R., Kuntze, A., & Walz, S. P. (2008). Gaming tourism: lessons from evaluating REXplorer, a pervasive game for tourists. In *LNCS. Pervasive '08: Proc. of the 6th int. conference on pervasive computing* Berlin: Springer.
2. Cheverst, K., Mitchell, K., Davies, N., & Smith, G. (2000). Exploiting context to support social awareness and social navigation. *ACM SIGGROUP Bulletin*, *21*(3), 43–48.
3. Ferris, B., Watkins, K., & Borning A. (2010). OneBusAway: results from providing real-time arrival information for public transit. In *Proceedings of CHI*. http://onebusaway.org/.
4. Fröhlich, P., Oulasvirta, A., Baldauf, M., & Nurminen, A. (2011). On the move, wirelessly connected to the world. *Communications of the ACM*, *54*(1)
5. Holland, S., Morse, D., & Gedenryd H. (2002). AudioGPS: Spatial audio navigation with a minimal attention interface. *Personal and Ubiquitous Computing*, *6*, 4.

6. Jones, M., Jones, S., Bradley, G., Warren, N., Bainbridge, D., & Holmes, G. (2008). Ontrack: Dynamically adapting music playback to support navigation. *Personal and Ubiquitous Computing*, *12*(7), 513–525.
7. Koozyt. http://www.koozyt.com.
8. Robinson, S., Eslambolchilar, P., & Jones, M. (2009). Exploring casual point-and-tilt interactions for mobile geo-blogging. *Personal and Ubiquitous Computing*. doi:10.1007/s00779-009-0236-5.
9. Robinson, S., Eslambolchilar, P., & Jones, M. (2009). Sweep-Shake: Finding digital resources in physical environments. In *Proceedings of MobileHCI*. Bonn, September 2009.
10. Strachan, S., Eslambolchilar, P., Murray-Smith, R., Hughes, S., & O'Modhrain, S. (2005). gpsTunes: controlling navigation via audio feedback. In *Proc. MobileHCI '05* (pp. 275–278).
11. Strachan, S., & Murray-Smith, R. (2009). Bearing-based selection in mobile spatial interaction. *Personal and Ubiquitous Computing*, *13*(4), 265–280.
12. Strachan, S., & Murray-Smith, R. (2009). Nonvisual, distal tracking of mobile remote agents in geosocial interaction. In *LoCA*, Tokyo.
13. Wikitude API. http://www.wikitude.org.