The COMPASS Location System

Frank Kargl and Alexander Bernauer

University of Ulm, Dep. of Multimedia Computing, Ulm, Germany

Abstract. The aim of *COMPASS* (short for *COM*mon *P*ositioning *A*rchitecture for *S*everal *S*ensors) is to realize a location infrastructure which can make use of a multitude of different sensors and combine their output in a meaningful way to produce a so called *Probability Distribution Function (PDF)* that describes the location of a user or device as coordinates and corresponding location probabilities. Furthermore, COMPASS includes a so called translator service, i.e. a build-in component that translates PDFs (or coordinates) to meaningful location identifiers like building names and/or room numbers. This paper gives a short overview on the goals and abilities of COMPASS.

1   Motivation

There are a lot of situations in mobile computing where mobile nodes need to determine their current position. Ubiquitous computing applications derive context information from this position, e.g. in order to determine whether a user is currently at home, at work or on the way in between. Location-aided routing protocols for ad-hoc networks need position information to support their routing decisions. Navigation systems naturally rely on precise position information to plan the further route of a car or pedestrian. To support this large demand that applications have for precise location information, a number of commercial and research projects are working on this subject. Section 2 gives an overview on some of these activities.

We have identified two major challenges that are not completely resolved yet:

1. Location information from multiple sensors needs to be combined effectively in order to present one and only one position to the application. Any single location sensor has drawbacks, e.g. is usually not available inside buildings, RFID sensors or WLAN/Bluetooth APs are only available where installed etc. So in order to provide reliable and pervasive location support, an architecture must use multiple sensors, combine their results and present this to the application. The application should not need to worry about what sensor(s) were used for the current position information. Additionally combining the results from multiple sensors may improve the precision of overall results.
2. Raw coordinates may not really be useful to an application that needs to know the position in terms of buildings, rooms, street names etc. So a location system should include an infrastructure to resolve the raw position information to some kind of *symbolic position.*

The primary focus of COMPASS will be to address these two issues by both including many different sensors into the system using a plugin interface and by providing a translator that is able to derive symbolic location information from the raw coordinates received from the locator. COMPASS is a software framework that can be used by arbitrary applications for location retrieval.

2   Related Work

The need of location systems is almost as old as mobile computing itself. Many of them use satellite navigation systems like GPS [G93] or the future Galileo system [G05]. A major problem of satellite navigation systems is the fact that the antenna of the receiver usually needs a direct line-of-sight towards a number of different satellites. So they are only useful for outdoor navigation.

As many ubiquitous computing projects include mostly indoor scenarios, researchers started to develop specialized indoor location systems. Prominent examples include the Cricket Location-Support System [P00] or the Bat [H97]. These systems make use of different kinds of sensors, like scanning for ultrasound or radio beacons or observing nearby WLAN or Bluetooth access points.

Unfortunately most of these location systems do not work together and many can use only one single kind of sensor. So there is a clear need for a framework that can combine the output of different kinds of location sensors into one single and consistent result.

Such a system has been proposed as part of the HeyWow project [H03]. In [A01, W02] the authors suggest the use of so-called probability density functions (PDFs) to represent the location measurement of one sensor or the combined measurement of multiple sensors. Other similar projects include [B03, H02].

As our architecture is based in part on these ideas, we first give some details on how position is represented in COMPASS before describing the architecture itself.

3   Position Representation

A major issue in positioning systems is how to express the position. COMPASS knows two kind of position representations: a geocoordinate based representation and one that delivers a semantical description of the current position, like the current room number or a street address. The functionality of COMPASS includes a mechanism to translate a geocoordinate to a semantic position description automatically, as sensors often deliver the first representation whereas applications often need the semantic representation.

No matter what kinds of sensors are in use to determine geocoordinates, most of them will inevitably introduces some kind of error. E.g. GPS has a typical error of a few meters, estimating the position based on available WLAN access points will deliver results with a precision of a few dozens to a few hundreds of

meters, depending on local conditions. So delivering a single point as position information will never be accurate.

Therefor COMPASS expresses all position information as *Probability Distribution Functions (PDFs)* like introduced in [A01, W02]. PDFs represent a two or three dimensional area in which they express the probability of being at a certain position. PDFs use a Cartesian coordinate system with a north-south (y), east-west (x) and up-down (z) axis. Additionally a PDF contains the origin expressed as WGS 84 coordinates. This way, multiple PDFs can be correlated and combined.
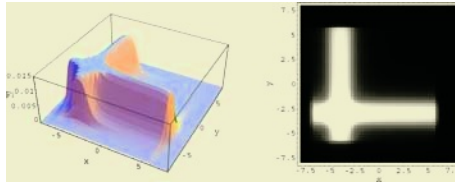

Fig. 1. Example of Probability Distribution Function (from [A01])

Figure 1 shows an example PDF which might represent a user that is inside a building with crossing corridors. This information can e.g. be the result of a radio sensors which detected that the mobile user entered this corridor and has not left it since. Combined with PDFs from other sensors, the position within the corridor might be narrowed down further.

In parallel to coordinates, positions may also be provided in a symbolic representation. At the moment we use a hierarchical string of the form "'country.city.streetname.streetnumber.roomnumber"' for simplicity. But we are now switching to a more powerful RDF-based representation that offers a very flexible description of locations. See the final section for an outlook on this mechanism.

4   Architecture and Components

4.1   Principles

COMPASS is designed to run on mobile devices. Therefore memory capacity, CPU speed and power consumption have to be taken into account. Depending on the application's needs the desired accuracy of position determination can reach from some centimeters to several hundred meters. COMPASS is designed to work with different degrees of accuracy to be usable for a wide spectrum of applications.

To gain a maximum of flexibility any dynamic content is separated from the COMPASS system and displaced to remote databases. A database is accessed using Web Services technology and is generally called service within the context of COMPASS. The application can optionally influence the selection of the

services. It is possible for both the application and the COMPASS system to replace services at runtime without deep impact.

4.2   Overview

Figure 2 shows the overall architecture of COMPASS.
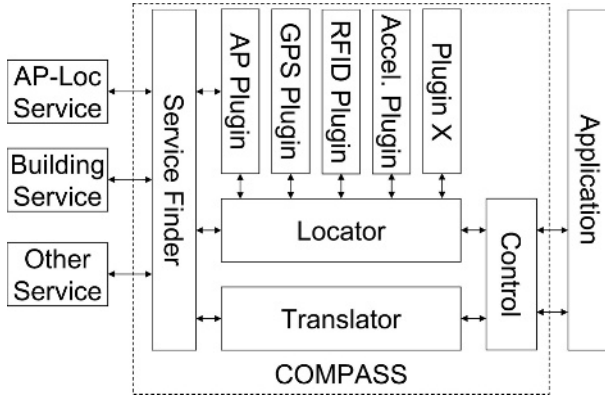

Fig. 2. COMPASS architecture

COMPASS has a plugin based design. For any source of position information exists a corresponding plugin. The plugins are connected to the so called Locator and deliver a PDF to it on demand. A plugin may use a service for accessing additional information. The task of the Locator is to determine the compound PDF of all PDFs supplied by the plugins. Additionally the Locator computes the position of the highest probability.

A plugin may register itself at the Locator to always get the latest compound PDF. This is useful for sensors which provide only relative position informations. To provide a human readable representation of the position with the highest probability there is a Translator component.

It will use webservices, that are able to convert PDFs to symbolic location informations. These symbolic location informations may be represented as hierarchical strings like "'germany.ulm.university.main_building.o27.3303"'. It is possible to specify the hierarchical depth of the response string. In the future we plan to use a more flexible, XML-based format instead of simple strings.

The Service Finder is responsible for finding proper RPC services and to assign them to the plugins and the Translator. Using standard Web Service technologies like WSDL, UDDI and SOAP, it will find local services that provide information to plugins or the Translator.

Locator and Translator are called by the Control unit which provides the API. It is also responsible for initialization of all components. The API provides either the compound PDF or the WGS84 coordinates of the most likely position. Additionally the application can retrieve the symbolic position information.

4.3   Plugins

Currently four plugin types are supposed to be used:

AP plugin: This module uses WLAN access points as source of position information. It needs a service to resolve the access point's MAC to a geographic position. The service provides the WGS84 coordinates of the AP and a power density spectrum. From this information the plugin can compute a PDF with respect to properties of the sensor's antenna. If multiple access points are within reach one PDF for each access point can be computed.

GPS plugin: A NMEA capable GPS sensor is used to retrieve the current position. This can immediately be transformed into a PDF with Gaussian distribution. The error depends on receiving conditions, number of available satellites, etc.

RFID plugin: RFID tags have only a short range. But if they are scattered throughout a building at doors and gateways they can provide position information with a very good accuracy. The RFID plugin is statefull and for instance logs if one enters a room. A service is needed for resolving the tag's IDs and to retrieve information about the building's structure. The RFID plugin typically provides a PDF with equal distribution for the whole room which was entered last.

Acceleration plugin: This plugin uses a gyro sensor to gain relative position information. With the help of the last know position the plugin can compute a PDF. The distribution function usually is a sphere around the last known position. If a compass is additionally used the sphere can be clipped.

5   Implementation

5.1   Probability Distribution Functions

A PDF maps from Cartesian coordinates to probabilities. Therefore every PDF has an origin, which is given in WGS84 coordinates. A PDF can be accessed by supplying coordinates and retrieving the corresponding probability. To implement a PDF on a hardware we need a finite and discrete representation. So every PDF has a resolution and a maximum expansion for each dimension. A PDF always covers a cuboid. When iterating over the cuboid the sum of the probabilities must always be one. It is expedient to agree on a maximum resolution for all PDFs. Our implementation uses a maximum resolution of 10 centimeters. This should be adjusted depending on the precision of the existing sensors and the desired accuracy.

The naive approach for an implementation of a PDF is a three dimensional array. This is easy to implement and has minimum time penalty for accessing. But this approach is not practical for all possible PDFs, as depending on the resolution and the expansion of the PDF it quickly blasts the memory capabilities

of any mobile device. To save memory the intern resolution can be reduced by using interpolation. Of course accuracy suffers from this.

A second possibility is to have a function representing a mathematical formula, which calculates the probability on demand. This approach has minimum memory requirements but is expensive at runtime. A mathematical description of some physical behavior often differs from reality or the best known formula is too complex for computation at runtime. Furthermore reality may differ from the mathematical description locally because of some irregularities such as obstacles which are not considered by the formula. So this approach is not practical for all possible PDFs, either.

To combine the advantages of both approaches we define the PDF container. A PDF container is either a PDF or a list of PDF containers. In the second case requests are delegated to the proper containers. So a container looks like a PDF but can cover a hierarchy of sub-PDFs, each being optimized for access on the set of coordinates they cover. When accessing the container it has to determine which container is responsible.

If the number of sub-PDFs is small, three dimensional polygon intersections are a good way to determine the responsible sub-PDF. If the number of sub-PDFs is large, Z curves [B99] are used to map the three dimensional coordinates to a linear search index of a binary tree.

5.2   Modules

Plugins: Every plugin creates its own thread on initialization. The Locator can trigger the determination of a PDF. When finished the PDF is returned using a callback to the Locator. The plugin is allowed to deliver a list of PDFs when there are multiple sources of information. But is is also allowed to compute a compound PDF on its own and deliver only this one. The plugin is allowed to return no PDF, if it is unable to determine one. The stub of the webservice is supplied by the Service Finder. A plugin is allowed to cache results from a service. But if the Service Finder assigns a new service the cache has to be invalidated. A plugin is allowed to be statefull. If the Service Finder assigns a new service the state has to be reset if it depends on the service. Otherwise the state is allowed to be reset.

Locator: The task of the Locator is to poll all plugins on demand and to deliver the compound PDF of all returned PDFs. The Locator creates its own thread on initialization. When triggered by the control unit the locator triggers every plugin to determine the PDFs. As soon as every plugin delivered one ore more PDF or after a timeout the compound PDF is built and returned to the control unit via callback. The compound PDF is a special PDF container. The difference to normal PDF containers is that there can be multiple responsible PDFs for a set of coordinates. For mathematical details on how to combine different PDFs, see [A01, W02].

Translator: The translator uses the point of maximum likelihood from the combined PDF and searches via the service finder for a suitable service that can determine a symbolic representation for that position.

Service Finder: The service finder is responsible to find webservices and to supply the plugins and the translator with a proper stub. The service finder is allowed to assign any service at any time to any module. It is also allowed to remove a service when the service is not reachable. In this case the module is unable to fulfill its task. The Service Finder can use several techniques to find an webservice. This includes Jini, UDDI or SLP.

6   Summary and Outlook

COMPASS provides an architecture that allows the concurrent use of multiple location sensors that can assist each other in finding positions and reduce potential errors. In addition the Translator provides symbolic position information that can be retrieved from local web services.

We are currently finishing a prototype implementation that includes two plugins (GPS and AP). As soon as this is finished, we will do some real-world analysis in order to verify, that this approach is practical and really decreases errors.

On the conceptual layer, we are investigating how to create a more flexible description of symbolic position information. Depending on the current context, an application may understand a location as "being in a certain city", "being inside a specific building", "being near some important monument", at a certain postal address, etc. In order to express this information about a given geolocation, COMPASS represents such facts as RDF/XML documents using the Resource Description Framework developed by the W3C [W05] as part of their Semantic Web initiative. The following RDF document shows an example, where the object "myself" is located at some coordinates, at an certain address and inside a specific building.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\#">
  <Object rdf:nodeID="myself">
    <locatedAt> <Coordinates rdf:nodeID="myCoords" lc:type="WGS84">
        <longitude parseType="Resource">
          <rdf:value>010°01.538E</rdf:value>
          <coordUnit>degree<coordUnit>
        <longitude>
        <latitude rdf:parseType="Resource">
          <rdf:value>48°27.182N</rdf:value>
          <coordUnit>degree<coordUnit>
        <latitude>
    <Coordinates> </locatedAt>
    <locatedAt>
      <Address>
        <street>Albert-Einstein-Allee></street> <number>11</number>
        <city rdf:resource="http://ulm.de/" />
      </Address>
    </locatedAt>
```

```
    <locatedAt>
      <Building rdf:about="http://uni-ulm.de/campus/UniOst">
        <inPart><BuildingPart>O27</BuildingPart></inPart>
      </Building>
    </locatedAt>
  </Object>
</rdf:RDF>
```

This way applications may be enabled to combine the location information with other documents from the Semantic Web like route information. Then e.g. navigation systems may automatically infer that you are on an Autostrada in Italy and that there the general speed limit is 130 km/h.

References

[A01]   M. Angermann & J. Kammann & P. Robertson & A. Steingaß & T. Strang, *Software representation for heterogeneous location data sources within a probabilistic framework*, International Symposium on Location Based Services for Cellular Users, pp. 107–118, Locellus 2001.

[B99]   Christian Boehm & Gerald Klump & Hans-Peter Kriegel, *XZ-Ordering: A space-filling curve for objects with spatial extension*, Proceedings of Advances in Spatial Databases, 6th International Symposium, SSD'99, Hong Kong, China, pp. 75–90, July 20–23, 1999.

[B03]   Jürgen Bohn & Harald Vogt, *Robust Probabilistic Positioning Based on High-Level Sensor-Fusion and Map Knowledge*, Technical Report nr. 421, ETH Zurich, Apr. 2003.

[G93]   I. Getting, *The Global Positioning System*, IEEE Spectrum 30, 12, pp. 36–47, December 1993.

[G05]   *Galileo Project Webpage*, http://europa.eu.int/comm/dgs/energy_transport/galileo/index_en.htm

[H97]   A. Harter & A. Hopper *A New Location Technique for the Active Office*, IEEE Personal Communications 4, 5, pp. 43–47, October 1997.

[H03]   *HeyWow Project* http://www.heywow.com/

[H02]   J. Hightower & B. Brumitt & G. Borriello *The location stack: A layered model for location in ubiquitous computing*, In Proceedings of the 4th IEEE Workshop on mobile Computing Systems & Applications (WMCSA 2002), IEEE Computer Society, pp. 22–28, Callicoon, NY, USA, June 2002.

[P00]   Nissanka B. Priyantha & Anit Chakraborty & Hari Balakrishnan, *The Cricket location-support system*, Mobile Computing and Networking, pp. 32-43, 2000.

[W05]   W3C, *The Resource Description Framework (RDF)*, http://www.w3.org/RDF/.

[W02]   K. Wendlandt & A. Ouhmich & M. Angermann & P. Robertson, *Implementation of Soft Location on mobile devices*, International Symposium on Indoor Localisation and Position Finding, InLoc 2002, Bonn, Germany, 2002.