Software Representation for Heterogeneous Location Data Sources Using Probability Density Functions

Michael Angermann, Jens Kammann, Patrick Robertson, Alexander Steingaß, Thomas Strang
Institute for Communications and Navigation, German Aerospace Center (DLR)
P.O. Box 11 16, D-82230 Wessling, Germany
Fax: +49 8153 28 1442 E-mail: {firstname.lastname}@dlr.de

Abstract

In this paper we present a framework for the representation of location information from various sources, such as satellite navigation systems, wireless positioning technologies, beacons, indoor navigation systems, human input, etc. These sources all operate at various degrees of accuracy and often suffer from independent errors. Their output, in terms of the location, can be represented more generally as a probability density distribution (PDF) of the location over a two or three dimensional space - typically Cartesian or other co-ordinates. Combining two or more such PDFs yields a more accurate PDF of the location and improves navigation under difficult circumstances such as indoors or in fading environments. To allow practical deployment of such a framework we define a simple software interface, using the Java programming language, that relies on the transfer of software objects and class files describing the individual PDFs. A number of different positioning sources can thus describe their individual location PDF using a Java class and object, and pass this to other (sub-)devices that have no a-priori knowledge of this PDF for combination with other PDFs.

1 Introduction

1.1 Motivation

Today's rapidly evolving positioning technology allows us to use several different positioning systems in order to determine the location of real world objects and people. Ongoing development of cell phones, PDAs and GPS receivers will lead to more capable integrated communication and navigation devices with a common interface to the human user. We call them wireless information devices (WIDs) with location awareness. One example for this could be a travel assistant which provides local relevant information like time table of a near-by bus/train stations and other location based services (LBS) [1].

Global Navigation Satellite Systems (GNSS) and network based positioning systems typically deliver geometric co-ordinates (e.g. WGS-84 or Gauss-Krueger) whereas radio beacon and manual user entry systems result more often in symbolic co-ordinates (e.g. `Airport.Terminal1.Level2.Gate13`) with different characteristics regarding precision, reliability, and coverage area. Usually this different kind of information is not compatible and needs to be handled separately, yet the possibility of linking this information together results in new applications and also more accurate positioning and better routing. For instance, routing is far more efficient and information becomes more valuable to the end-user when the representation by co-ordinates is replaced or augmented by a topographic one. If we are able to combine different forms of the description of location from different sources then we are able to achieve higher precision, greater degree of redundancy and also simplified overall usability for the end-user. The improvement of accuracy is the main focus of the work presented in this paper.

We show how the mapping between geometric and symbolic representation can be accomplished by means of geometric objects with attributes. These geometric objects can be formally viewed as probability density functions (PDFs) with usually uniform distribution across their area or volume. Going one step further, non-uniformly distributed PDFs may result from individual navigation techniques such as satellite systems or time-of-arrival schemes in mobile radio. The important concept is that all PDFs which are currently available can be combined to calculate the overall PDF of the position. In practice, object oriented programming languages like Java allow software objects to carry the mathematical representation of the PDF within themselves which lets WIDs determine the relationship between different object types (e.g. geometric / symbolic object) without further support by the net-

work. In other words, software objects with attributes and member functions represent the above geometric shapes and location PDFs. Many interesting PDFs can be constructed using simple mathematical functions such as normal distributions, the inverse tangent function, basic arithmetic operations, and powers. In fact this permits symbolic position information in distributed networks (e.g. server centric networks with beacons in hot spots), as well as the efficient combination of various positioning techniques such as satellite navigation and mobile radio systems. Furthermore, in the case of navigation in buildings, the operator of such a building (e.g. airport) is able to define the software representation which maps co-ordinates to topographic descriptions (e.g. gates, terminals, halls, lounges, shops, transport, etc) and allows easy navigation and routing calculations to take place within this self-defined framework.

The paper is organised as follows: we begin by outlining different location technologies, and follow this by a brief summary of reference co-ordinate systems. Then we define the PDF notation and illustrate how PDFs are combined to yield a more accurate position. We then present a simple Java interface that can be used as a possible basis for software standardisation. We conclude by presenting some outlooks on how this technique might be extended.

2 Location Models

2.1 Sensors and their errors

Location information can result from various sources. Generally, they can be technical, human or derived from the context. Technical sources which provide absolute location information include the Global Positioning System (GPS), network based location [2, 3] and indoor navigation and communications systems based on short-range wireless [4].

GPS provides worldwide location information at high accuracy (errors typically 20-30 meters). However it requires visibility of the sky and so its errors depends on the geometrical position of the receiver with respect to the satellites being tracked and effects of the propagation conditions. The error can be most simply represented by the GDOP (geometrical dilution of precision).

Location information can also result from a mobile communication system. This can be done either network-based or handset-based. Network based location methods include Time Difference of Arrival

(TDOA), Angle of Arrival (AOA) or Multi-path Analysis. Handset based techniques use network-assisted GPS, Advanced Forward Link Trilateration (AFLT) or Enhanced Observed Time Difference (E-OTD).

Especially for indoor navigation, short range wireless communication can be used for inherent location information: The reception of infrared, ultra sonic [5] or low power radio signals (e.g. emitted from a Bluetooth [6] equipped Local Service Point) means proximity to the stationary sender whose location is usually known with high precision. The position error mainly depends on the range of the short range communication system.

Another source of location information comes from the human user himself. Addresses, zip codes and area codes can be mapped to points or areas with the help from information stored in databases. These databases can be structured hierarchically: E.g. `"Germany.Munich.Airport.Terminal_A.Gate2"` represents from left to right locations with increasing precision at the expense of an increased complexity of the underlying database whose fault tolerance or multi-lingual support mainly influences the precision of the obtained locating information. ZIP codes are typically also formed hierarchically. However, the structure of area codes depends very much on the country. So do location mistakes due to single digit errors: E.g. in Germany, "08xxx" stands for cities in Bavaria whereas area codes in the United States are more or less randomly distributed over the entire country (e.g. "408 - San Jose, California" and "407 - Orlando, Florida"). Other man-made classification depend on the environment the user of a location information system is situated.

Finally, location information can also be derived from the context a person is in: E.g. an electronic travel assistant which knows a person is booked on a train from A to B can make an assumption on the person's location with additional information like actual time, updated time table and route information. In this case, accuracy of the determined position depends on the combined reliability of all information sources that span the given context.

It is important to note that in order to combine information from any of the sensors mentioned above, a common reference system for coordinates gets applied as described in the next paragraph.

2.2 Reference System

To provide compatibility to the GPS navigation system we use WGS 84 coordinates on a global scale and Cartesian coordinates on a local scale.

The WGS 84 coordinate System is based on an ellipsoid which has a long axis aligned to the rotation axis of the earth. The WGS 84 coordinates give the longitude λ and the latitude φ, and the altitude h above the ellipsoid. The parameters for the ellipsoid are:

Large half axis	a	6378137 m
Ellipticity	f	1/298.257223563
Angular velocity	ω	$7.292115 \cdot 10^{-5}$ rad/s
Geocentric gravitation constant	GM	$398600.5\ km^3/s^2$
2nd zonal harmonic	$C_{2,0}$	$-484.16685 \cdot 10^{-6}$

Since the earth is non-isotropic the height above sea level (from the geoid) and the height above the ellipsoid differs (see Fig. 1). This difference can reach

Figure 1: Comparison between geoide and ellipsoid. Taken from [7].

up to 100 m.

To convert WGS84 coordinates into Cartesian coordinates we use [7],

$$\vec{X} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (N+h)\cos(\varphi)\cos(\lambda) \\ (N+h)\cos(\varphi)\sin(\lambda) \\ \left((1-e^2)\,N+h\right)\sin(\varphi) \end{pmatrix} \quad (1)$$

with the factor

$$N = \frac{a}{\sqrt{1-e^2\sin^2(\varphi)}} = \frac{a}{\sqrt{1-f(2-f)sin^2(\varphi)}}. \quad (2)$$

The small half axis b can be calculated by

$$b = a(1-f). \quad (3)$$

The 1st numerical excentricy e is defined by

$$e^2 = \frac{a^2 - b^2}{a^2}. \quad (4)$$

For the reverse transformation to elliptic coordinates we get:

$$h = \frac{\sqrt{x^2 + y^2}}{\cos(\varphi)} - N \quad (5)$$

$$\varphi = \arctan\left(\frac{z}{\sqrt{x^2+y^2}}\frac{1}{1-e^2\frac{N}{N+h}}\right) \quad (6)$$

$$\lambda = \arctan\left(\frac{y}{x}\right) \quad (7)$$

An example:

Oberpfaffenhofen in Germany is at

$$\begin{pmatrix} \varphi \\ \lambda \\ h \end{pmatrix} = \begin{pmatrix} 48.0863^\circ\ N \\ 11.2786^\circ\ E \\ 576\,m \end{pmatrix} \quad (8)$$

$$\widehat{=}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 4.1865 \cdot 10^6 \\ 8.3492 \cdot 10^5 \\ 4.7237 \cdot 10^6 \end{pmatrix} m \quad (9)$$

2.3 Models and Relationships

When looking at all the different types of sensors available today, we can separate them into two major groups: The first group, delivering some kind of vector in a n-dimensional coordinate system, are using a geometric model to represent a location (GPS, etc). In the most easy case, sensors of that kind deliver a *point* in an appropriate coordinate system, and it is very easy to calculate for instance the distance between two points in the same coordinate system. Even if the coordinate systems differ, only one of them needs to be transformed into the other one, and the problem of distance measuring can be easily solved. When the sensors deliver not only a point but a multi-dimensional shape, which is spanned by a set of vectors, the calculation of overlapping regions, minimum-distances etc. are more complex, but solvable by the use of mathematics.

Dealing with sensors using symbols (e.g. `airport Terminal1.Level2.Gate3`) to represent a location is somewhat more complicated; They are referred to as using a symbolic model (GSM cell ID, room-identifiers etc.).

Symbols can not be as easily processed by computers as geometric vectors. They do have relationships between sets of symbols. E.g. Gate3 is nearby Gate4, or GSM cell 412 partly overlaps ZIP code zone 81234. But the more this is conceptionally self-explanatory to humans, the less it is usable as a base for calculations

by the machine. That causes the necessity for a mapping of symbols to a reference coordinate system. This is usually done by large databases.

But as mentioned before, humans are more interested in achieving symbolic names when using electronic systems for navigation. It is much easier to handle a routing recommendation like "go down the corridor, 3rd door on the right" then getting a vector and a map of the area with a coordinate grid. Thus, the mapping-direction from geometric vectors to symbols is also of interest. One approach is the use of a *semi-symbolic hierarchical location model* as described in [8]. This approach uses objects called *SemiSymbolLocator* containing symbolic information as well as geometric information to represent locations. Moreover, the object also contains information about the hierarchical classification (locatedAt, containsEq etc.) of its data.

By using such SemiSymbolLocator objects it is possible to examine spatial relationships like overlap, inclusion, adjacency and distance.

Any navigation system uses sensor data to detect the current position of the user and to set this position in relation to its internal knowledge about the surrounding area. This internal knowledge is mostly implemented by the use of 2-dimensional geographical maps, which have a fixed relation to a reference system (WGS84). Mapping is done by calculating the point or the area in the reference system which fits the sensor's location information best, as exact as the sensor's data allows.

Such a system can be significantly improved, if the system takes into account

- different characteristics of multiple sensor types

- The fact that symbols of any characteristics can be modelled by a superposition of basic n-dimensional shapes.

A sensor delivers a location information and an "area of validity" (AOV) of that information. This AOV is mainly determined by the possible error of the information. The AOV can be modelled for simplicity in a first step by the use of simple geometric shapes. For instance the information from a GSM antenna sector can be modelled by a simple triangle (2D) or cylinder "pie-slice" (3D). GPS coordinates are points, and ZIP-codes cover an area, which can be modelled as polygons. One can say that every sensor type has its own characteristic concerning the AOV. Combining or super-positioning data from other - or even the same - type of sensors with different characteristics increases the accuracy of the resulting location information, which is an intersection of all the available sensor data in that case. One example of superposing different characteristics of the same sensor are the circular segments as a result of superposing of GSM antenna sectors and GSM timing advance measurements.

But not only the sensor data can be modelled by using simple geometric shapes, the space used to compare to represented by symbols can also be. Imagine any desired room. The idea is to fill up the volume of the room with lots of primitive volume elements of only a few types like cubes, spheres and others. We call this a *collection of primitives*, where the primitives are virtually positioned side-by-side to fill up the volume. Obviously the precision will yield better, the more and smaller primitive elements are used.

Inside a collection, every primitive can be defined by only a few parameters. The collection is the "cage" for the primitives and determines an absolute coordinate system for the primitives. If desired, an implementation of the collection can choose to use a different coordinate system inside, but is required to provide a mapping function to the reference coordinate system.

The next step towards modelling the real world better is to allow the primitives to overlap. This idea has been published among others in [9] with the concepts of auras. Further improvements are to allow the primitives to be additive or subtractive. The possibility of subtracting a primitive from a collection reduces the necessary number of small primitives, but the precision doesn not change.

There is still one remaining disadvantage: The shape of every single primitive is sharp. When using them in navigation systems, the result is always to be "in" or "out" of a specified area. That is why we call them *hard primitives*. The disadvantage may become clear when imagining the superposition of two different sensor information with sharp boarders [see fig. **??**]: The receiver of a navigation system has to determine to be inside or outside of a specific sector. Even if the user is very close to, but outside of the sharp border, the resulting location estimation is unprecise.

A solution for this are soft primitives, which are presented in the next paragraph.

2.4 Soft primitives

As seen in the last section, dealing with sharp-bordered shapes may result in a conflict situation. So

Figure 2: Error with sharp borders.

the solution is to build *soft primitives* by using Probability Density Functions (PDF).

They have the same desired characteristics as hard primitives

- only a few parameters are necessary to define the PDF

- super-positioning is an easy mathematical operation

- they can be positioned in a absolute coordinate system

The most valuable gain is that they fit much better to reality. It is much more precise to model an antenna sector by the use of a PDF than by the use of a triangle. And because of the mathematical characteristics of a PDF (see section below), the conflict situation described in the previous section doesn't exist.

Through the use of PDFs, every location information depends on a probability ("You are with a probability of 95% in the antenna sector 47"). Even so all relationships between location informations ("The area with 96% abidance probability is ..."). So the navigation system handling soft primitives needs to provide appropriate threshold values for all calculations.

2.5 Primitive Probability Density Functions

We now illustrate the way in which we combine the outputs of several positioning sources to yield a probability density function (PDF) of the location over space. It is important at this stage to make some assumptions:

1. We shall first assume that the sources are independently disturbed, in other words that each source of the position suffers from uncorrelated errors.

2. Each individual positioning function or device i returns a PDF of the location. The location is defined within the co-ordinate system (x, y, z).

3. This PDF is deemed to be accurate in the sense that it correctly models *all* errors of the function. These errors must include technical failures, measurement inaccuracies, malevolent attack on the system, and other frequent or infrequent events. Many of these errors will result in the PDF taking on small but non-zero values far from the peak or main area. As a result, the PDF is an accurate indication of the reliability of the correct estimation, without being biased.

2.6 Derivation sketch

We now briefly sketch the derivation of the optimal combiner of different location sources. It must first be pointed out that the independence criterion does *not* imply that positioning sources are uncorrelated; in fact we expect that they will be correlated. The PDF of source i is actually a conditional density distribution, $p_l^{(i)}(l \mid o(i))$, conditioned on a specific set of discrete observations or measurements $o(i)$. For example, a GPS receiver will yield a PDF of the location l conditioned in effect on its antenna input signal and HW/SW characteristics and configuration, as well as the GPS's current status.

Given a number n of these PDFs, $p_l^{(1)}(l \mid o(1))$ ... $p_l^{(n)}(l \mid o(n))$, we are able to compute the total PDF of the location given all observations $o(1) ... o(n)$, $p_l^{(t)}(l \mid o(1), o(2), ..., o(n))$.

To do this we formulate the first of the above assumptions and write:

$$P\{o(i) = o_1 \mid l = l_x\} \qquad (10)$$
$$= P\{o(i) = o_1 \mid l = l_x, o(j) = o_2\},$$
$$\forall \, i \neq j \; ; \; \forall \, o_1, o_2, l_x.$$

This means that the probability of the estimator with index i receiving its observations $o(i) = o_1$ under the assumption of the location being l_x is independent of the value o_2 of the set of observations of another estimator j.

It is relatively straightforward - using Bayes rule [10] and the assumptions above - to show that

$$p_l^{(t)}(l \mid o(1), o(2), ..., o(n)) = \qquad (11)$$

$$\frac{\prod\limits_{i=1}^{n} p_l^{(i)}(l \mid o(i))}{\int\limits_{-\infty}^{\infty} \int\limits_{-\infty}^{\infty} \int\limits_{-\infty}^{\infty} \prod\limits_{i=1}^{n} p_l^{(i)}(l \mid o(i)) \; dxdydz}.$$

If we are dealing with just discrete definitions of l - e.g. topological descriptions - then we can replace PDFs by probabilities $P^{(i)}\{l \mid o(i)\}$ and the integrals become sums. This also applies when we are treating a discrete grid.

2.7   Interpretation

The result (11) means that PDFs are multiplied over the location space and then the result normalised to unity in the integral. Some interesting observations can be made at this point:

1. The location PDF is forced to zero for those l if any single estimator i has a zero in $p_l^{(i)}(l \mid o(i))$ for such a l.

2. As a consequence, one must be very careful to correctly model the PDF even for unlikely l. For instance, a localised beacon transmitter might carry the ID or the co-ordinates of a room it is installed in. If this beacon is falsely installed, then it will wreck havoc with any further processing, even if it is "outvoted" by other estimators.

3. One has to be careful with the *interpretation* of topological descriptions of location.

As an example of the third point assume two estimators yielding probabilities for a set of topological locations $l = \{\text{munich}, \text{augsburg}, \text{vienna}\}$, with $P^{(1)}\{l = \text{munich} \mid o(1)\} = 0.6$, $P^{(1)}\{l = \text{augsburg} \mid o(1)\} = 0.1$, $P^{(1)}\{l = \text{vienna} \mid o(1)\} = 0.3$; and $P^{(2)}\{l = \text{munich} \mid o(2)\} = 0.3$, $P^{(2)}\{l = \text{augsburg} \mid o(2)\} = 0.45$, $P^{(2)}\{l = \text{vienna} \mid o(2)\} = 0.25$. Then finally $P^{(t)}\{l = \text{munich} \mid o(1), o(2)\} \approx 0.70$.

If, however, we bunch the locations augsburg and vienna to "non-munich", then we get $P^{(1)}\{l = \text{non-munich} \mid o(1)\} = 0.4$, $P^{(2)}\{l = \text{non-munich} \mid o(2)\} = 0.7$, and $P^{(t)}\{l = \text{munich} \mid o(1), o(2)\} \approx 0.49$. In other words, in the second case we would deduce the most likely location to be vienna or augsburg. Interestingly, the likelihood of this is not the sum of the likelihoods of vienna and augsburg for the first experiment; this is because here the two estimators "agree" most with munich, and "disagree" with respect to the other two locations.

We have so far, of course, assumed no correlation of error sources. In a distributed system it will be difficult to handle correlations between the errors of the different estimators for practical reasons, even if such correlations could be represented mathematically and be computed. Correlations which exist but are not taken into account usually result in over accentuation of the finally calculated PDF of the location. In practise, such correlations will probably pose little problems due to different positioning techniques being used and combined. Errors such as a software bug in a mobile radio system that effects base stations of multiple networks, for example, will produce such correlations, or inaccuracies of commonly used HW elements of a joint mobile radio and satellite navigation receiver.

3   Simulation

As a set of examples for five positioning sources in two dimensional space (i.e. for a constant altitude) we have used:

1. A Gaussian PDF (Fig. 3) with mean μ in two dimensions and two-by-two co-variance matrix Σ. Such a PDF may be the result of a satellite navigator, with a certain shape of the Gaussian PDF depending on the satellite constellation, noise, multi-path, etc.

2. A "shark fin" shaped PDF (Fig. 4) that represents the location PDF resulting from the reception of a sectorised mobile radio base station signal with a certain signal strength. For a strong signal strength this PDF will be more pronounced, since the likelihood is then high that the receiver is within the antenna sector.

3. A round doughnut shaped PDF (Fig. 5) that depends on the time advance given by a certain mobile radio base station (e.g. the same one as 2. above).

4. An inverted "shark fin" shaped PDF (Fig. 6) that represents the location PDF resulting from the *weak* reception of a sectorised mobile radio signal. For a very weak signal strength or no signal at all this PDF will be more pronounced, since the likelihood is then high that the receiver is not the antenna sector. The knowledge of the location and sector angle can be obtained from a different nearby mobile radio base station (e.g. the one of 2. and 3. above) which passed the

PDF in Fig. 6 as a function of the received signal strength at the correct channel.

5. A "mesa shaped" symmetrical PDF (Fig. 7) that is valid for a certain time instant (and changes over time) to represent the PDF of a person based on the knowledge that this person was definitely at a certain location at a known time and was travelling by foot (e.g. she bought a newspaper using an electronic wallet / payment system). Such a PDF would be generated and updated by the WID carried by a person and extend outwards with time.

When we combine these PDFs using (11), we obtain the PDF in Fig. 8. The sharp peak can be used to locate the person to a high degree of accuracy. Different combinations of subsets result in the PDFs in Figs. **??**. We see that not even using the first PDF (Fig. 10) can yield a useful result. An interesting case is that of Fig. 9 where we have two peaks (of different size) and thus a certain degree of ambiguity.

Map matching is easily performed, as illustrated with the floor layout of Fig. 13, and combination with the PDF of Fig. 8. The result, in Fig. 14, is now even more sharply defined than that of Fig. 8.

4 Implementation

In the following we want to bridge the gap between the theoretical concept laid out so far and a practical implementation. We want to demonstrate how easily the concept of using the PDF as a primary means for exchanging location information can be coded in a standard programming language.

4.1 Object Oriented Representation

We suggest an object oriented representation as programming paradigm for implementation of the concept. Object oriented programming languages like Java [12] are available for many platforms and are moving quickly into small devices like mobile phones, personal digital assistants or navigation receivers. The notion of containing data in objects and implementations in classes, becomes very comfortable when exchanging the rather complex data and algorithms between multiple devices over networks.This is no new idea. As an example one recognises similarities of these ideas to the concept of wavelets [11].

The definition and implementation of some data container classes is suggested to represent typical re-occurring types of data. Some of these frequently needed classes are:

PDensity is used for representation of a probability density, should have double precision, may take all values from $0.. + \infty$.

WGS84Distance is used for representing Euclidean distance in meters, should have float precision, may take all values from $0.. + \infty$.

WGS84Sigma is used for representing a standard deviation in meters, should have double precision, may take all values from $0.. + \infty$.

2DWGS84CovMatrix, 3DWGS84CovMatrix are used for representing two- and three-dimensional covariance matrices, coefficients are in meters, should have double precision, may take all values from $0.. + \infty$.

2DWGS84Point, 3DWGS84Point are used for representing points in WGS84 system (two- and three-dimensional), they include methods to get the Cartesian coordinates in meters with respect to a local Cartesian coordinate system defined by a local origin.

These classes should be kept on all the participating entities (Location Servers, Location Information Providers etc.) that deal with the raw information on location probability. Alternatively, methods to dynamically load the classes at execution time over the network can be applied, such as the Jini protocol [13].

It is the objects of certain classes that are used to store and transport the actual information on the probability of a certain location.
In order to know the meaning (semantics) of the supplied objects the supplier and user of the location information have to agree on a common interface.
Object oriented programming languages, like Java, provide a mechanism to formalize this agreement in so called *interfaces*. A simple, illustrating example is given in the following:

```
interface 2DPDFInformation {
    /** Return the probability density at the specified
     * two-dimensional point or grid in space, time or
     * time period. Depending on the input type the
     * result is a scalar probability denisty, one-,
     * two- or three-dimensional array of probability
     * densities..
     */
```

```
        PDensity getPDensity(2DWGS84Point point);
        PDensity getPDensity(2DWGS84Point point,
                             TimeStamp timeStamp);
        PDensity [] getPDensity(2DWGS84Point point,
                             TimePeriod timePeriod);


        PDensity [][] getPDensity(2DWGS84Grid grid);
        PDensity [][] getPDensity(2DWGS84Grid grid,
                             TimeStamp timeStamp);
        PDensity [][][] getPDensity(2DWGS84Grid grid,
                             TimePeriod timePeriod);
}
```

In most applications the PDF will be static in time. Nevertheless we included four methods that accept a `timeStamp` or time period to keep the interface general from the beginning.

For the 3-dimensional case the interface is similar:

```
interface 3DPDFInformation {

        PDensity getPDensity(3DWGS84Point point);
        PDensity getPDensity(3DWGS84Point point,
                             TimeStamp timeStamp);
        PDensity [] getPDensity(3DWGS84Point point,
                             TimePeriod timePeriod);

        PDensity [][][] getPDensity(3DWGS84Grid grid);
        PDensity [][][] getPDensity(3DWGS84Grid grid,
                             TimeStamp timeStamp);
        PDensity [][][][] getPDensity(3DWGS84Grid grid,
                             TimePeriod timePeriod);
}
```

We have split the interface in a 2-dimensional and a 3-dimensional one. In most practical cases a certain provider of location information will predominantly implement only one of these two interfaces. These interfaces are everything the provider and user of location information have to agree on. Every class that is supposed to conform to this standard, would have to implement them.

If the objects of these classes are to be shipped via networks they additionally have to implement the `serializable` interface in Java. The class for the Gaussian Spot, which we discussed in the theoretical discussion above, could look similar to the implementation in the Appendix B.

From that example, we can see, that representing and giving information about the PDF of a measured location is straightforward.

It should be mentioned, that from a practical point of view it is not necessary to normalize the PDFs. As the operation that combines the information of multiple PDFs is strictly a multiplication and the selection of the most probable location is performed by choosing the point with the resulting maximum. Nevertheless, from a mathematical perspective of formal correctness the PDFs have to be normalized. In fact, this normalization can help to prevent numerical problems when multiplying too small numbers with each others.

The multiplication operation cannot only be used for deciding on the correct location. We can apply it in conjunction with the PDF and an *Attractiveness Distribution Function* to choose between two potential places to receive an equivalent service at.

We have to illustrate this with an example: For every restaurant of a fast-food chain a profile is defined that quantizes its *attractiveness* or nearness from other locations. We call this profile an Attractiveness Distribution Function or ADF. All ADFs should be normalized. Typical ADFs will decrease monotonically with the distance to the center of attraction. Nevertheless barriers like a river or infrastructure that does not allow the direct path can result in asymmetric ADFs.

If we want to choose between two restaurants we multiply each restaurant's ADF with the client's PDF and integrate over the resulting distribution in two respectively three dimensions. We then choose the restaurants whose ADF resulted in the highest value from integration.

5   Results and Conclusions

We have demonstrated a new concept for representation of positions taking into account the uncertainty of the location estimate. In contrast to the state of the art were the uncertainty is not taken into account we can even use very bad position estimates (e.g. from GSM stations) to improve the position estimate of the user. Another important difference to the state of the art is the ability to give an position estimate for every case. This position estimate is proved to be optimal. This new method of a concatenated estimation of the user's position opens the possibility to use every information about the users position.

References

[1] A. Steingass, M. Angermann, and P. Robertson, "Integration of navigation and communication services for personal travel assistance using a jini and java based architecture," in *Proc. GNSS '99*, (Genova, Italy), October 1999.

[2] M. Angermann, "Navigation capabilities of future mobile communication systems – will global nav-

igation satellite systems become obsolete ?," in *Proceeding of GNSS '99*, 1999.

[3] T. Rappaport, J. Reed, and B. Woerner, "Position location using wireless communications on highways of the future," *IEEE Communications Magazine*, October 1996.

[4] Y. Zhao, *Vehicle Location and Navigation Systems*. Artech House, Inc., 1997.

[5] A. H. A. Hopper, P. Steggles, A. Ward, and P. Webster, "The anatomy of a contecxt-aware application," in *Proc. GNSS '99*, (Seattle, USA), August 15-20 1999.

[6] http://www.bluetooth.com.

[7] G. Seeber, *Satelliten Geodaesie*. de Gruyter, 1989.

[8] U. Leonhardt, *Supporting Location-Awareness in Open Distributed Systems*. PhD thesis, Thesis at the Department of Computing, Imperial College of Science, Technology and Medicine, University of London, 1998.

[9] O. Droegehorn, K. Singh-Kurbel, M. Franz, R. Sorge, R. Winkler, and K. David, "A scalable location aware service platform for mobile applications based on Java RMI," in *Linnhoff-Popien and H.-G. Hegering (Eds.): USM 2000, LNCS 1890*, pp. 296–301, September 2000.

[10] A. Papoulis, *Probability, Random Variables and Stochastic Processes*. New York: McGraw Hill Book Co., 1984.

[11] A. Graps, "An introduction to wavelets," *IEEE Computational Science and Engineering*, vol. 2, no. 2, 1995.

[12] K. Arnold and J. Gosling, *The Java Programming Language*. Addison-Wesley, 1996.

[13] K. Arnold, B. O'Sullivan, R. W. Scheifler, J. Waldo, and A. Wollrath, *The Jini Specification*. Addison-Wesley, 1999.

A   Figures

Figure 3: PDF of the location for estimator 1.

Figure 4: PDF of the location for estimator 2.

Figure 5: PDF of the location for estimator 3.

Figure 6: PDF of the location for estimator 4.

Figure 7: PDF of the location for estimator 5.

Figure 8: PDF of the location after combining all five estimates.

Figure 9: PDF of the location after combining estimates 1, 2, 3, and 4.

Figure 10: PDF of the location after combining estimates 2, 3, 4, and 5.

Figure 11: PDF of the location after combining estimates 1,2,3, and 5.

Figure 12: PDF of the location after combining estimates 1,2,4, and 5.

Figure 13: PDF of a simplified floor plan with two crossing corridors.

Figure 14: Resulting PDF using all five location PDFs and the floor plan.

B   Class GaussianSpot

```
class GaussianSpot implements 2DPDFInformation, Serializable {

    private 2DWGS84Point localOrigin;
    private 2DWGS84Point center;
    private 2DWGS84CovMatrix cM;

    PDensity getPDensity(2DWGS84Point point) {

        PDensity pDensity;

        pDensity = new PDensity(2DGaussianPDF(cM.getCXX,
                                              cM.getCXY,
                                              cM.getCYX,
                                              cM.getCYY,
                                              center.getLocalX(localOrigin),
                                              center.getLocalY(localOrigin),
                                              point.getLocalX(localOrigin),
                                              point.getLocalY(localOrigin)));

        return PDensity;

    }

    PDensity [] getPDensity(2DWGS84Point point,
                            TimePeriod timePeriod) {
        // .... implementation goes here ....
    }

    PDensity [][] getPDensity(2DWGS84Grid grid) {
        // .... implementation goes here ....
    }

    PDensity [][] getPDensity(2DWGS84Grid grid, TimeStamp timeStamp) {
        // .... implementation goes here ....
    }

    PDensity [][][] getPDensity(2DWGS84Grid grid, TimePeriod timePeriod) {
        // .... implementation goes here ....
    }

    private double 2DGaussianPDF(double a,     // cXX of covariance matrix
                                 double b,     // cXY of covariance matrix
                                 double c,     // cYX of covariance matrix
                                 double d,     // cYY of covariance matrix
                                 double e,     // mean value in dimension of x1
                                 double f,     // mean value in dimension of x2
                                 double x1,double x2) {

        return java.lang.math.exp(0.5 *
        (-(-f+x2) * (-b*(-e+x1)+a*(-f+x2)/(-b*c + a*d))
         -(-e+x1) * (d*(-e+x1)-c*(-f+x2)/(-b*c + a*d)))) /
            ( 2.0 * java.lang.math.sqrt(-b*c + a*d) *
              java.lang.math.PI );
    }
}
```