MOBVIS FP6-511051
Deliverable Report

D2.3

"Report on Representations, System Architecture, SW/HW Interfaces, and Standardisation Efforts"

Vision Technologies and Intelligent Maps for Mobile Attentive Interfaces in Urban Scenarios
Project co-funded by the European Commission
Sixth Framework Programme (2002-2006)
Information Society Technologies
FP6-2002-IST-C / FET Open
STREP

Due date of deliverable:	December 31, 2006 (revised version)
Actual submission date:	February 27, 2007
Start date of project:	May 1, 2005
Duration:	36 months

Work package	**2 – Specifications**
Task	**3 – Representation, Architecture and Interfaces**
Lead contractor for this deliverable	**UL**
Editor	**Matjaž Jogan**
Authors	**Matjaž Jogan, Dušan Omercevic, Aleš Leonardis, Patrick Luley, Tâm Huynh, Bernt Schiele, Linde Vande Velde, Alexander Almer, Lucas Paletta**
Quality reviewer	**KTH**

Project co-funded by the European Commission within the Sixth Framework Programme (2002–2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

CONTENT

1. EXECUTIVE SUMMARY

The objective of work package 2 is to define and provide the technical infrastructure for the multi-sensor based context extraction, including camera and multi-sensor equipment, and the mobile client and server infrastructure used in the different demonstration scenarios. It develops a common view on relevant representations, defines the functional relations between mobile system components, and agrees on associated SW interfaces. It also characterizes typical use cases and scenarios.

This report summarises the activities related to task 2.3, which is oriented towards two sub-themes:

- an early agreement on the choice of representations for visual features, objects, map format and context and event descriptors
- a draft specification of the functional outline of the system architecture, determining the HW/SW interfaces and a detailed outline between the specific interdependencies between the technical tasks.

The outline of the report is as follows. We briefly describe the requirements for representations of map data, objects, features and events in the context of efficient manipulation and cooperative exchange of data. In the second part of the report we describe a functional outline of the system architecture which represents a baseline for cooperative work and further development of demonstrators.

This task will result in an agreement on the basic architecture and functionalities of the multi-sensor setup and the mobile client and server infrastructure which will be realized in several demonstrator setups. Since the integration of such a system is a complex task which involves development of mobile clients with multi-sensor processing capabilities combined with distributed communication and data processing on remote servers, an early agreement on the basic outline of the system with its subsystems, the software and hardware interfaces, and the middleware (glue), is a first step towards the development of demonstrators in MOBVIS. As specified in the project proposal, the MOBVIS system will iteratively increase complexity in the interaction between the various system components. This document has therefore a status of a living document and will be updated during the project.

2. SPECIFICATIONS ON REPRESENTATION

The key issue in delivering specifications that define the representation of data and conceptual entities in MOBVIS system is how to ground object/event/context representations in a way that eases integration without too many constraints for future research. The representations should ease the exchange of data and experimental results between partners and should support to the overall flow of information in the system.

The basic questions to answer are how to represent features, how to represent objects, events and context and multimedia content and how to represent these entities in the framework of a geo-coded database.

2.1 Map

This chapter will give a general overview of the state of the art of current available static and dynamic TA map content, static and dynamic multi media content and how this content is currently linked. This chapter contains also information on the standard Tele Atlas mapping content and the Tele Atlas map formats which will be used in the MOBVIS project.

2.1.1 STANDARD TA MAP CONTENT

The provision of digital cartographic data is essential for all of today's navigation systems. Without them, the knowledge of one's position would be all but useless. The generation process of maps involves extensive fieldwork with teams of cartographers literally driving millions of kilometers each year, creating, testing and updating data. Typically, the centre line is used to represent a road, with a series of "shape" points. Red points designate intersections and other significant points. Simultaneously, other employees analyze existing maps, city plans and aerial, satellite photographs and image sequences captured with mobile mapping vehicles in order to build up the greatest possible quantity of information. This information is then combined and digitized. The final result is a digital map that does not just contain the roads themselves: the type of road, its direction, bridges, tunnels and other topological details are also included. Moreover, each section of road can also contain over 150 attributes; these include data such as speed and weight limits, turn restrictions, addresses and also POIs (Points Of Interest: restaurants, hotels, hospitals, petrol stations etc.).

The final maps have accuracy within 5 m and 10 m in urban and rural areas respectively. This level of accuracy is sufficient as the reduced visibility of satellites in urban areas- as a result of the "Urban Canyon" effect- reduces the accuracy of the positioning system anyway. The greatest problems lie not in their accuracy but in their distribution. Currently digital maps are stored in-vehicle on a CD-ROM or DVD. This has always presented difficulty with respect to updating data. Although the map provider may have access to very recent data, the user will typically not change their data disk more frequently than every six months. Hence, the data used can often be in the worst-case scenario incorrect or in the best-case scenario partially out of date. The data component that is least likely to change in the intervening period is that of road layout. Far more likely, is that the attributes are going to change, but these can include direction and speed restrictions, a rather critical part of pre-trip route planning or on-trip route navigation.

This chapter will give an overview of where to find information on the standard digital map content and standard formats which will be used as input in the MOBVIS demonstrator.

This chapter will also describe the use of the GeoEngine mapping application programming interface (API), used to develop mapping applications. GeoEngine offers a full solution. GeoEngine supports all functional areas related to navigation and location based services applications, such as map display, route planning, geo-coding and reverse geo-coding. GeoEngine has a proprietary protocol between server and client, which is optimized for fast access and good performance. Using the GeoEngine sample code allows the developers to develop a new application in a rapid way, and the resulted map application has a very good performance.

2.1.1.1 GEOGRAPHICAL CONTENT

The geographical content consists out 2 parts:

- Standard MultiNet

- 2D City Maps – produced in the framework of the MOBVIS project.

The specifications of all these features are stored in two documents:

- MultiNetDataModel3.4.1 (800 pages) – Extensions for MOBVIS

This document contains for all topics described in the Tele Atlas Database Data Specifications, the way of storing them in the MultiNet Product. Data Storage in MultiNet is according to the GDF Data Model described in GDF 3.0.

- TA MultiNet TM version 3.4.1 Data specifications (1600 pages) – Extensions for MOBVIS

This document contains the Standard Data Specifications of the Tele Atlas MultiNet. It also included already the enhanced specifications specified for the MOBVIS project.

The MultiNet Standard Data Specifications describe which real world information should be captured in the Tele Atlas MultiNet and how the different information components are mutually related. These specifications describe an optimal database produced as Street Network coverage, as Interconnecting Network coverage or as Major Road Network coverage, all containing a maximum amount of information.

The actual information contained at a certain moment in the product can be less than the one described in the Standard Data Specifications (depending on certain projects, etc…) However, the actual content of the product is not meant to be described in the Standard Data Specifications but in the Database Coverage and Contents Specifications which is released per country and per product release.

2.1.2 STANDARD TA MAP FORMAT

2.1.2.1 SHAPE FORMAT

The Shape File Format will be used as input map format for the first MOBVIS demonstrator.

All details on the Shape File Format can be found in MultiNet™ Shape File 4.3.1 Format Specifications. Tele Atlas MultiNet Shape File Format offers the complete Tele Atlas database contents structured according to a ready-to-use layered data model and in a standard GIS format. It is available in a run-time data exchange format that is easy to implement into any GIS-based application.

MultiNet Shape File is designed for direct use with standard GIS software and tools and is optimized for fast cartographic display, accurate geo-coding and rapid optimal route calculation. It is available in Shape File Format for Western Europe and the United States.

2.1.2.2 MAPACCESS FORMAT

MapAccess format will be used as input map format for the second demonstrator.

Information on the standard Map Access format can be found in Tele Atlas MultiNetTM (USA) MapAccessTM Format 4.0

The MapAccess format for MultiNet™ databases is a highly compressed file format designed for fast access to and display of map information. This document describes the MapAccess format for MultiNet databases. This document is intended for system integrators who are developing geographic applications.

MultiNet databases in MapAccess format (called MapAccess databases for simplicity) contain road network data in a highly compressed file format that allows applications to quickly find and display database information. MapAccess databases are a source of road and landmark data for map applications; they are not a source of geographic boundaries. MapAccess databases are typically used in geographic applications developed with MapAccess Development Tools (see the next section). They are well-suited for applications that determine spatial relationships, display maps, or print maps. Accompanying the MapAccess databases are GeoIndex databases, which contain complete address information. Applications can use a GeoIndex database to determine the map locations for street addresses, and the addresses for latitude/longitude coordinates. The MapAccess format provides current address information and includes major landmarks, one-way street encoding in selected urban areas, richer graphics, shape points, and other enhancements.

2.1.3 MOBVIS SPECIFIC TA MAP CONTENT SPECIFICATION

2.1.3.1 ENHANCED CONTENT

Enhanced map content will be produced in the framework of the MOBVIS project and used as input for the MOBVIS demonstrator in order to fulfill the MOBVIS user requirements. Details on those specifications can be found in the:

- MultiNetDataModel3.4.1 (800 pages) – MOBVIS extension
- TA MultiNet TM version 3.4.1 Data specifications (1600 pages) – MOBVIS extension

The enhanced content will include both static and dynamic content. Static content will consist mainly of features and objects which relate to pedestrian navigation. This relates mainly to urban signs and infrastructure which is not part of the standard content and to salient landmarks (buildings, parts of buildings, other elements that support navigation). Other static content is mainly scenario dependent and will be defined in detail during the project.

Dynamic information contains different types of content. It can refer to the location of people and cars, but can also be the momentary location of field hospitals, movement of fire, etc… Location information of pedestrians, vehicles etc. can be provided by GPS information captured with the mobile devices connected to a GPS device. Location or movements of firewalls can be indicated by the people in the field, by speech, text, or by a sketch. This kind of dynamic information can be mapped on top of the geographical information, as soon as meta-information on the position and direction of the location and or movement or the object is given.

2.1.3.2 2D MAPS

2D City Maps is the database representation of a number of map display entities related to the contours and outlines map of visual and non-visual elements. *2D City Maps* contains 3 types of map display data: Town Blocks, Town Buildings, and Town Railways. Each type consists of an area representation and a line representation, resulting in 6 entities. For Town Blocks and Town Buildings, the line representation indicates the details that can not be drawn as a polygon. No height information is available for the 3 types of map display data.

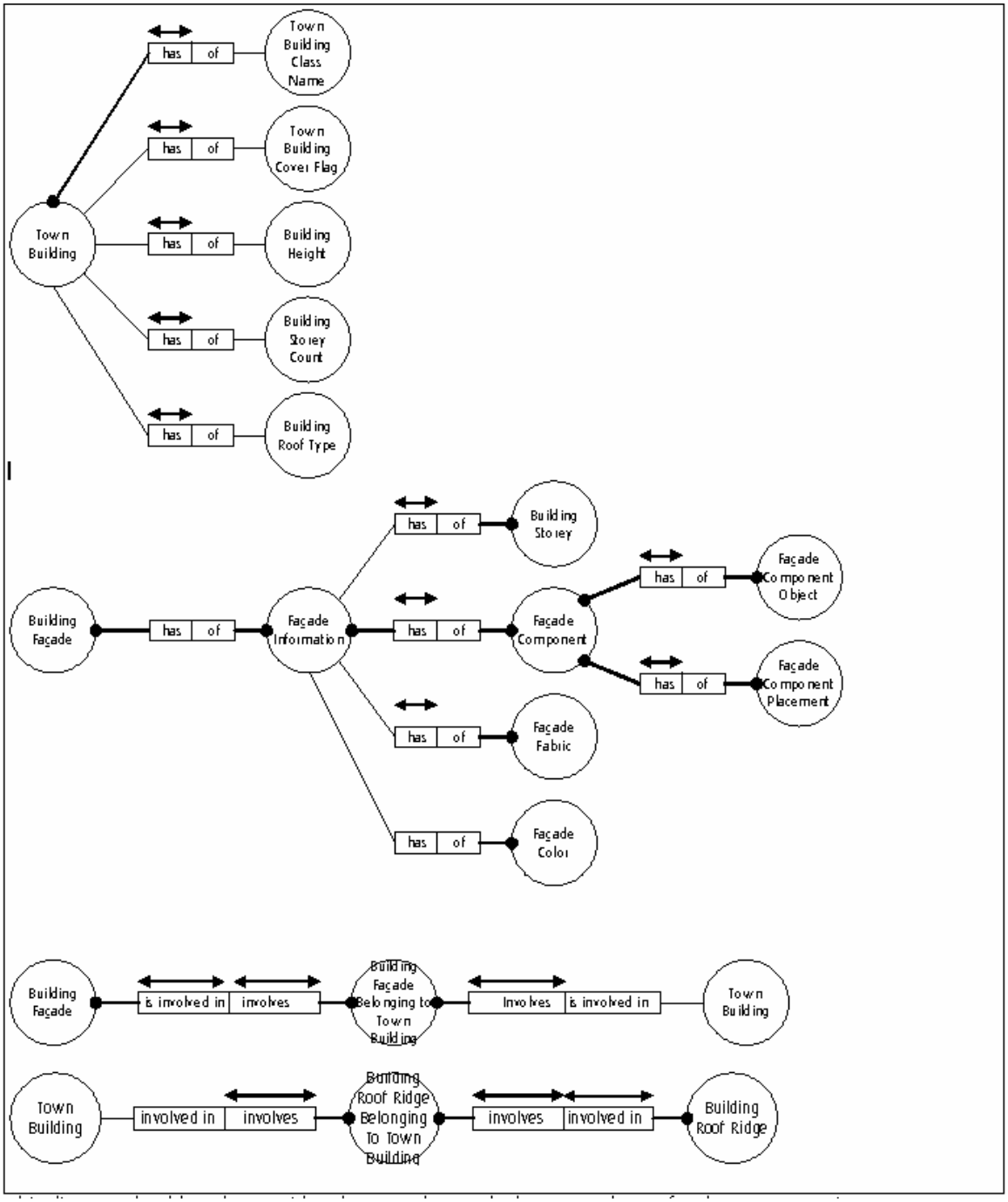


Figure 1: Data model for 2D and 3D maps.

A part of the Graz old city centre was selected as the test area for which TA will acquire the data for the first demonstrator. Figure 1 depicts the area and the paths which will be travelled by the acquisition van.

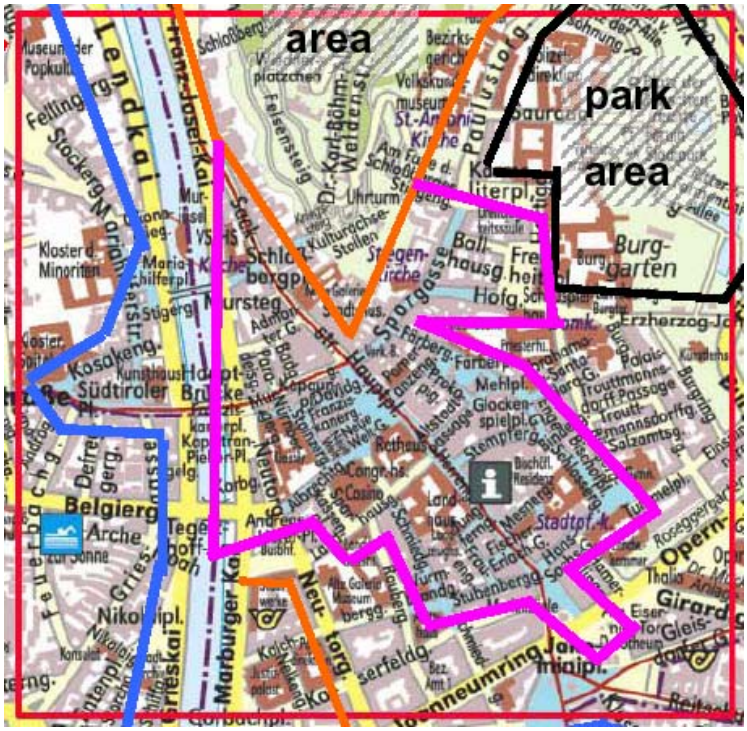


Figure 2: MOBVIS 2D City Map Prototype test area (Graz, Austria).

2.1.3.2.1.1 3D Maps

Basic 3D City Maps is the database representation of a number of map display entities related to the contours and outlines map of visual elements including basic height information. Basic 3D City Maps contains 3 types of map display data including basic height information: Town Blocks, Town Buildings, and Town Railways. Each type consists of an area representation and a line representation. For Town Blocks and Town Buildings, the line representation indicates the details that can not be drawn as a polygon. Height information is available for Town Buildings although it does not represent specifics on the façades of buildings.

2.2 Objects

Objects are the most common semantic primitives in artificial sensing. In MOBVIS, representation and recognition of perceptual objects is an important functionality of the attentive interface. Here we review different types of perceptual objects and discuss ways of representing them.

2.2.1 TYPES OF OBJECTS IN MOBVIS

Each singular use case in MOBVIS will orientate towards addressing a different subset of objects which relate to specific tasks related to pedestrian navigation in an urban area. Roughly, two categories of objects can be defined according to their spatio-temporal nature:

- Static or Geo-referenced objects include buildings, landmarks, urban infrastructure, road signs
- Dynamic objects which change position (cars, buses, people, faces, company brands on billboards…)

Both static and dynamic objects can be expected at a number of different layers. Several categories can be defined which are dependent on a specific scenario. Categories can start at a high level of classifying the sky, buildings or vegetation. At a lower level one can expect categories such as buildings, vehicles, signs. Unique instances can either be derived from

category descriptions and augmented with entity-specific information, or can have a representation which is independent from any of the category representations. Most of the urban infrastructure elements such as ‘zebra crossing’ and ‘news stand’ can be efficiently represented as instances of a certain category without specific features apart from their coordinates. A landmark building, e.g. the Kunsthaus in Graz, will be represented as a specific entity and referred to as ‘Kunsthaus’, however its representation can be either unique or rather derived from the more general representation of a ‘building’.

Further, objects can be considered at different levels of granularity, e.g. objects declared as buildings can have a facade and several windows, which are also objects.

2.2.2 TYPICAL OBJECT DESCRIPTIONS

Objects are described in terms of features which can be extracted or otherwise sensed as being parts of, connected to, or describing a certain instance of an object. It is still an open problem how to represent objects in order to allow for good categorization/ recognition results and it is part of the MOBVIS project to advance the knowledge of multimodal object representation for categorization and recognition. Besides being designed for recognition, a representation should be also easy to construct (learned), easy to update (open-ended), and efficient to store.

Representations involve visual features, structural properties, and multimodal instances such as sound or motion or higher level descriptors, such as functionality, emotional or tactile properties.

The nature of object models typically determines also the perceptual, learning and recognition algorithms. Following several theories on representation, the major directions which were considered in the past, mostly in the field of computer vision are:

- Generic (category based) versus exemplar based representations
- Object centred versus viewer centred representations
- Shape based and appearance based representations
- Representations based on global features and local features

In general, these approaches try to answer the following questions:

- How is visual knowledge of an object encoded?
- What information is recovered from an image to recognize an object?
- How is this information compared to the stored knowledge?

Generic representations usually store a prototypical model which represents a category of objects, usually describing structural (3D) properties, but also visual features which are common to members of a category. The prototypical model is usually a rather abstract description of a category. In contrast, exemplar based representations build their model on a number of examples of category members.

Object centred representations are modelled as being independent on the observer – they are typically based on geometrical 3D models, while viewer centred representations model the objects as being observed by a subject. In computer vision, views are essentially projections from 3D to a 2D plane; viewer centred representations therefore store e.g. a number of typical images of an object.

Objects can be described in terms of their reflectance properties (appearance), in terms of their 2D or 3D shape or with associated multimodal data. When extracting a description from the sensor data one has to select the interesting elements to include in the representation.

These elements – features – can be either global (a property derived from the whole object or scene) or local (a property derived from a local area on the object or in the scene).

Object recognition (recognizing a previously seen object) is considered to be easier than object categorization (recognizing an object in terms of some category (visual, functional…)).

An efficient category representation shall encompass properties that relate to most of the members of a category. In doing this, generic representations are more efficient than exemplar ones. A category representation often encodes the properties at a much higher level, while a representation used for recognition can directly include low level properties, e.g. reflectance. It is therefore clear that a careful choice of representation is crucial for any efficient technical solution.

2.2.3 VISUAL FEATURES

Apart from global features such as image statistics and frequency transforms, visual features can be defined as local, meaningful and detectable parts of the image. Typical examples of visual features are sharp variations of intensity or regions with uniform colour, as they typically reflect some physical properties of objects in the scene (edges, planes...).

Since the *contours* in the image typically delineate objects from the background, detection of boundary points (points with a surrounding where the intensity values undergo a sharp variation) is often used for finding *edges* (connected chains of edge points). Edges are detected using algorithms which are based on filtering with specialized filters.

Another category of commonly used features is *corners*, which can be defined as points in centre of a corner-like structure of intensity. Corners can be found on contour intersections. However they often do not have a meaningful interpretation in the physical phenomena. Corners can be found e.g. via analysis of a matrix which describes the surrounding of an image point.

Important features are *straight lines* which often reflect informative contours of man-made and urban objects and can be found using e.g. the Hough transform. Other interesting geometrical features which can be modeled parametrically, sudh as circles and ellipses can also be detected using the Hough transform.

2.2.3.1 LOCAL DESCRIPTORS

A specific type of image features are descriptors which are computed for local interest regions (ROI). ROI can be defined by simple mechanisms, e.g. regions centred on corners. However, if a high repeatability rate of detection is needed, only regions covariant with a class of transformations give good results. This ensures that regions can be detected even when objects in the scene undergo significant changes in e.g. scale, shape, or viewpoint change.

In a thorough comparison, Mikolayczik et. al. [1] compare six types of **ROI detectors** which are covariant to affine transformations. The first pair of detectors uses affine normalization around Harris corners (Harris affine) and Hessian points (Hessian affine). The remaining four detectors operate with maximally stable extremal regions (MSER), edges (EBR), intensity extrema (IBR) and entropy (salient regions). Since the document includes most of the state-of-the-art approaches, it can be used as reference. The authors evaluate ROI detectors with respect to their computational complexity, the density of detected regions, the average size of the regions, and to repeatability. A summary of the results is as folllows:

- Computational complexity
    - Harris affine: *O(n)+O((m+k)p),* where *n* is image dimensionality, *p* denotes the number of points found by Harris detector, *m* the number of investigated scales, and *k* the number of iterations in shape adaptation,
    - Hessian affine: *O(n)+O((m+k)p),* where *n* is image dimensionality, *p* is the number of points found by Hessian detector, *m* the number of investigated scales, and *k* the number of iterations in shape adaptation,
    - MSER: *O(n)+O(n*log(log(n))),* where *n* denotes image dimensionality,
    - EBR: O(n)+O(pd), where *n* is the image dimensionality, *p* the number of corners, and *d* the average number of edges/corner,
    - IBR: *O(n)+O(p)*, where *n* is the image dimensionality and *p* is the number of intensity extrema,
    - Salient regions: *O(nl)+O(e),* where *n* is image dimensionality, *l* is the number of ellipses investigated at each pixel, *e* is number of extrema detected in the first step.
- Region density
    - High (1000-2000) for Harris affine, Hessian affine, EBR and low (500-1000) for MSER, IBR and salient regions in an 800x640 image. Density strongly depends on the scene type so these numbers can vary significantly. As the detectors respond to different types of visual structure, they can be considered as complementary.
- Region size
    - Harris affine and Hessian affine yield more small regions on the average. Since the size of the regions strongly depends on the input image (texture, scale), it can not be evaluated for a general case.
- Repeatability
    - Under viewpoint change (best: MSER),
    - Under scale change (best: Hessian Affine, MSER, Harris affine),
    - Under blur (worst: MSER),
    - JPEG artifacts (best: Harris affine, Hessian affine),
    - Ambient light change (best: MSER).

Local interest regions are a basis for the computation of **local descriptors**. State of the art descriptors (also evaluated by Mikolayczik et. al. [2]) are shape context, steerable filters, PCA-SIFT, gradient location and orientation histograms (GLOH), differential invariants, spin images, SIFT, complex filters, moment invariants and cross-correlation of image intensities. The parameters of local descriptors to evaluate are:

- Dimensionality
    - The derivatives-based descriptors can be computed up to an ordinary order. Typically the dimensionality of these descriptors is from 5-15 dimensions. The dimensionality of the SIFT descriptor is typically 128; GLOH and PCA-SIFT

dimensionality depends on the numbers of eigenvectors used (typically 50-128); cross correlation can go up to 400 and more.

- Recall
    - Under scale change (best: GLOH, SIFT, worst: complex filters, differential invariants)
    - Under rotation (best: GLOH, SIFT, shape context)
    - Under viewpoint change of more than 50 degrees (best: GLOH, SIFT)
    - Under image blur (best: GLOH, PCA-SIFT)
    - JPEG artefacts (best: SIFT, GLOH, PCA-SIFT)
    - Illumination changes (best: GLOH)

The overall storage capacity depends on the number of regions to be described and the length of the descriptor. Here we provide only approximate numbers for the case of MSER as the region detector and SIFT as the region descriptor.

Representation	Size	Storage
JPEG image	1600x1200 pixels	620KB on the average
SIFT descriptors of MSER regions (ASCII encoding)	1600 regions on the average	620KB on the average
SIFT descriptors of MSER regions (binary encoding)	1600 regions on the average	200KB on the average

Table 1. Image size and storage requirements for local descriptors.

2.2.3.2 DESCRIPTORS FOR CATEGORIZATION

Descriptors described herein have a high recall rate and are good for describing and recognizing specific objects. If is however disputed whether they can be efficiently used for categorization, where one typically aims at representing features which are shared across objects in a category. Representations for image categorization can be therefore constructed by using a set of basic features in an architecture where these basic features are being combined to represent more complex structures. Examples of such frameworks are e.g. boosting or hierarchical representations.

2.2.4 RECOMMENDATIONS FOR MOBVIS

The vision part of the MOBVIS system will in a great part operate with local photometric descriptors which act as representations of regions of interest and are appropriate for the task of retrieval. To find the regions of interest (ROI) in images, we shall consider interest point detectors that can detect regions which are characteristic and informative. Detectors can implement several invariant properties such as scale invariance, rotation invariance and affine invariance. The selection of ROI detectors will be made according to specific demands on invariance and repeatability rate. In particular, the descriptive power for urban scenery should be evaluated. Several types of photometric descriptors that are invariant to image transformations (change of viewpoint, illumination…) shall be considered. The selection of descriptors shall be made according to demands for high repeatability rate (for retrieval) and invariance. Further, different descriptors might be more or less suitable for searches in massive databases. Most likely scenes consist of, not just the features seen in a single

image, but from a large set of images. Discriminance as a function of dimensionality becomes an important factor in such cases.

According to the evidence in the literature, Harris affine, Hessian affine and MSER in combination with GLOH, SIFT and PCA SIFT are efficient in a number of different scenarios. This does not imply a straightforward adoption of these methods, as their performance in typical MOBVIS scenarios has to be critically tested and analyzed.

For representing categories such as pedestrians, windows etc., simple features such as Haar wavelets can be used in combination with boosting. Other methods will be tested and developed during the project.

2.2.5 REFERENCES

[1] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L. A comparison of affine region detectors. Accepted in International Journal of Computer Vision – 2005.

[2] Mikolajczyk, K., Schmid, C. A Performance Evaluation of Local Descriptors, PAMI(27), No. 10, October 2005, pp. 1615-1630.

2.3 Multi-Modal Context

Context and context-awareness are often seen as key-elements in ubiquitous, wearable and mobile computing. Within the MOBVIS project we will concentrate on context information relevant for the identification of situations, activities, and locations specifically for an urban pedestrian scenario.

2.3.1 LAYERS OF ABSTRACTION

There exist various levels of abstractions for context information, from the raw sensory information from individual sensors, to high-level concepts that have been abstracted from the information from various sensors:

- **sensor-level**: basically the raw sensor data ranging from single sensor readings to streams of sensor data,
- **feature-level**: extraction of relevant features from the typically continuous sensor data stream,
- **sensor fusion**: fusion of various sensor modalities (on the feature level),
- **context extraction and abstraction**: modelling and recognition of particular context information such as location or user activity which enables to abstract away from the particular set of sensors employed,
- **context-fusion**: fusion of various context information for example to trigger particular system actions.

To relate this to MOBVIS deliverable 4.1, where in section 2.2 we present several (layered) architectures similar to this model, we opt for treating these layers of abstraction to be better able to fuse across modalities. Architectures such as the widget-based approach have less obvious opportunities to combine abstractions from various modalities. Additionally, it allows a practical bottom-up approach where sensor information is processed in increasing levels of complexity, towards contexts that are required by the application.

Clearly the higher levels of abstraction are very important to make context information usable and useful within the scope of the project. In order to achieve these abstractions various levels of representations and recognition methods need to be developed. One particular type of context information is event-based where events can be used directly to trigger system actions. As outlined in MOBVIS Deliverable D6.1.1, we will dedicate a specific component within the Attentive Interface to handle the generation of context information, i.e., the Context Interpreter (CI). As described in detail within D6.1.1, we will propose a recursive definition to outline a hierarchically structured context architecture.

2.3.2 MOBVIS USAGE AND EXAMPLES

Within the scope of the project we concentrate on multi-modal context information. Therefore representations on the feature and context level, which allow effective context information, as well as the raw sensor data level will be important. For these three levels of abstraction, we give the most prolific examples envisioned within the MOBVIS project:

- Sensors: these can be categorized under three types of sensors: location sensors which allow estimates of position (such as GPS, WiFi, GSM, etc.), inertial sensors (for example accelerometers) which monitor physical motion of the user, and vision.

- Features: most of the raw data from the sensors layer is pre-processed according to often well-known and established features. For MOBVIS specifically we have for instance: local/global visual features, speed, direction, motion patterns, and Cartesian coordinates.

- Context types: location, activity, time and (social) interaction. According to scenarios, one could envisage types of locations at different levels of abstraction, e.g.: map coordinates, street address, streets or zones, neighborhoods, parks, shops, shopping malls, stations, or landmarks. Activity might include commuting, shopping, taking a touristic city tour, going for a recreational walk, or even more generic levels of activity, such as socialization and solitude.

While location based sensing (LBS) will provide a most obvious type of context, various other types of sensors will be investigated for providing useful context information. Examples include user activity sensed via inertial sensors or audio signatures from the environment that would support place or situation recognition. Similar to the object types we can differentiate between public and private context information as well as between static and dynamic context information.

Within MOBVIS, particularly these modalities have a defined role:

- Vision: will be primarily used for location context in terms of geo-localization and categorization (e.g. geometrical context), and for priming other recognition/categorization tasks.

- Inertial sensors: a particular type of context that will be investigated is based on inertial sensor information. In particular this can be used to estimate velocity, direction and distance of movements which in turn can be used to recognize user activity as well as an additional modality for location estimation. The inertial sensors also allow the recognition of patterns for specific activities.

- Location sensors: particular dedicated sensors, of which GPS is the most well-known, are able to give an immediate estimation of position, and thus should be considered

for priming any task where location might improve the search space.

It is important to note though, that in many cases these are expected to have a complementary role, the location sensors can for example reduce the search space of recognition tasks within the vision module, or information from the vision modality can enhance the recognition of the user's activity. Although we specify these three modalities (and associated contexts) as the main focus for MOBVIS, we do intend to include these cross-modal benefits in our evaluations.

An important characteristic of context information is that most context recognition will not be perfect. Therefore one can also involve the user in determining the context within a semi-automatic procedure, e.g., by notifying the user about a situation and requesting to give input. This requires high level of abstractions of context information, which can be used and understood by users. Here, we do not intend to explicitly address the concrete contexts that will be possible to distinguish in a final demonstrator, our focus is foremost on location descriptors and activities that are mentioned in MOBVIS deliverable D2.1. Concrete examples will be described within MOBVIS Deliverable D6.1.1 and D6.1.2.

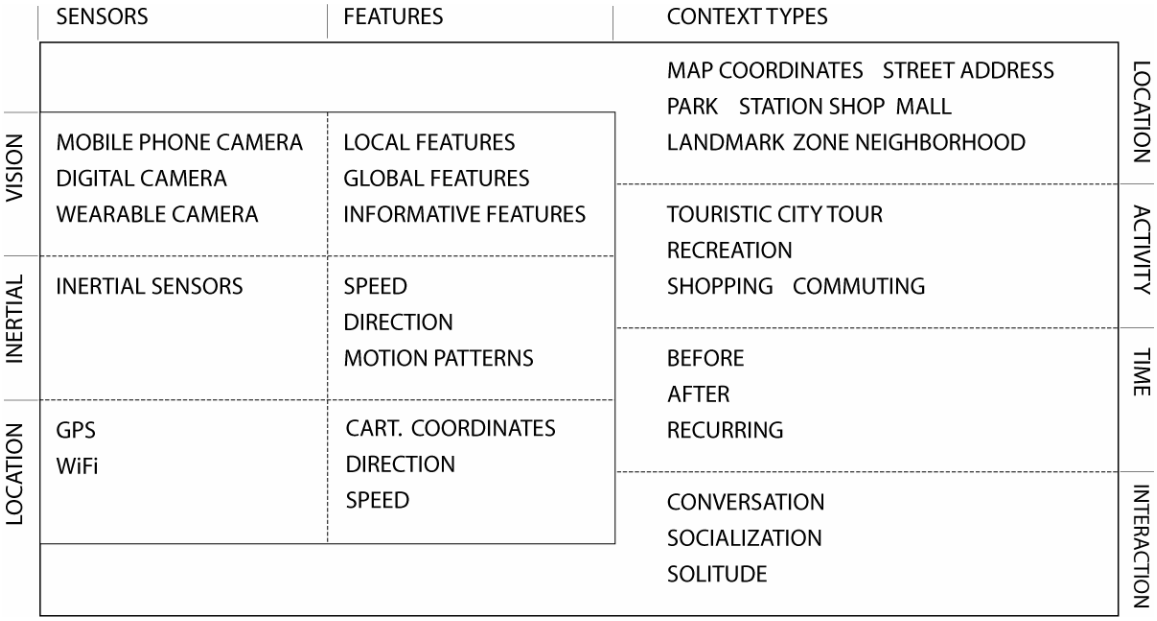


Figure 3: Modalities and abstraction levels of context in MOBVIS. Because of the complementary role of modalities, an interplay of context information is possible at all levels of abstraction.

2.3.3 Preliminary Evaluation from User Experience

For first examples of multi-modal context use in MOBVIS we will evaluate the use of different sensor types (GPS, WLAN for absolute positioning, Dead Reckoning for relative positioning) in several scenarios in Darmstadt as well as in common scenarios in Ljubljana. This offers the possibility to evaluate the mutual advantages of the combination of multimodal location based and vision based positioning techniques. The scenarios will concentrate on location awareness in environments typically frequented by pedestrians, where, as preliminary studies show, available location sensors alone often fail to provide a reliable stream of position estimates which is needed for attentive processing. Special attention will be given to the suitability of priming for the vision task, and achievable accuracy and availability of the different sensor combinations in a realistic scenario. Another aspect of the evaluation will be the possibilities of initial localisation by single snapshots with only coarse priming by other sensors (Part-of-city level), and the continuation of the refined position with relative positioning information. As initial localisation is done by matching the local visual features of

the input image with features in the geo-referenced database, the location context from the Context Interpreter will be used to limit the search space.

In terms of activity recognition, we plan to investigate the use of wearable accelerometers to recognize common activities that users are likely to perform in urban scenarios. We will record a range of such activities and evaluate the performance of different machine learning algorithms with respect to measures such as recognition accuracy, efficiency and robustness.

2.3.4 RECOMMENDATIONS FOR MOBVIS

The main reason for using multiple modalities of context within MOBVIS, is, firstly, the need for a large spectrum of information for redundant descriptions of situations, and secondly, in order to enable the Attentive Interface to use results from one modality to attend – or index - into the information of a complementary modality (Task 6.1).

Three categories of modality have been identified for MOBVIS: Vision, Inertial Sensing, and Location Sensing, which give a clear separation in what type of data is being captured, but since we expect them to be complementary for various of MOBVIS' relevant aspects of context, a layered context architecture is preferred. It is thus possible to combine the data from the different modalities at feature level, to obtain contexts more reliably. Details on the structure of the layered context architecture will be described with respect to the Attentive Interface in MOBVIS Deliverables D6.1.1 and D6.1.2, respectively.

2.4 Representation in the Spatial Database

The spatial database stores all geo-referenced data and it can be implemented by any of the spatially enabled relational database management systems (RDBMS). Objects and features with a known geo-location can be stored and referenced to by their coordinates. In a standard geo-spatial database such an indexing enables most of the required functionalities.

It is important however to realize that due to a large amount of data stored in the MOBVIS spatial database, a more efficient retrieval mechanism is needed. In particular this refers to the nature of the representation (as mentioned before in 2.2.2): an object centred versus a viewer centred representation.

The standard way of representing objects as linked to their 3D coordinates results in an object-centred representation, since it does not describe the properties of an object as they can be observed by the viewer. In contrast, a viewer centred representation stores descriptions of objects and features as they can be seen from a certain viewpoint. To give an example, descriptors like SIFT are only insensitive to changes of around 40 degrees. Thus, in order to represent a facade in a way to support recognition from an arbitrary viewpoint, the model has to contain descriptors as seen from a large set of viewing angles and distances. If these descriptors are linked to the information on the viewpoint direction, we can access at each position a viewer-centred representation. Indeed, if we want to recognize an object, the retrieval of such a representation is much more useful than having a representation which does not provide view-related information. Another useful aspect of viewer centred representation is that not all features or objects have a known geo-location. Since they were observed from some viewpoint, they can be associated only with the coordinate (or rather a visibility area) and direction from where they were observed.

An object centred representation can either incorporate all the information related to multiple views (e.g. a full set of possible descriptors for an object) or a single model (e.g. a 3D model). It is possible to make the object centred data viewer centred if it is encoded properly.

This can be done by estimating the visibility region related to a certain viewpoint and by depth culling. Of course, the viewing direction has to be known in order to estimate the visibility regions. The viewing direction can be approximated either from navigation sensors or by computer vision methods. In this perspective, a coarse categorization of the momentary view direction is a useful functionality of the attentive interface.

The position of the viewer can be calculated with the application of projective geometry. In order to do this, a number of (object centred) features with known geo-locations is needed. Another possibility is to process multiple views of a certain object.

In summary, the nature of the database shall be stressed in the light of a certain scenario, the types of objects, and the functionalities of the demonstrator. It is however expected that both object and viewer centred representations will be used. A possible indexing schema is shown in Figure 4. Areas shaded in different colours illustrate visibility areas. E.g. a city square is a large area from which most of the features on the surrounding buildings can be seen. These features can therefore be linked directly to the area instead to a certain geo-position.

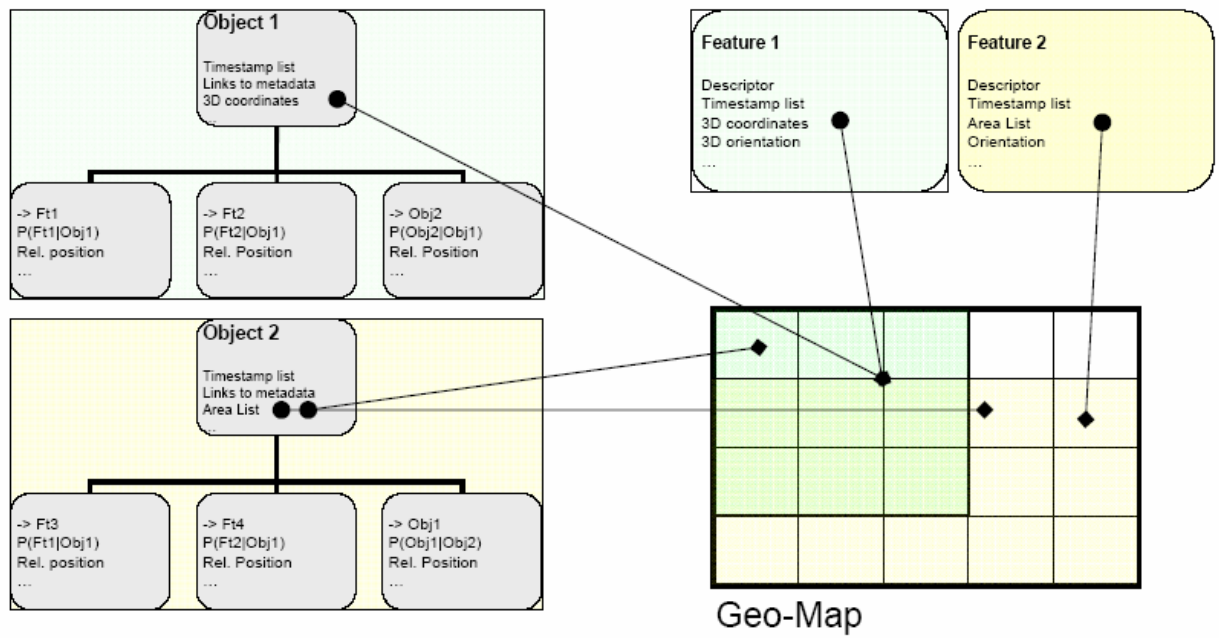


Figure 4: A hypothetical outline of the spatial database.

3. AN OUTLINE OF THE CLIENT-SERVER ARCHITECTURE

In the process of refining the first draft description of the MOBVIS system an early decision was made on the client-server nature of the system. The system consists of a mobile device on the client side (mobile phone, PDA, tablet PC, wearable computer), equipped with sensors and GPS/UMTS/WiFi/WiMax connectivity. The client communicates with a server side which provides intensive computation functionalities, access to the centralized intelligent map and most of the storage of geo-coded multimedia data. In this first draft we ignore details which will emerge when specifying different use cases which will involve specific demands, such as having several instances of map and data collections (public/user/private) and where hardware/software resources will be distributed between the server and the client according to the requirements of each of the scenarios separately.

3.1 REQUIREMENTS

In this section we briefly analyse what are the requirements for a MOBVIS demonstrator system. The functional requirements describe what the system should do, while non-functional requirements describe properties of the system per se.

3.1.1 FUNCTIONAL REQUIREMENTS

Functional requirements for a MOBVIS demonstrator system are:

I. The user shall be able to use the services provided wherever there is a third generation wireless network connection available.

II. The system shall keep track of users positions at all times as accurately as possible.

A. The user's mobile device shall provide the system with the position information available by positioning systems (GPS, Galileo, radio beacons, inertial sensors, etc.).

B. The system shall be able to utilize location information provided by wireless network operators.

III. The system shall be able to collect multimedia data about the mobile user's environment.

IV. The system shall have an internal representation of the mobile user's environment and preferences.

V. The system shall be able to augment the internal representation of the mobile user's environment with information extracted from collected multimedia data.

VI. The system shall enable an interaction and collaboration among the users of the system. The user shall have a full control on which of his private data can be shared with others.

VII. Based on the internal representation of the mobile user's environment and preferences the system shall provide the mobile user upon his request with smart mobile vision services.

3.1.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements for a MOBVIS demonstrator system are:

I. The system shall ensure that data is protected from unauthorized access.

II. The system shall be built on open standards.

III. The system shall be developed with a cross-platform operation in mind.

IV. Due to a non-critical nature of the service provided high reliability and availability of the service shall not be a priority.

V. The system shall be divided into a server-side and a client-side. The server-side of the system shall never be an initiator of a session with the client-side. There are two issues supporting this requirement, security and addressing.

A. Security. Due to security concerns most of computer networks prevent access to network nodes from outside of the network.

B. Addressing. Due to a limited addressing space most of computer networks utilize some sort of a network address translation, thus preventing the initialization of a session from outside of the network. This issue will only be resolved with a proliferation of IPv6 into everyday networking.

3.1.3 USABILITY REQUIREMENTS

The mobile user shall be able to use the services provided by the MOBVIS system on a mobile device. Usability requirements stem from limitations of mobile devices. These limitations are:

I. limited battery power

Both, a wireless transmission and a client-side data processing consume a limited battery power of the mobile device. The system architecture should reflect a trade off between a server and a client-side data processing. Data processing modules should be designed in such a way that they can be easily moved from the server to the client side and vice versa.

II. data transmission costs

For foreseeable future wireless data transmission costs will not be negligible. Therefore the amount of data transmitted should be kept low.

III. small screens and limited user input

Small screens and limited user input possibilities of mobile devices limit user interaction. These considerations do not present considerable limitations on the system architecture and should be addressed in other documents of a project specification.

IV. Start-up time

A mobile user has an intense interaction with his/her surrounding. An application that is not available instantaneously is deemed useless by the user. An exact limit on the maximum start-up time acceptable to the mobile user should be agreed on in further usability studies.

V. response time

The same arguments valid for the start-up time stand also for a response time. An acceptable maximum response time should also be determined in the further usability studies.

3.2 System Architecture, Hardware snd Software Interfaces

A general outline of the client and server architecture is depicted in Figure 5.

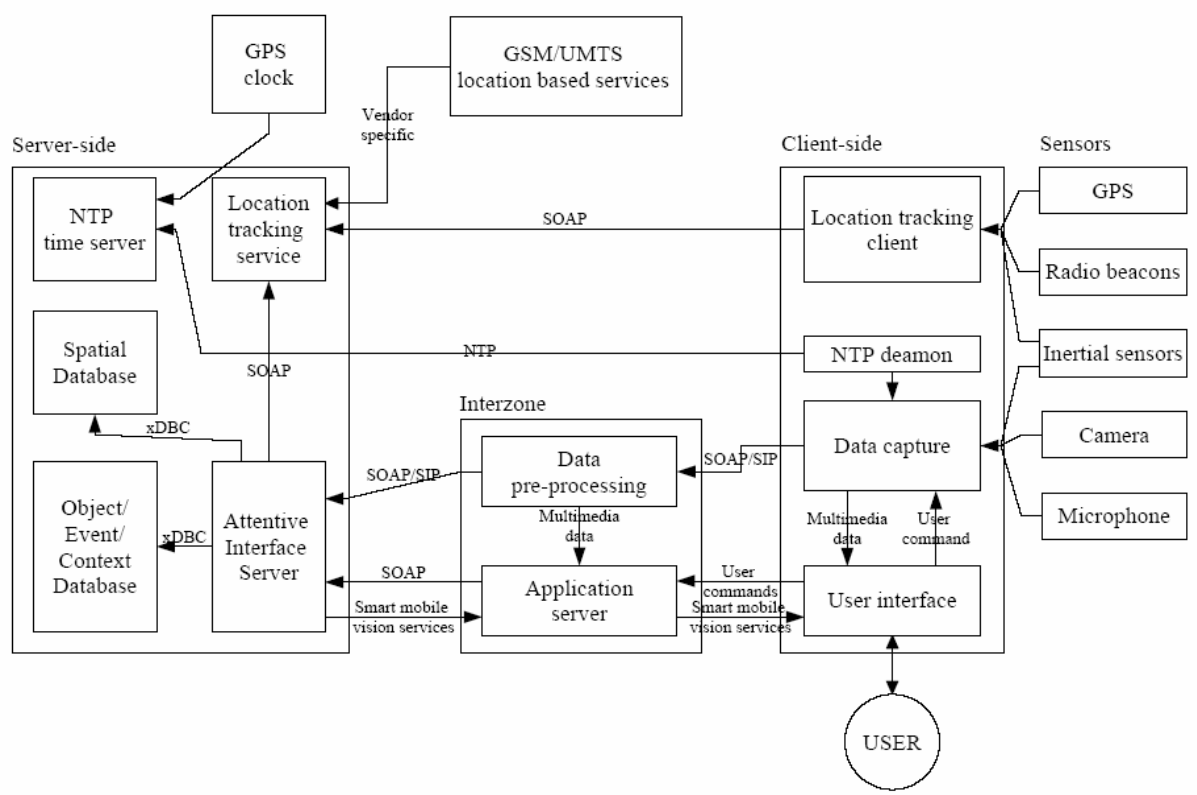


Figure 5: MOBVIS Client-server architecture

The system architecture proposed is a multi-tier client-server architecture. The system consists of three groups of modules: the server-side, the client-side and an *interzone* that specifies modules that can be easily moved from the server to the client side and vice versa.

3.2.1 COMMUNICATION BETWEEN MODULES

The communication between modules not in the same group shall be implemented by a SOAP for a general type of communication and by a session initiation protocol (SIP) for a communication that involves multimedia elements.

- SOAP Version 1.2 (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics. (M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H. Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. http://www.w3.org/TR/2003/PR-soap12-part1-20030507/)

- SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility - users can maintain a single externally visible identifier regardless of their network location. (IETF RFC 3261, “SIP: Session Initiation Protocol”)

The communication between modules of the same group can be implemented by the SOAP or the SIP or additionally by a CORBA.

- CORBA (Common Object Request Broker Architecture) provides platform-independent programming interfaces and models for portable distributed object-oriented computing applications. Its independence from programming languages, computing platforms, and networking protocols makes it highly suitable for the development of new applications and their integration into existing distributed systems.
  (M. Henning and S. Vinoski, Advanced CORBA Programming with C++, Addison Wesley 1999, page 14)

3.2.2 TIMESTAMPING DATA

An agent geo-location and collected data is annotated by a timestamp. Encapsulation of data with the geo-location is done on the server side at the time of inserting the data in the database by comparing the timestamp of data collected and the agent geo-location at this particular timestamp. For the time-stamping to work clocks on all devices must be synchronized. The system should provide a NTP (Network Time Protocol) time server and all the mobile devices should run a NTP daemon. The NTP time server should be synchronized with a GPS clock.

- NTP (Network Time Protocol) provides the mechanisms to synchronize time and coordinate time distribution in a large, diverse internet operating at rates from mundane to light-wave. It uses a returnable-time design in which a distributed subnet of time servers operating in a self-organizing, hierarchical-master-slave configuration synchronizes local clocks within the subnet and to national time standards via wire or radio. The servers can also redistribute reference time via local routing algorithms and time daemons. (IETF RFC 1305 - "Network Time Protocol (Version 3) Specification, Implementation and Analysis")

3.2.3 SERVER-SIDE

3.2.3.1 ATTENTIVE INTERFACE SERVER

The core service provided by the MOBVIS project will be an attentive interface. The attentive interface consists of three components, (1) multi-modal context awareness, (2) vision based object recognition, and (3) intelligent map technology. The attentive interface provides a way to cut down the numerous hypotheses on the real world, by first aggregating context information, then applying context to make vision based object awareness feasible, and incremental updating of map based geo-information to provide a knowledge base for future context exploitation.

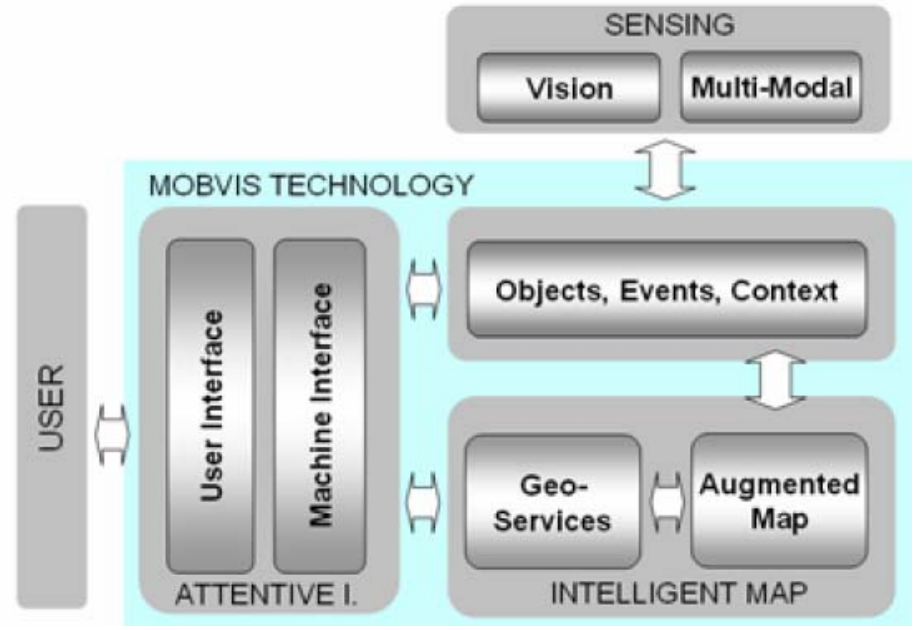


Figure 6: Concept of the interaction between the attentive interface and related modules.

The proposed system architecture implements the attentive interface as a server side process. The server process receives request through the SOAP protocol and multimedia data from the client through the SIP protocol. The attentive interface server responds to the user request by providing the user with the smart mobile vision services.

In order to provide the smart mobile vision services the attentive interface server employs a spatial database, an object/event/context database and a location tracking service. All these processes are server side based too.

3.2.3.2 SPATIAL DATABASE

The spatial database stores all geo-referenced data needed for building an intelligent map and vision based object recognition. The spatial database can be implemented by any of the spatially enabled relational database management systems (RDBMS).

3.2.3.3 OBJECT/EVENT/CONTEXT DATABASE

The object/event/context database stores:

- descriptions of objects for vision based object recognition
- events descriptions
- user context

Preferably the object/event/context database should be implemented by the same RDBMS as the spatial database. Due to specific performance requirements of the object/event/context database the relational model might not be adequate. That is why the object/event/context database is kept as a separate entity in the MOBVIS system architecture.

3.2.3.4 LOCATION TRACKING SERVICE

The location tracking service keeps track of users positions at all times as accurately as possible. It receives and stores location data sent by users' mobile devices and location data provided by wireless network operators. The location tracking service is accessible through the SOAP protocol.

3.2.3.5 NTP TIME SERVER

For the data time-stamping (see 3.4.b) clocks must be synchronized. NTP time server is synchronized with the GPS clock and provides a reference time for the mobile devices.

3.2.4 CLIENT-SIDE

The client side represents most of the functionalities that can be realized on a lightweight mobile device and involves sensors which provide input to the local sensor pre-processing modules of the attentive interface client. The client-side consists of a user interface, a data capture module, a location tracking client and has sensors attached.

3.2.4.1 USER INTERFACE

The user interface receives user commands, initiates data capture, relays user commands to an application server and presents smart mobile vision services to the user. In order to allow for interoperability on different platforms, the user interface should be separated from the data acquisition part of the client infrastructure as much as possible. The communication between the user interface and the sensors attached to the mobile device should be only through the data capture module. The user interface should be implemented using platform independent technologies (Java, Web…).

3.2.4.2 DATA CAPTURE

The data capture module acts as an interface to the sensors attached to the mobile device. The communication between the data capture module and the sensors is vendor specific. The data capture module continuously or on user command sends forth data from sensors to a data pre-processing module and to the user interface. For sending multimedia data it initiates a session with a data pre-processing module through the SIP. All the data is being time-stamped.

3.2.4.3 NTP DEAMON

In order for the data capture module to timestamp data with the accurate time, NTP daemon provides it with the reference time synchronized with the NTP time server.

3.2.4.4 LOCATION TRACKING CLIENT

Location tracking client communicates with the sensors providing location information in a vendor specific way. The location information is processed and it is sent to the location tracking service on the server-side through SOAP. The location tracking client sends location data at every change of a mobile device position and periodically at some pre-specified time interval.

3.2.5 INTERZONE

Due to the contradicting requirements regarding the mobile devices (limited processing capabilities, limited battery power, cost of wireless transmission, latency issues, start-up and response time) it is difficult for some modules to predict early on in the project whether they should be implemented on the server or the client side. For such modules we have defined a special group called the *interzone*. The modules of this group should be implemented in such a way that they can easily migrate from the server to the client side and vice versa.

3.2.5.1 DATA PRE-PROCESSING

The data pre-processing module does a local processing of the data received from the data capture module on the client-side. The processed data is sent to the attentive interface server either through the SOAP or the SIP.

3.2.5.2 APPLICATION SERVER

The application server receives user commands captured by the user interface and relays them to the attentive interface server through the SOAP. The attentive user interface responds to the user commands or events in the user environment by providing the smart mobile vision services. The smart mobile vision services are provided by the attentive interface server in the presentation independent data structures. The exact format of these data structures will be specified later on in the project. A role of the application server is to generate a representation of the smart mobile vision services provided that can be presented to the user by the user interface.

3.3 Sensor Interfaces

Within the scope of the MOBVIS project, various types of sensors will be used for collection of location data and other types of context information. In the following we will focus on the sensor-level interface for the inertial sensors, as they only offer a non-standard, proprietary interface - other than sensors such as GPS devices, for instance, which usually conform to an open standard such as NMEA 0183. At the context abstraction level (cf. Section 2.3, Multi-Modal Context) the information from inertial sensors is particularly useful, since it can be used for location estimation as well as for user activity.

3.3.1 INERTIAL SENSORS

This section introduces the MTx inertial sensors developed by XSens [XS05, XSD05] that will be used in the scope of the MOBVIS project. We will give a short overview of the sensor functionality and then describe the software interface that is used to access the sensor data.

The MTx sensor offers the following output, all of which is computed in hardware (in previous models the orientation output was computed in software):

- 3D Orientation (quaternions, Euler angles or rotation matrix)
- 3D Acceleration (+/- 10g @30Hz)
- 3D Rate of Turn (+/- 1200deg/s, @40 Hz)
- 3D Earth Magnetic Field (+/- 750 mGauss @10Hz)
- Temperature(-55 .. +125° Celsius)

Figure 7: MTx inertial sensor.

Currently there are two ways of interfacing with the MTx sensors - either using direct communication over the serial COM port, or by using the Windows COM API1. The latter can only be used on the Windows platform, but provides a higher level of abstraction and enables applications such as MATLAB, LabVIEW, Excel, etc. to interact with the sensors.

3.3.1.1 COMMUNICATION VIA THE SERIAL COM PORT

Communication with the sensors via the serial COM port is performed through a binary message based protocol. This low-level mode of communication has the advantage of being platform independent, and it offers fine-grained control over the sensors. In order to simplify development, XSens provides a C++ class called Xbus that implements the serial port communication and buffering, as well as all messages of the binary protocol. Following is a short overview over the functions that are offered by the Xbus class. More detailed information can be found in the document “MTi and MTx Low-level Communication Documentation” which ships with the sensors.

- Function for initialization and termination of communication, e.g.
  - openPort(): Opens and initializes serial port
  - openFile(): Opens new or existing file.
  - close(): Closes the handle to a port or file.
- Functions for reading and writing messages defined in the binary protocol, e.g.
  - readMessage(): Reads the next message sent by the device or stored in a file.
  - writeMessage(): Writes a message to the serial port.
  - reqSetting(): Requests a setting of the device.
  - setSetting(): Sets a setting of the device.
- Functions for retrieving data contained in messages, e.g.
  - getValue(): Retrieves sensor readings from a protocol message.
- Low-level COM-port and file functions, e.g.
  - readData(): Reads bytes from the serial port or file.
  - writeData(): Writes bytes to the serial port or file.
  - setPortQueueSize(): Sets input and output buffer size of serial port.

3.3.1.2 COMMUNICATION VIA THE WINDOWS COM API

The COM object implementation of the sensor interface is called MT Object. It is part of the XSens SDK and offers a high level interface to the sensors that encapsulates all low-level communication over the serial port. Other than the Xbus class described in the previous section, the COM object ensures backwards compatibility to older sensor models such as the MT9. Figure 8 shows the relation between the application, the MT Object and the sensor. MT Object offers (amongst others) the following inputs and outputs:

- Inputs:

[1] XSens also offers a shared object implementation for access to the sensors on the Linux platform, however, at the time of writing this only works with older models (e.g. MT9).

- COM port number
- output mode, format of orientation data
- gain, correction interval, weighting factor
- inclination

- Outputs:
  - orientation data (quaternion, Euler angles or rotation matrix)
  - calibrated sensor data (3D acceleration, rate of turn, magnetic field)
  - event notification when new orientation values are calculated (call to clients' sink object)
  - gain, correction interval and weighting factor
  - error code

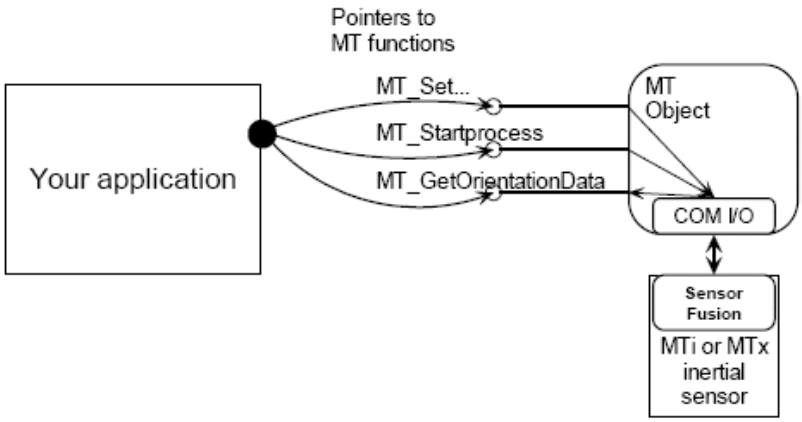


Figure 8: Communication between application and sensor via a COM object [XSD05].

Polling vs. Events

One can decide whether to use a polling mechanism or an event based mechanism to retrieve data from the MT Object. In polling mode, the user queries the MT Object whether new sensor data is available by calling MT_GetOrientationData() or MT_GetCalibratedData(). The object will then return the most recently calculated data. An internal buffer stores up to 256 samples that have not yet been fetched by polling calls, which allows the application to retrieve more than only the most recent sample. When using the event mode, the MT Object notifies the sink when new data is available for retrieval.

In polling mode the update rate of the application is decoupled from the sampling rate of the sensor, which means that the application can decide when and at what rate it wants to obtain new data. This is useful if the update rate of the application is lower than the sensor sampling frequency (usually 100 Hz). In addition to that, polling is slightly more straightforward to implement. When every sample should be retrieved at the same rate as the sampling frequency, the use of events is advised. In terms of timing, events are more reliable than polling on the Windows platform, since it is guaranteed that the retrieval of data after an event is always scheduled within 10 ms.

3.3.2 REFERENCES

[XS05] XSens Technologies, http://www.xsens.com, date of access: 19. Sep. 2005

[XSD05] XSens Technologies, "MT Software Development Kit Documentation", Revision C, 8/8/05

3.4 Georeferenced Multimedia Databases

3.4.1 POSTGRESQL

PostgreSQL is a powerful object-relational database. It is one of the most widespread and developed database and it is Open Source, the Software is for free. It is evolved from a project of the University of California at Berkley. In the meantime the development is made from commercial companies. PostgreSQL has an extensive functionality, the SQL99 standard is widely implemented and thus provides a high level of standardization. The easy way to run the database requires less special know-how. A multitude of management and administration tools help working with PostgreSQL. To store spatial geo-information, the PostgreSQL needs the spatial extension PostGIS.

3.4.2 SPATIAL EXTENSION POSTGIS

PostGIS is the spatial extension for storing and managing geodata in the freely available database server PostgreSQL. Through PostGIS PostgreSQL can be used as backend for GIS applications. This spatial extension from the company Refractions implements the SimpleFeatures specification of the OGC and allows the storage and management of vector geometries in PostgreSQL. With the implementation of the GEOS library, PostgreSQL/PostGIS offer standardized GIS functionality with operations and methods of full GIS software.
The web-based script language PHP is able to call and run these GIS methods directly in the database. Thus applications are shifted in the web, optionally for viewing of the data without storing or as output in tables.
The advantages are multi-usability and the use of complex SQL-queries for analyzing geodata. The implementation from PostGIS data is supported by the UMN Map server.

The OpenGIS specification defines two standard ways of expressing spatial objects: the Well-Known Text (WKT) form and the Well-Known Binary (WKB) form. BothWKT and WKB include information about different types of 2-dimensional objects. Examples of the text representations (WKT) of the spatial features are as follows:

- POINT(0 0)
- LINESTRING(0 0,1 1,1 2)
- POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
- MULTIPOINT(0 0,1 2)
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))
- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING((2 3,3 4)))

Additionally the spatial reference system (SRID) has to be defined to implement spatial objects in the database.

The implementation of a dataset into the table gobjekte can be seen in the following example. ’31293’ is the epsg-code for the Austrian date MGI, reference meridian M34.

INSERT INTO gobjekte (ID, NAME, LINIEN) VALUES (7, 'Linie1', GeomFromText('LINESTRING(1000000 5220000, 990000 5230000)', 31293));

3.4.3 REFERENCES

- http://www.ccgis.de/produkte.html
- http://62.153.231.87/alk/edbs2wkt/help/postgis.htm
- PostGIS Manual

3.5 Other Databases

3.5.1 MYSQL

MySQL is a fast and robust, multi-thread and multi-user SQL database Server (SQL = Structured Query Language). The application area is in high performance applications and in the implementation of widespread Software. MySQL is a relational database management system, which means the database stores data in separate tables instead of storing them in one big store. MySQL was developed for handling big databases and is therefore proper for managing big amounts of data.

3.5.2 FIREBIRD

Firebird is an Open Source relational database management system, which is based on a commercial product, the Interbase V6.

3.6 STANDARD TA MAP API: GEO ENGINE

Tele Atlas North America has provided an SDK, GeoEngine®, which can be used for application development using MapAccess data. GeoEngine is a library of APIs designed for rapid development of map applications. Applications can be developed on a variety of development platforms including C, C++, and Visual Basic.GeoEngine supports Windows 95, 98, 2000, and NT operating systems. The core product is a suite of components used by developers to configure the appearance of maps and provide user functionality such as panning and zooming. Other components include the following: (1) Geo-coding Plug-in Supports geo-coding, batch geo-coding, and reverse geo-coding (2) Spatial Query Plug-in Supports indexing, retrieval, and displaying of user data (3) Pathfinding Plug-in Supports path finding (4) Navigation Plug-in Supports in-vehicle navigation

3.6.1 INTRODUCTION TO GEOENGINE

GeoEngine is a modular development environment (SDK) developed by Tele Atlas and used for the development of mapping applications. It includes the following features:

- Platform dependencies (hardware/OS) are isolated in one component.
- Supports multiple components for reading map data (DALs).
- Provides components for general map rendering and dynamic labeling, with extensive configuration controls.
- Provides components to manage application defined overlay data.
- Provides components to support client/server communication of map data in either a binary or xml format.

- Support for graphic output, such as bitmap, gif, Windows metafile, Targa.
- Support for pkzip/gzip archives.
- Additional functionality available for the MapAccess data format, path finding and routing, Geocoding, and Navigation (positioning).

3.6.2 SYSTEM ARCHITECTURE OF GEOENGINE

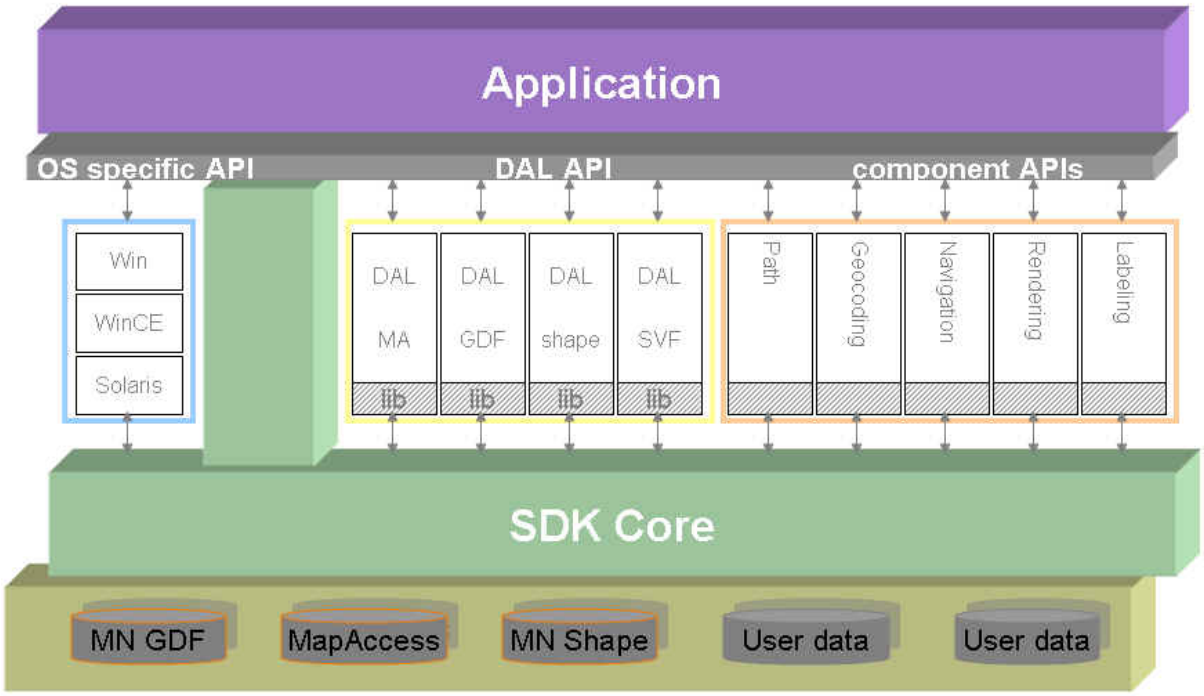


Figure 9: System architecture of GeoEngine

3.6.3 BASIC FUNCTIONALITIES

The core product is a suite of components used by developers to configure the appearance of maps and provide user functionalities

- Geocoding plug-in: supports geo-coding, batch geo-coding and reverse geo-coding
- Pathfinding plug-in: supports path finding
- Navigation plug-in: supports in vehicle navigation

3.6.4 INPUT MAP DATA FORMAT

The map database is stored using the MapAccess format of Tele Atlas. Access to it is facilitated by GeoEngine library of Tele Atlas (which is written in C). Direct access to it by invocation can be done using C, Java (using JNI) or via remote access such as sockets over http or possibly using XML or Soap over http (further experiments and analysis might change the preferred form.

A server application would probably allocate some number of handles to support incoming requests for data. If the design uses multiple threads, then a handle could be assigned to each such thread. Otherwise some serialization scheme would select the handle to use.

- MapAccess Data Format for is the format required to develop Map Display and Routing and Guidance functionalities. MapAccess data is designed as an access format, so it is compact. Data for display only for the entire US for example is about 1.4 Gb.
- Index files (IDX) are required for geocoding and reverse geocoding functionalities.

3.6.5 OUTPUT

The design of the application determines which components are used and the form of the output that can be produced. For example a server application would not need to draw or label a map if data is being transmitted for local drawing. But it would need map drawing and labeling if it is transmitting a finished gif file.
Map data can be explored, selected/searched/enumerated by locality. Currently there is no query for data by data attributes.

3.6.6 CLIENT SERVER COMPONENTS

The client/server components aid the application in the task of formatting and parsing map data, and the mechanics of making it available within the framework. It does not try to manage a communication channel. That is left to the application. Thus the application can choose any style of communication desired, and it does not place any special requirements on either the client or server environment. So a standard server environment should be adequate.

3.6.7 HARDWARE REQUIREMENTS OF APPLICATION SERVER AND MAP DATA SERVER

The map database and data server application may reside on any Windows machine. The client application may reside on any Windows or Windows-CE machine (arm/sh3/emulation). (UNIX - has not been exercised for some time and may require updates).
A client application on a Windows CE device will probably need 3-4 Mb of memory. This includes space for program code and temporary map data that is downloaded to the client to support local display user interaction.

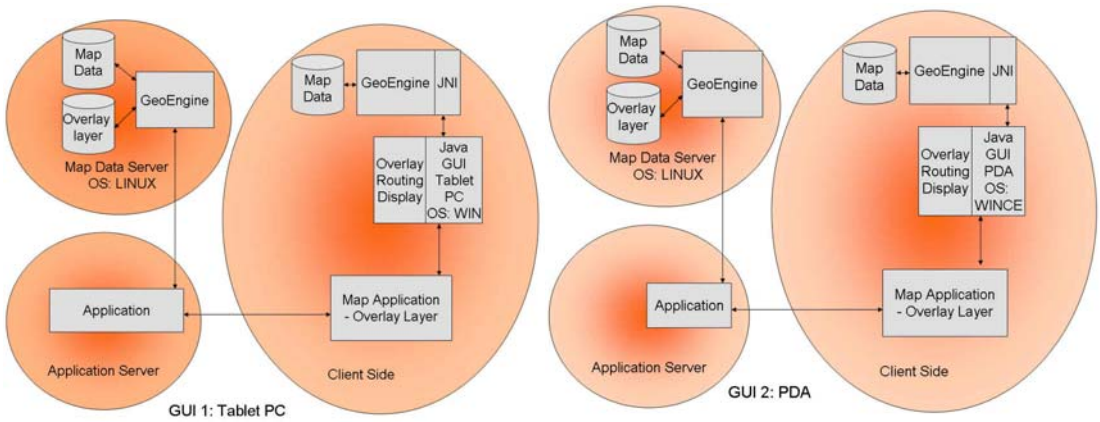


Figure 10: Example: Application server versus map data server for tablet PC and PDA.

3.6.8 REQUIRED OPERATING SYSTEM

Applications can be developed on a variety of development platforms including C, C++ and Visual Basic. GeoEngine supports Windows 95, 98, 2000, NT operating systems and Windows CE platforms.

3.6.9 PROGRAMMING LANGUAGE

The modules are written in C, and are highly portable (except OS specific module). In the Windows environment, GeoEngine is provided as either static libraries or DLLs. Applications that use GeoEngine can be written in most high level languages: C/C++, Visual Basic, Delphi, C++Builder, Java. The binary footprint is very small. Here is a list of DLL sizes in bytes from a recent build for the most commonly used modules.

emw32.dll Core	114,688
wnmw32.dll Windows platform	98,304
icmw32.dll Icon Resources	90,112
mrmw32.dll Map display	122,880
lbmw32.dll Labelling	57,344
upmw32.dll Overlay points	36,864
ulmw32.dll Overlay lines	57,344
uamw32.dll Overlay area	65,536
mamw32.dll MapAccess DAL	77,824
vsmw32.dll Server support	32,768
gcmw32.dll MapAccess Geocoding	593,920
nvmw32.dll MapAccess Navigation	172,032
pamw32.dll MapAccess Path finding	335,872

Table 2. Most commonly used DLLs in GeoEngine and corresponding sizes (in bytes).

Run time memory overhead (data) is typically about a 1 Mb or less to manage a map display. Multiple displays are easy to do, add very small additional overhead, and are designed to be threading safe in a multithreaded environment such as Windows.

3.6.10 DOCUMENTATION

The manuals serve as a general introduction to GeoEngine. For information specific to modules, plug-ins, and utilities, following documents are available:

- GeoEngine Mapping Toolkit Overview: Explains how to use the Mapping Toolkit. Includes map concepts, basic concepts, and a glossary of terms. This manual should be used in conjunction with all other functional reference documentation.
- Core Module API
- Windows Module API or UNIX Module API
- Display Module API: These functional reference books explain the foundational modules used by most applications. Together, these three modules provide basic services such as map interpretation, memory allocation, and map print/display capability.
- GeoEngine Demonstration Program: Explains how to use the demonstration program included with the Mapping Toolkit. The program illustrates much of the functionality.
- Geocoding Plug-in: Explains how to implement the Geocoding Plug-in. This plug-in geocodes-- finds the latitude and longitude coordinates for a given address – and reverse geocodes -- finds an address from given latitude and longitude coordinates. The Geocoding Plug-in enables both interactive and batch geocoding.
- Navigation Plug-in: Describes how to use the GeoEngine Navigation Plug-in with navigation sensors to enable vehicle navigation.
- Pathfinding Plug-in: Describes the Pathfinding Plug-in, which enables applications to find the best path between two points and to get directions for traveling the path.
- GeoEngine Utilities: Describes special utilities. Utilities can overlay points, lines, areas, and circles on a map, speed map display when wide lines predominate, save an image to the Windows clipboard, and save an image to a bitmap. Utilities also include a resource file containing the standard set of icons used.

3.6.11 RELATION BETWEEN GEOENGINE AND OGC STANDARDS

Using OGC standards has as obvious advantage that it is open and interoperable. On the other hand, possibly no full solution can be provided easily. The developer has to gatherall the components himself, maybe from open source or even commercial domain. In addition it is for sure that the performance of the client applications built with OGC standards are lower then when using the GeoEngine full solution, as OGC require web based interfaces. Using OGC standards means also map data to be offered and supported in GML. Right now Tele Atlas does not yet support GML as standard product.

If there are other reasons to comply with OGC, such as involving data or services from other sources than Tele Atlas, we recommend solving this issue at client side, i.e., to implement a MOBVIS client based on GeoEngine and extend it with OGC support to access the other data/services.

Detailed investigation how to use OGC as a standardised interface in MOBVIS, and how to interface with the GeoEngine software, are part of work package 5. Results will be part of the deliverables in work package 5.

3.7 OGC - Standard – Specifications

In regard to the MOBVIS system architecture standards for the transfer and the provision of geographical data specified by the OpenGeoSpatial consortium (OGC, formerly known as OpenGIS consortium) should be used. Standardised OGC interfaces will be implemented to provide geo-data to the different components of the system architecture. The MOBVIS client will also use the OGC interfaces, like a web map service, to query the geo-data for map visualisation. This chapter will give an overview of OGC standards and services, which will be used within the MOBVIS project.

3.7.1 INTRODUCTION

The OpenGeoSpatial consortium (OGC, formerly known as OpenGIS consortium) is well known to be a non profit standards organization, a union of (not only) the global players in the geospatial community. Together with ISO/TC211, the Technical Committee 211 it formulates Geomatic Standards. All information regarding their standards work can be found at
http://www.geospatial.org [June 2005]

OGC does not only publish approved standards but also serves as a communication- and discussion platform within the geospatial community. OGC's "technical baseline document" holds links to > 140 documents describing Implementation specifications, Discussion papers, Abstract specifications, recommendation papers etc.

Therefore and due the highly dynamic process of publishing standards we do not only focus on approved well known standards like for instance the Web Map Service but also on Services currently in discussion and not yet approved as standards.

Currently activity inside of Common Architecture thread of the OGC Web Service 2 Test bed. The focus of the Common Architecture tread has been to add support for WSDL/SOAP/UDDI to OGC baseline services and to test these implementations using COTS development tools. The majority of this document describes implementation issues and problems the group encountered while defining the usage of these technologies in conjunction with OGC web service interfaces. The appendices in this document describe lessons learned using WSDL and SOAP definitions with various COTS development environments. Best use of this document requires that the reader have at least a basic knowledge of WSDL/SOAP/UDDI.

3.7.2 OGC REFERENCE MODEL

The OGC Object Reference Model (ORM) describes the framework of all ongoing work; it is the technical baseline of OGC. It is a “must-read” for the understanding of OGC even if it is not a standard itself, but a document due to change.

The OpenGIS Reference Model (ORM) provides an architecture framework for the ongoing work of the OGC. Further, the ORM provides a framework for the OGC Technical Baseline. The OGC Technical Baseline consists of the currently approved OpenGIS Specifications as well as for a number of candidate specifications that are currently in progress.

The ORM has the following purposes:

- Provides a foundation for coordination and understanding (both internal and external to OGC) of ongoing OGC activities and the Technical Baseline;
- Update/Replacement of parts of the 1998 OpenGIS Guide;
- Describes the OGC requirements baseline for geospatial interoperability;
- Describes the OGC architecture framework through a series of non-overlapping viewpoints: including existing and future elements;
- Regularize the development of domain-specific interoperability architectures by providing examples.

The ORM is a living document that will be revised on a regular basis to continually and accurately reflect the ongoing work of the Consortium. Background information for this chapter can be found at http://portal.opengeospatial.org/files/?artifact_id=3836 [JUNE 2005].

Figure 11 illustrates an informative value chain for geospatial information within an enterprise or an information community. The value chain starts with geospatial information sources entering an interoperable environment, which then pass through geoprocessing chains, creating intermediate value-add geospatial-based products along the way. Figure 11 shows several examples of such value-add products including fusion (combining, correlating, annotating, and interrelating geospatial information from many sources into a single structure) and analysis (operating on geospatial information for the purpose of deriving new information, extracting results or understanding its nature and significance). The last step in the value chain involves creating finished products that contain geospatial information, or are derived from geospatial information, for both internal customers and external customers. Typical products provide functions such as visualization and portrayal, reporting, analysis or information transfer and dissemination.

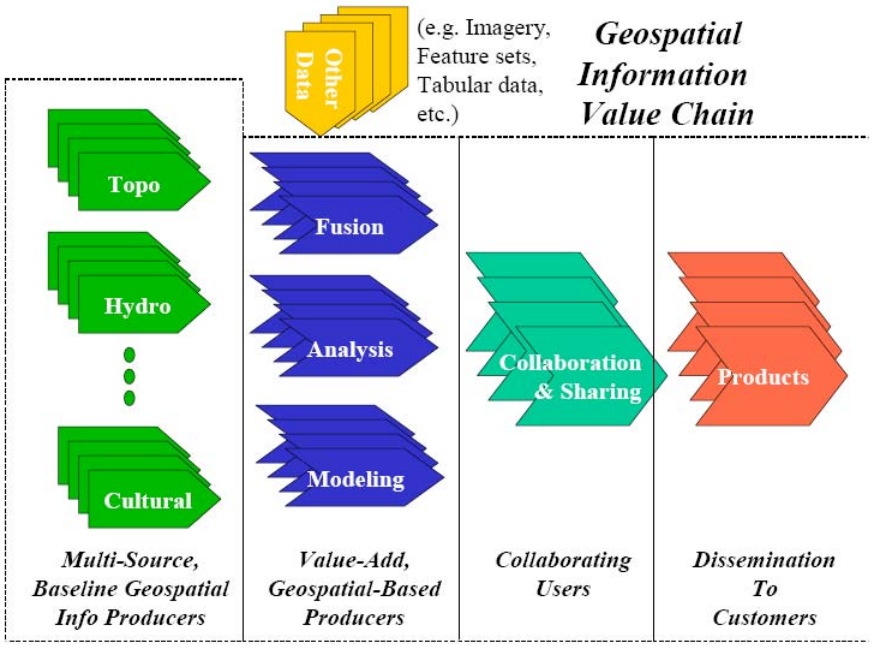


Figure 11: Representative geospatial information value chain.

The ORM adopts the so called Reference Model for Open Distributed Processing (RM-ODP), a standard for architecting open distributed systems. The RM-ODP addresses five different *viewpoints* on systems:

Viewpoint Name	Definition of RM-ODP Viewpoint
Enterprise	Focuses on the purpose, scope and policies for that system.
Information	Focuses on the semantics of information and information processing.
Computational	Captures component and interface details without regard to distribution
Engineering	Focuses on the mechanisms and functions required to support distributed interaction between objects in the system.
Technology	Focuses on the choice of technology.

Table 3. Different viewpoints on systems made by RM-ODP (source: 03-040_OpenGIS_Reference_Model_ORM_version_0.1.2.pdf).

The *Enterprise* viewpoint focussed on business perspective, and policies, the *Informational* viewpoint on semantics of information and information processing like Geographic Features, general feature models etc. The Computational viewpoint deals with a systems being treated as sets of (web) services that interact at interfaces. The *Engineering* viewpoint in turn is concerned with communication, computing systems, software processing etc. (keywords: mapping logical architectures to physical architectures, thick vs. thin clients etc). Last not least the *Technology* viewpoint deals with underlying infrastructures in distributed systems (keywords Feature Model, Coverage Model etc).

Service category	Description
Human Interaction	Services for managing user interfaces, graphics, multimedia, and presenting compound documents.
Information Management	Services for managing the development, manipulation, and storage of metadata, conceptual schemas, and datasets.
Workflow	Services that support specific tasks or work-related activities.
Processing	Services that perform large-scale computations; a processing service does not include capabilities for providing persistent storage of data or transfer of data over networks. Sub-categories of processing: - Geographic processing services – spatial - Geographic processing services – thematic - Geographic processing services – temporal - Geographic processing services – metadata
Communication	Services that encode and transfer data across networks.
System Management	Services for managing system components, applications, and networks (including access control).

Table 4. Service categories in OCG to better classify, catalogue and find relevant services.

OGC's computational viewpoint in the Object Reference Model defines the OpenGIS Service Framework and within this categorises the different services into 6 different top level categories in order to better classify, catalogue and find relevant services:

We have identified three main categories of interest for NNR, namely *Human Interaction*, *Information management* and *Processing*. We will deal with these categories in the next chapters.

3.7.3 THE OGC WEB SERVICES SERVICE FRAMEWORK OSF

The OWS Service Framework (OSF) identifies services, interfaces and exchange protocols that can be utilized by any application. OpenGIS Services are implementations of services that conform to OpenGIS Implementation Specifications. Compliant applications, called OpenGIS Applications, can then "plug into" the framework to join the operational environment.

By building applications to common interfaces, each application can be built without a-priori or run-time dependencies on other applications or services. Applications and services can be added, modified, or replaced without impacting other applications. In addition, operational workflows can be changed on-the-fly, allowing rAPId response to time-critical situations. This loosely coupled, standards-based approach to development results in very agile systems—systems that can be flexibly adapted to changing requirements and technologies

The next figure, the OWS Service Framework, shows the collaboration and dependence of the categories and services:
The OSF is designed to meet the following purposes:

- Provide a framework for coordinated development of new and extended services
- Enable interoperable services through standard interfaces and encodings
- Support publishing, discovery and binding of services through service metadata
- Allow separation of data instances from service instances
- Enable use of a provider's service on another provider's data
- Define a framework that can be implemented in multiple ways

The OSF is a profile of the OGC services taxonomy. The OSF categorizes services into five categories that correspond to the OGC services taxonomy top-level domains as shown in Table 5.

OSF Service Categories	OGC Service Taxonomy Categories
Application Services	Geographic Human Interaction
Registry Services	Geographic Information Management
Data Services	Geographic Information Management
Portrayal Services	Geographic Human Interaction
Processing Services	Geographic Processing Interaction

Table 5. OGC services taxonomy top-level domains.

3.7.4 GEOGRAPHIC SERVICES TAXONOMY

The following sub-clauses provide examples of geographic services within the geographic services taxonomy. It is not required that a system provide any service listed in these sub-clauses. It is required that if a system provides a service named in these sub-clauses that the service shall be categorized as defined in these sub-clauses. A service catalogue compliant with this International Standard shall categorize service metadata instances in the categories of the geographic service taxonomy.
If a service uses the name of an example service, the service shall provide the functionality that is defined in these sub-clauses. For example, if a service titled catalogue viewer is provided, it shall perform the services defined for the catalogue viewer in the geographic human interaction services category. Systems providing services should name services as found in the service examples.

3.7.4.1 GEOGRAPHIC HUMAN INTERACTION SERVICES

Geographic human interaction services shall be a category in the geographic service taxonomy. Examples of human interaction services for working with geographic data and services:

- Catalogue viewer. Client service that allows a user to interact with a catalogue to locate, browse, and manager metadata about geographic data or geographic services.
- Geographic viewer. Client service that allows a user to view one or more feature collections or coverage. This viewer allows a user to interact with map data, e.g., displaying, overlaying and querying. An example is the viewer client generator defined in ISO 19128.

3.7.4.2 PORTRAYAL AND HUMAN INTERFACE

Portrayal is the presentation of information to humans, e.g., a map. A map is a two-dimensional visual portrayal of geospatial data; a map is not the data itself. Two or more maps with the same geographic extent and coordinate reference system can be accurately layered to produce a composite map. Information types associated with geospatial data visualization are shown in the context of the portrayal process

1. Image or picture of the data, e.g., a map to be displayed.
2. Display elements, e.g., lexical description of graphics to be drawn onto the target display.

3.7.4.3 THIN AND THICK CLIENTS

The engineering viewpoint helps to articulate a key distinction among distributed systems:

- Thin clients rely on invoking the services of other components (servers, middleware) for most of the computation they need to function in the system; they also rely on other components to manage most of the data and metadata they need.

- Thick clients handle much of the necessary computation and data/metadata management themselves; and rather than invoking the processing services of other components, they obtain their inputs through low-level data-access requests.

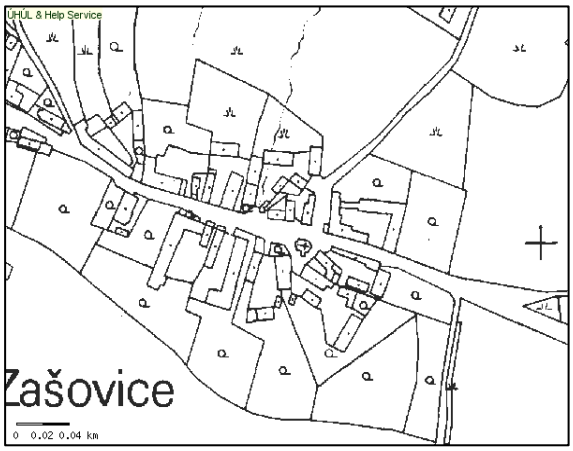
A thick client requires less functionality on the part of the server and other components; but a thin client is easier to build or to embed into general-purpose software components. The distinction often has quite tangible implications: thin clients are typically simple software with limited functions and flexibility, and smaller RAM and CPU requirements, often suitable for handheld or mobile devices. Thick clients usually require a significant portion of (at least) a microcomputer's resources, but provide greater flexibility and capacity to decode, transform, render, and interact with retrieved data.

3.7.4.4 RELEVANT STANDARDS

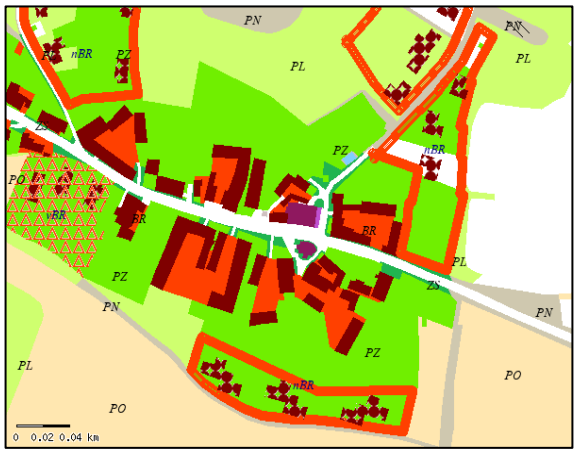
3.7.4.4.1 WMS/SLD implementation

A Web Map Service (WMS) produces maps of geo-referenced data. It defines a "map" as a visual representation of geodata; a map is not the data itself. This specification defines three WMS operations: GetCapabilities returns service-level metadata, which is description of the service's information content and acceptable request parameters; When requesting a map, a client may specify the information to be shown on the map (one or more "Layers"), possibly the "Styles" of those Layers, what portion of the Earth is to be mapped (a "Bounding Box"), the projected or geographic coordinate reference system to be used (the "Spatial Reference System," or SRS), the desired output format, the output size (Width and Height), and background transparency and colour.
When two or more maps are produced with the same Bounding Box, Spatial Reference System, and output size, the results can be accurately layered to produce a composite map. The use of image formats that support transparent backgrounds allows the lower layers to be visible. Furthermore, individual map Layers can be requested from differentServers. The WMS specification thus enables the creation of a network of distributed Map Servers from which Clients can build customized maps.
A particular WMS provider in a distributed WMS network need only be the steward of its own data collection. This stands in contrast to vertically-integrated web mapping sites that gather in one place all of the data to be made accessible by their own private interface.

1. server - cadastral map



2. server - thematic map



WMS result

Figure 12: Example for WMS result.

This specification standardizes the way in which maps are requested by clients and the way that servers describe their data holdings. The document defines three operations, the first two of which are required of every WMS.

- **GetCapabilities** (required): Obtain service-level metadata, which is a machine-readable (and human-readable) description of the WMS's information content and acceptable request parameters.
- **GetMap** (required): Obtain a map image whose geospatial and dimensional parameters are well-defined.
- **GetFeatureInfo** (optional): Ask for information about particular features shown on a map.

A standard web browser can ask a Web Map Service to perform these operations simply by submitting requests in the form of Uniform Resource Locators (URLs) [IETF RFC2396]. The content of such URLs depends on which of the tasks is requested. All URLs include a specification version number and a request type parameter. In addition, when invoking GetMap a WMS Client can specify the information to be shown on the map (one or more "Layers"), possibly the "Styles" of those Layers, what portion of the Earth is to be mapped (a "Bounding Box"), the projected or geographic coordinate reference system to be used (the "Spatial Reference System," or SRS), the desired output format, the output size (Width and Height), and background transparency and colour. When invoking GetFeatureInfo the Client indicates what map is being queried and which location on the map is of interest. When two or more maps are produced with the same Bounding Box, Spatial Reference System, and output size, the results can be accurately layered to produce a composite map. The use of

image formats that support transparent backgrounds (e.g., GIF or PNG) allows the lower Layers to be visible. Furthermore, individual map Layers can be requested from different Servers. The WMS GetMap operation thus enables the creation of a network of distributed Map Servers from which Clients can build customized maps. A particular WMS provider in a distributed WMS network need only be the steward of its own data collection. This stands in contrast to vertically-integrated web mapping sites that gather in one place all of the data to be made accessible by their own private interface.
Because each WMS is independent, a WMS must be able to provide a machine-readable description of its capabilities. This "Service Metadata" enables Clients to formulate valid requests and enables the construction of searchable catalogues that can direct clients to particular WMSes. A WMS may optionally allow the GetFeatureInfo operation. If it does, its maps are said to be "queryable," and a Client can request information about features on a map by adding to the map URL additional parameters specifying a location (as an X, Y offset from the upper left corner) and the number of nearby features about which to return information.

Cascading Map Servers

A "Cascading Map Server" is a WMS that behaves like a client of other WMSes and behaves like a WMS to other clients. For example, a Cascading Map Server can aggregate the contents of several distinct map servers into one service. Furthermore, a Cascading Map Server can perform additional functions such as output format conversion or coordinate transformation on behalf of other servers.

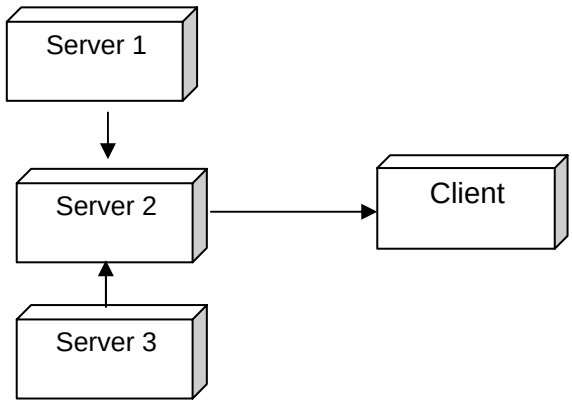


Figure 13: Cascading Map Servers.

3.7.4.4.2 Styled Layer Descriptors

The current OpenGIS Web Map Service (WMS) specification supports the ability for an information provider to specify very basic styling options by advertising a preset collection of visual portrayals for each available data set. However, while a WMS currently can provide the user with a choice of style options, the WMS can only tell the user the name of each style. It cannot tell the user what portrayal will look like on the map. More importantly, the user has no way of defining new styling rules. The ability for a human or machine client to define these rules requires a styling language that the client and server can both understand. Defining this language, called the StyledLayerDescriptor (SLD), is the main focus of this paper, and it can be used to portray the output of Web Map Servers, Web Feature Servers and Web Coverage Servers. In many cases, however, the client needs some information about the data residing on the remote server before he, she or it can make a sensible request. This led to the definition of new operations for the OGC services [see Section 6.6] in addition to the definition of the styling language.
There are two basic ways to style a data set. The simplest one is to colour all features the same way. For example, one can imagine a layer advertised by a WMS as “hydrography” consisting of lines (rivers and streams) and polygons (lakes, ponds, oceans, etc.). A user might want to tell the server to colour the insides of all polygons in a light blue, and colour the

boundaries of all polygons and all lines in a darker blue. This type of styling requires no knowledge of the attributes or “feature types” of the underlying data, only a language with which to describe these styles. This requirement is addressed by the **FeatureTypeStyle** element in the SLD document.
A more complicated requirement is to style features of the data differently depending on some attribute. For example, in a roads data set, style highways with a three-pixel red line; style four-lane roads in a two-pixel black line; and style two-lane roads in a one-pixel black line. Accomplishing this requires the user to be able to find out what attribute of the data set represents the road type. WMS already has an optional operation that fulfils this need, called **DescribeLayer**. This operation returns the feature types of the layer or layers specified in the request, and the attributes can be discovered with the **DescribeFeatureType** operation of a WFS interface.

3.7.4.4.3 Filter encoding Specification

Open Geospatial Consortium introduces XML based querying language for web services - Filter Encoding. It may be used not only for queries on vector data. Both attribute and spatial based queries are supported.

There is short list of operators:

- Spatial: Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains, Within, Beyond
- Comparison: PropertyIsEqualTo, PropertyIsNotEqualTo, PropertyIsLessThan, PropertyIsGreaterThan, PropertyIsLessThanOrEqualTo, PropertyIsGreaterThanOrEqualTo, PropertyIsLike, PropertyIsNull, PropertyIsBetween
- Logical: And, Or, Not
- Mathematical: And, Or, Sub, Div.

3.7.5 GEOGRAPHIC MODEL/INFORMATION MANAGEMENT SERVICES

Examples of model/information management services for working with geographic data and services:

- Feature access service. Service that provides a client access to and management of a feature store. An access service may include a query that filters the data returned to the client. ISO 19125-1, ISO 19107 and ISO 19111 are relevant to feature access.
- Map access service. Service that provides a client access to a geographic graphics, i.e., pictures of geographic data. ISO 19128 is relevant to map access. *(WMS see previous chapter}*
- Coverage access service. Service that provides a client access to and management of a coverage store. An access service may include a query that filters the data returned to the client. ISO 19123 and ISO 19111 are relevant to coverage access.
- Product access service. Service that provides access to and management of a geographic product store. A product can be a predefined feature collection and metadata with known boundaries and content, corresponding to a paper map or report. A product can alternately be a previously defined set of coverages with associated metadata.
- Feature type service. Service that provides a client to access to and management of a store of feature type definitions. The static and dynamic information models for a feature type catalogue are provided in ISO 19110.
- Catalogue service. Service that provides discovery and management services on a store of metadata about instances. The metadata may be for dataset instances, e.g., dataset catalogue, or may contain service metadata, e.g., service catalogue. ISO 19115 is relevant to catalogue service for dataset metadata.
- Registry Service. Service that provides access to store of metadata about types. Types are vocabularies that can be organized and related to each other. Example registries are information community registries, type dictionaries, service registries and schema registries

- Gazetteer service. Service that provides access to a directory of instances of a class or classes of real-world phenomena containing some information regarding position. An information model for a gazetteer is provided by ISO 19112.

3.7.5.1 OGC SPECIFICATION - WEB FEATURE SERVICE (WFS)

WFS is oriented for vector data exchange on internet. It uses GML format for data transfer. Both geometric and attribute information are transferred with GML. Also OGC Web gazetteer Service profile is derived from this service. An extended capability of the service enables transactional operations like insert/update/delete features on remote server. WFS uses also Filter Encoding for querying.

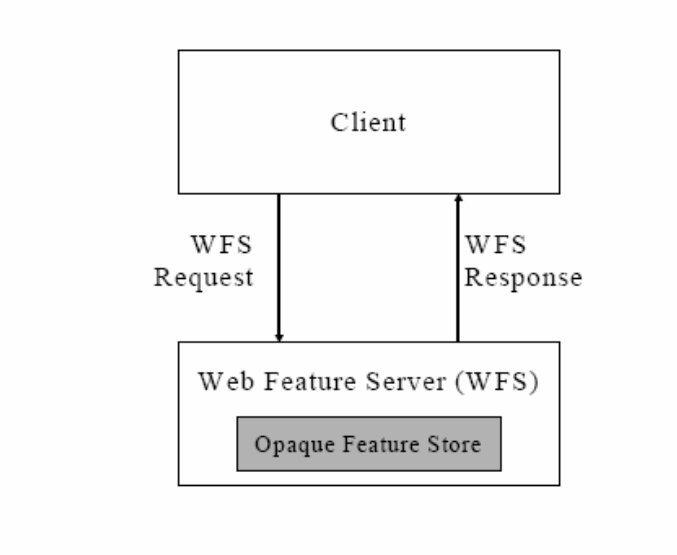


Figure 14: Web feature Server (WFS).

Operations

To support transaction and query processing, the following operations are defined:

GetCapabilities

A web feature service must be able to describe its capabilities. Specifically, it must indicate which feature types it can service and what operations are supported on each feature type.

DescribeFeatureType

A web feature service must be able, upon request, to describe the structure of any feature type it can service.

GetFeature

A web feature service must be able to service a request to retrieve feature instances. In addition, the client should be able to specify which feature properties to fetch and should be able to constrain the query spatially and non-spatially.

GetGmlObject

A web feature service may be able to service a request to retrieve element instances by traversing XLinks that refer to their XML IDs. In addition, the client should be able to specify whether nested XLinks embedded in returned element data should also be retrieved.

Transaction

A web feature service may be able to service transaction requests. A transaction request is composed of operations that modify features; that is create, update, and delete operations on geographic features.

LockFeature

A web feature service may be able to process a lock request on one or more instances of a feature type for the duration of a transaction. This ensures that serialisable transactions are supported.

Based on the operation descriptions above, three classes of web feature services can be defined:

Basic WFS

Basic WFS would implement the GetCapabilities, DescribeFeatureType and GetFeature operations. This would be considered a READ-ONLY web feature service.

XLink WFS

An XLink WFS would support all the operations of a basic web feature service and in addition it would implement the GetGmlObject operation for local and/or remote XLinks, and offer the option for the GetGmlObject operation to be performed during GetFeature operations.

Transaction WFS

A transaction web feature service would support all the operations of a basic web feature service and in addition it would implement the Transaction operation. Optionally, a transaction WFS could implement the GetGmlObject and/or LockFeature operations.

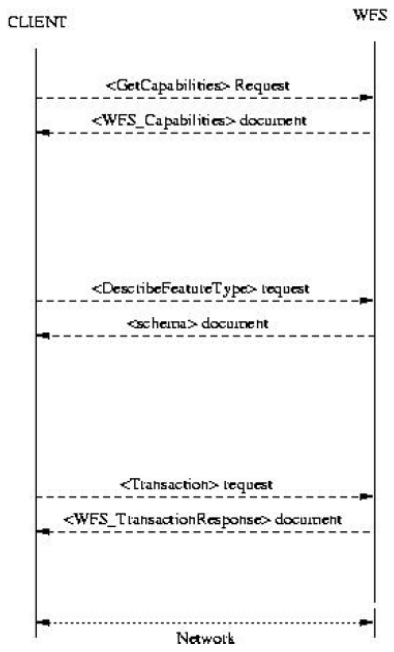


Figure 15: Protocol diagram.

3.8 Attentive Interface

The Attentive Interface is a central component in MOBVIS for the exploitation of context for mobile vision purposes (Figure 6). This interface is important in several respects. Firstly, its user sided part, i.e., the *Attentive User Interface (AUI)*, maintains track of user's task and behaviour via controlled interaction with the user. This involves a kind of decision making in notifying the user about the context state, informs about system driven annotation, and aims at requesting user input at crucial stages of a mobile vision task, such as, navigating, positioning, or vision based annotating. Secondly, we are concerned with the *Attentive Machine Interface (AMI)*, a fully autonomous functionality with the general aim to perform reasoning and decision making about the current context state and about goal driven information acquisition. The core reasoning engine of the AMI will execute a hypothesise-and-verify schema in order to refine hypotheses about the context state. E.g., in order to perform object recognition, it would propose several object hypotheses in the context state which then would be verified by location based signals that restrict possible objects from the geo-context. Finally, the AMI would operate using attention in the sense of a selective fusion operation, i.e., by reinforcing attentive, context-driven features to extract regions of interest for further processing, e.g., in object search. The AMI is bringing in the system intelligence that will provide expectation schemes that would take actual advantage from any context type in order to prime further image analysis, making in this sense smart mobile vision become reality.

A control module will receive final estimates on the individual context state (e.g., position, vision based context, activity, user profile, etc.) and provide decision making to perform targeted 'perceptual' actions in terms of visual routine execution, with the goal to extract the relevant information regarding user and context. This task performs highly innovative research on attentive mechanisms which we understand in terms of operations on information selection (feature space), object hypothesis rejection or verification (object search), image segmentation (regions of interest), and module selection (filtering, cascading, multi-cueing). In addition, use case related context will be extracted from simple user profiles (activity, task, environment) to prime the mobile vision task. From this emerging technology, we expect results of context based place and object recognition, e.g., by focusing interpretation on specific regions in the image.

A typical example is search for pedestrians in the local neighbourhood of a detected sidewalk, instead of searching for them on facades of buildings. The strategy can involve a sequence of operations, e.g., first, localisation of the road, second, localisation of sidewalk, third, searching for head-shoulder-patterns, fourth, verification by human body shape recognition, and so forth. The methodology on sequential attentive selection of information will be evaluated towards computational models of decision trees, Bayesian networks, (Hierarchical Partial Observable) Markov Decision Processes (HPOMDP/MDP), reinforcement learning schemes, and time series prediction.

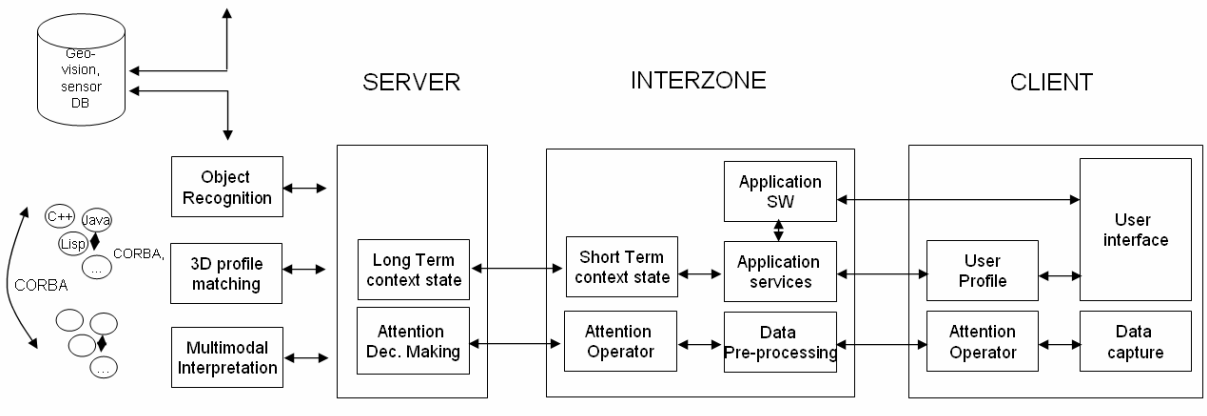


Figure 16: System architecture of components relevant for the Attentive Interface functionality.

Figure 16 shows components of the system architecture (related to Figure 5) that are relevant for the operation of the Attentive Interface. On the *client* side, user specific information will be collected to add to on-line user profiling which is then forwarded and processed within the attention related application specific mobile services. Data capture is pre-processed in the *interzone* module, while attention operators tuned by the Attentive Decision Maker (*server* side) will be capable of selecting, parametrising and masking on-going information processing. The *interzone* will finally be either client or server side. However, there we will integrate a kind of short term context memory that will include all relevant information to determine a context state. This state can feed back to attention operators to tune specific interpretation and information selection. On the server side, the decision maker will forward information to multimodal interpretation, 3D information recovery and object recognition modules. These components can interact via CORBA interfaces. On the other hand, the information resulting from these components will be processed in the attentive decision maker to feed back attention parameterisation for the client and *interzone* information processing.

The attentive interface should operate in terms of an intelligent middleware, receives continuous sensor data, is aware of each users context state, throws events (push service), processes client requests (client triggered), controls server modules (vision, geo-, sensor-processing, …) and sends responses to the client. Communication with the client will most probably be based on SOAP. The Interface will be specified in detail within Deliverable D6.1.1 which is due month 18.