Ph.D. Thesis Proposal: Supporting Sensor Fusion for Context-aware Computing

Huadong Wu, (whd@cs.cmu.edu)
The Robotics Institute, Carnegie Mellon University

ABSTRACT

Humans can convey very complicated ideas easily because they have an implicit understanding of their environmental situation, or "context". "Context-aware computing" research aims at intelligently connecting computers, users, and the environment; its sensing technology research — research of intelligently integrating multiple sensors and multiple sensor modalities, or sensor fusion — however is not entirely up to the challenge yet.

A user's situational context may include various levels of relevant information about his/her activities and intention as well as the current environment (with higher-level contexts being derived from multiple lower-level contexts and sensors' outputs), and a context-aware computing system can possibly sense such a wide range of information only via a network of sensors working in concert. However, different sensors have different resolutions and accuracies as well as data rates and formats, and the sensed information may have overlaps or even conflicts. To sense "context" information hence means that some sensor fusion technologies are indispensable.

Context-aware computing systems developed thus far usually have their system architecture heavily depending on the sensors they are using, their sensors and the sensed context information are highly coupled, and they typically used very limited context information such as user identity, location, and time. The "Context Toolkit" system developed in the Georgia Institute of Technology promotes separating context from the sensors that sense the context information through using sensor widgets, modularizing and standardizing software interfaces, thus makes replacing existing sensors and/or adding new sensors to incorporate more and more-complex context information easier.

However, the Context Toolkit system lacks the capability or mechanism to address the uncertainty and ambiguity problems essentially co-existing with sensing and sensors, thus it cannot properly handle information-overlapping issues and resolve conflicts from multiple information sources. This proposed PhD thesis research challenges to provide a generalize-able sensor fusion architectural support for context-aware computing through addressing the intrinsic uncertainty and ambiguity problems in context sensing.

Based on an assumption that context information can be decomposed into discrete facts and events according to a simplified context model, sensor widgets (inherited from a corresponding Context Toolkit system) are modified to generate context observations of hypotheses according to the context information model. For each kind of the predefined context observations or hypotheses, there is a sensor fusion mediator in the system to coordinate all the sensors and to statistically combine information from multiple sources. The proposed methodology is to standardize the representation of context information

and their communication interfaces, meanwhile leave the implementation of generating context observations from sensors up to the sensors' widgets, which are supposed to only report the results of their local sensor fusion processing in temporal dimension. The proposed system will be implemented with dynamic context databases to further facilitate sensor fusion at different levels.

The proposed sensor fusion architecture backed with dynamic context database is expected to have performance improvements over a Context Toolkit system as:

1) Providing uncertainty and ambiguity information about the sensed context, so that the user applications have an idea about how these information should be trusted, thus providing references about how these information should be used.
2) The sensed context information is automatically classified and consolidated, whereas conflicts are expected to be already resolved. In other words, the sensed context is further separated from the sensing implementations, so that the user applications can concentrate on better using contexts instead of where and how the contexts are got;
3) The layered structure of "sensors with widgets", "context observations with a sensor fusion mediator", and the final "context information" makes it easier to apply artificial intelligent algorithms to extract more context (and/or more-complex context) information and to dynamically choose appropriate senor fusion schemes according to the environment change.
4) The system's layered structure plus the uncertainty representation provide a good tradeoff in balancing insulating abstract context to simplify context usage and knowing context-sensing details to improve context-sensing processes. It makes replacing existing sensors and adding new sensors easier, and it provides a way for some sensors to gracefully die out in a sensors' dynamic configuration.
5) The way that the system aggregates contexts, implemented with dynamic context databases as well as using a context server, makes it easier to support "situation description" function — i.e., a user-application specifies a complex context situation to the system and the system takes care of verifying when all the conditions are met and notifies the application.

The theme of this proposed thesis research is demonstrate the hypothesis that synergistic interactions between sensor fusion and context information can greatly facilitate the sensor fusion process, provided that we have an appropriate context representation and a good software architecture support. Two application scenarios will be implemented to test the feasibility of the idea.

The primary concept-demonstration system will be a user identification application in an instrumented office room, where the context information will be mainly about whether there are registered users in the entrance part and inside part of the room, and the basic method to statistically combine context information will be firstly implemented with Dempster-Shafer theory of evidence. The developed software tool package will be used in the Motorola Research Lab in Schaumburg, Illinois.

The secondary concept-demonstration system will be to use context information to tune up sensor fusion algorithms in object tracking around a moving vehicle (posts, trees, pedestrians, other vehicles, etc.). The context information is mainly the vehicle's velocity and location, which can provide constraints about how fast the being tracked objects can move. The research result is to be used by the Robert Bosch Research Center, North America (Pittsburgh, Pennsylvania).

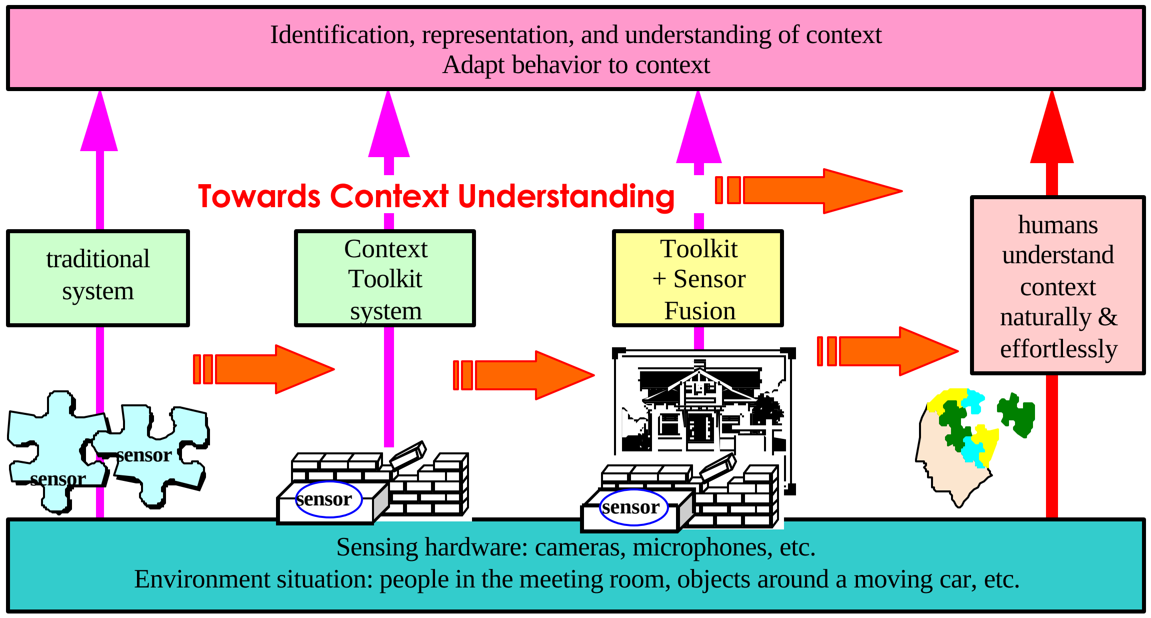

Identification, representation, and understanding of context
Adapt behavior to context
Towards Context Understanding
traditional system
Context Toolkit system
Toolkit + Sensor Fusion
humans understand context naturally & effortlessly
sensor
sensor
sensor
sensor
Sensing hardware: cameras, microphones, etc.
Environment situation: people in the meeting room, objects around a moving car, etc.

Figure 1. Towards context understanding: Context Toolkit promotes separation of context usage and context-sensing; the proposed system provides sensor fusion support

PH.D. THESIS PROPOSAL: SUPPORTING SENSOR FUSION FOR CONTEXT-AWARE COMPUTING 1

ABSTRACT........ 1

1. BACKGROUND AND MOTIVATION 6

1.1. CONTEXT-AWARE COMPUTING BACKGROUND 6

1.1.1. What "context information" includes *6*

1.1.2. Trends in context-aware computing *7*

1.2. SUPPORTING CONTEXT-AWARE COMPUTING 8

1.2.1. Georgia Tech Context Toolkit *9*

1.2.2. Missing part: sensor fusion *10*

1.3. CONTEXT VERSUS SENSOR FUSION 11

1.3.1. Sensor fusion frustration and opportunity *11*

1.3.2. Interactions can help *12*

1.3.3. Layered and modularized architecture desired *13*

2. CONTEXTS AND CONTEXT-SENSING 14

2.1. CONTEXT CONTENTS AND REPRESENTATION 14

2.1.1. Context classification *14*

2.1.2. Context representation *15*

2.2. CONTEXT UNCERTAINTY MANAGEMENT 17

2.2.1. Sensor fusion and its methods *17*

2.2.2. Dempster-Shafer theory of evidence method *18*

2.2.3. Other alternative methods *19*

Classical inference and Bayesian inference 19

Voting fusion 20

Fuzzy logic method 21

Neural network method 21

2.3. CONTEXT AND SYSTEM ARCHITECTURE 22

2.3.1. The role of context in system *22*

2.3.2. IEEE 1451 "smart sensor" standard concerns *23*

3. SUPPORTING SENSOR FUSION 25

3.1. SENSOR FUSION METHODOLOGY 25

3.1.1. Fuse along two dimensions *25*

3.1.2. Sensor fusion procedures *26*

3.2. SENSOR FUSION REALIZATION 27

3.2.1. Information layer and function modularization *27*

3.2.2. Information propagating and updating *29*

3.3. A USER-IDENTIFICATION IMPLEMENTATION 30

3.3.1. Context in the system *30*
3.3.2. System configuration.. *30*
3.3.3. Demonstration, software package, and documentation.. *32*

3.4. AN OBJECT-TRACKING IMPLEMENTATION 32
3.4.1. Application and system description *33*
3.4.2. Using context to tune up sensor fusion processes.. *34*

4. WORKING PLAN AND EXPECTATIONS.. 36

4.1. SCHEDULE AND EXPECTATION.. 36

4.2. CONCEPT-DEMONSTRATION EXPECTATIONS 37

4.3. EXPECTED CONTRIBUTION, FUTURE WORK 38

5. REFERENCES 39

1. Background and Motivation

1.1. Context-aware Computing Background

Humans can convey very complex ideas to each other and react appropriately according to circumstances because they implicitly use the "situational" or "context" information. The idea of "context-aware computing" is to have computers understand our real world so that the human-computer interactions can happen at much higher abstraction levels like human-to-human interactions, hence to make "ubiquitous computing" realistic and valuable.

Thus, it can be seen that the better term to describe the idea might be "context-aware HCI (human-computer-interaction)". However, since the "context-aware computing" terminology is well established, we will stick with it in this document.

1.1.1. What "context information" includes

To give some examples of context-aware computing, let us imagine some application scenarios where the ubiquitous computing system can greatly enhance the quality of some services to us or to our personal productivity.

- Example 1: Suppose you are new to a place (a city, a mall, a tradeshow etc.) and would like to have your computer collect the "relevant" information and give you a tour. A good "context-aware computing technology"-enabled system should somehow be able to know your available time and your interests and preferences, tentatively plan a tour for you, get your feedback and guide you point-to-point through the visiting. During the tour, the system should be able to sense and guide your focus of attention, adjust the content description details and delivering rhythm, and adaptively include or drop out some contents – just as a human tour guide usually does it naturally.
- Example 2: A cellular phone with pager function can now connect your phone calls and deliver your email in your handset. A "context-aware computing"-enabled personal information management system should further know what you are doing and adjust its behavior accordingly. For example, you may want it to deliver only time-sensitive e-mails to your cellular phone set, use text-voice function to read them when you are driving a car etc.; when you are in a meeting, you would like it to route the unimportant incoming calls to your voice mail box, to vibrate your cellular phone handset instead of ringing to announce an important call etc.
- Example 3: A "context-aware computing"-enabled home service system should detect the activities of its occupants: the room temperature should be adjusted not just based on the time of the day but based on the current occupants' preferences and their actual activities. Some additional potential functionalities are: it can tell whether your children are doing good (they are not crying, they are not fighting against each other etc.) in the other rooms, it can notice and remind you that when cooking you are distracted by other events and forget you that you have some stuff still being cooked on the stove, and it may be able to help you to remember where you might have misplaced your keys yesterday etc.

To make the above examples-illustrated intelligent human-computer-interactions possible, the first step is to have computers somehow understand some basic facts that are so obvious to our human beings, such as being outside in winter means very different from being outside in summer.

The term "context-aware" was first introduced by Schilit et al [2][3] to address the ubiquitous computing mode where a mobile user's applications can discover and react to changes in the environment they are situated. The firstly emphasized three important aspects of context were: where you are, whom you are with, and what resources are nearby.

Almost any available information at the time of interaction can be regarded as "context". The following list is just an example of some commonly considered information contents based on Korkea-aho's [5] context enumeration:

- Identity of the user
- Spatial information: location, orientation, speed, acceleration, etc.
- Temporal information: time of the day, date, season of the year, etc.
- Environment information: temperature, air quality, light, noise level, etc.
- Social situation: who you are with, people who are nearby, etc.
- Resources that are nearby: accessible devices, hosts, etc.
- Availability of resources: battery, display, network, communication bandwidth, etc.
- Physiological measurements: blood pressure, heart rate, respiration rate, muscle activities, tone of voice
- User's physical activity: talking, reading, walking, running, etc.
- User's emotional status: preferences, mood, focus of attention, etc.
- Schedules and agendas, conventional rules, policies etc.

In the attached appendix "Towards Context Modeling: Context Classification", an attempt was made to give a complete list of commonly conceived context and to classify them into categories.

1.1.2. Trends in context-aware computing

After surveying the existing work in context-aware computing, Dey and Abowd [1] gave a formal definition of context as

> "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or objects that is considered relevant to the interaction between a user and application, including the user and applications themselves."

With this definition, then the "context-aware computing" is defined as:

> "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task[1]".

[1] The term "task" here means the applications that are running in the system to provide service functions to the user.

Their proposed context-aware computing classification includes three categories: (1) presentation of information and services to a user; (2) automatic execution of a services; and (3) tagging of context to information for later retrieval.

Though context-aware computing can be an extremely broad concept, the research and development is currently at its infant stages: typically, only one or very few pieces of context information is sensed and used.

The most successfully used context information is perhaps the user's location information. Among those early successful location-aware computing projects, often referenced are the Olivetti Research Ltd. (now AT&T Labs, Cambridge, U.K.)'s "Active Badge" (1998-1992)[6], the Xerox's PARCTAB (1992-1993)[7], the Georgia Institute of Technology's CyberGuide (1995-1996)[8], and the University of Kent at Canterbury's "Fieldwork" or "Stick-e" (1996-1998)[9][10].

For those early and even later-on successfully developed "Active Map" or "Tour Guide" application scenarios (such as University of Karlsruhe et al's TEA I and TEA II, Georgia Institute of Technology's CyberGuide, MIT Media Lab's comMotion, and University of Lancaster's GUIDE etc.), basically the user's current location is the only used context information: the vicinity map is updated or the nearby point-of-interests is displayed blindly — without knowing the user's actual focus of attention, preference, intention or interest at that time.

Recent context-aware computing research development trend is to integrate more context information. Some examples of the more complicated and advanced context-aware computing researches are: Microsoft's "EasyLiving", Georgia Institute of Technology's "Aware Home", and Carnegie Mellon's "Aura".

The "EasyLiving" project aims at developing a prototypal architecture and the necessary technologies for intelligent office environments, where a group of dynamically collected smart devices can work together to provide users access to information and services. Currently the EasyLiving system can handle a single room and 10's of devices with dynamic changes to their configuration; one to three people can simultaneously use the facility. [11][12]

Georgia Institute of Technology's "Aware Home" creates a living laboratory for research in ubiquitous computing for everyday activities in home life of a family or elderly occupants. The house construction was report to complete by the end of 1999, and the room environment instrumentation as well as some experiments and case study is in progress. [13]

Carnegie Mellon's "Aura Consortium" consists of a series of ubiquitous computing research projects in Human-Computer-Interaction, wearable computers, intelligent networking, software composition etc. The research goal is to provide each user with an invisible halo of computing and information services that persists regardless of location so that the users can interact with their computing environments in terms of high-level tasks. (http://www.cs.cmu.edu/~aura/)

1.2. Supporting Context-aware Computing

Although the context-aware computing research development trend is to integrate more contexts and more complex context information, generally speaking, the progress is far less than desired. Among others, one important reason is perhaps the lack of system modularization standards and context-sensing support. The Context Toolkit system (http://www.cc.gatech.edu/fce/contexttoolkit/) developed in the Georgia Institute of Technology GVU (Graphics, Visualization & Usability) Center attempts to provide such system-modularizing support.

1.2.1. Georgia Tech Context Toolkit

Over years of evolution, with regard to computer inputs, especially with regard to the usage of keyboard and mouse devices, the input mechanism becomes very well defined over various applications — in the perspective of availability and common functional assignments. It is now logical to have the applications use standard GUI instead of having to handle the low-level input device drivers' tasks directly. Based on this observation, Dey et al. in Georgia Institute of Technology's GVU (Graphics, Visualization & Usability) Center developed the so-called Context Toolkit system attempting to facilitate context applications in analogy to extract user input contents from computer standard input devices' implementation.

As illustrated in Figure 2, the Context Toolkit consists of three basic building blocks: context widgets, context aggregators and context interpreters[2].

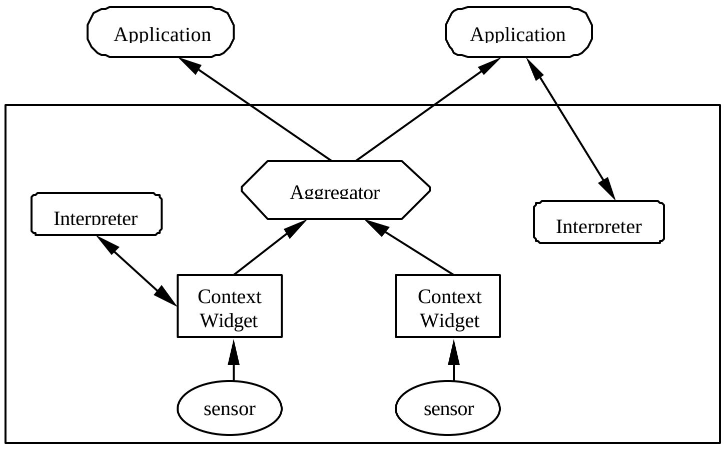


Figure 2. Georgia Tech Context Toolkit component and context architecture

The figure also shows the relationship between sample context components: arrows indicate data flow. The context components are intended to be persistent: instantiated independently of each other in separate threads and on separate computing devices, they are executed all the time.

Context widgets encapsulate information about a single piece of context, such as location or activity, for example. They provide a uniform interface to components or applications that use the context, hiding the details of the underlying context-sensing mechanism(s), allowing them to treat implicit and explicit input in the same manner. Context widgets allow the use of heterogeneous sensors that sense redundant input, regardless of whether that input is implicit or explicit. Widgets maintain a persistent record of all the context they sense. They allow other components to both poll and subscribe to the context information

[2] The figure and the following system description are from reference [16].

they maintain. Widgets are responsible for collecting information about the environment.

A context interpreter is used to abstract or interpret context. For example, a context widget may provide location context in the form of latitude and longitude, but an application may require the location in the form of a street name. A context interpreter may be used to provide this abstraction. A more complex interpreter may take context from many widgets in a conference room to infer that a meeting is taking place.

A context aggregator is very similar to a widget, in that it supports the same set of features as a widget. The difference is that an aggregator collects multiple pieces of context. In fact, it is responsible for the entire context about a particular entity (person, place, or object). Aggregation facilitates the access of context by applications that are interested in multiple pieces of context about a single entity.

It is a great improvement thrust that the Context Toolkit promotes hiding the details of the lower-level sensors' implementation from context extraction, thus allowing the underlying sensors to be replaced or substituted without affecting the applications. However, since the "context" usage is far away from having any conventions to follow as in the case of computer GUI and keyboard/mouse usage, the idea of insulating sensors' implementation from context sensing cannot go much further with more and more sensors being incorporated.

1.2.2. Missing part: sensor fusion

Dey, the Context Toolkit author, listed 7 benefits (or fulfill 7 requirements [22]) to use the toolkit as of: (1) applications can specify what context they require to the system; (2) separation of concerns and context handling; (3) context interpretations convert single or multiple pieces of context into higher-lever information; (4) transparent distributed communications; (5) constant availability of context acquisition; (6) context widgets and aggregators automatically store context they acquired; (7) the Discovers support resource discovery.

Context-aware computing is about understanding and properly reacting to context information, its success will surely depend critically on the performance improvements of context sensing technologies. However, from context-sensing support point of view, the Context Toolkit system also has some limitations with regarding to lack of sensor fusion support:

No intrinsic support for context uncertainty indication: by default, any pieces of context information are regarded as correct and unambiguous, there is not a convenient way to indicate uncertainty or inaccuracy concerns in context sensing and presentation

No direct sensor fusion support: an application needs to query or subscribe all available sensor widgets that can provide it with interested context, and it is up to the application to decide whether there is any overlap or conflict between any two pieces of sensed context

Unnecessary difficult to develop further, when sensors' pool is large: Every sensor has this own widget and some sensed data need to be converted by interpreters before they can be used. When the sensor's pool is large, it is not easy for an application to arrange all possible context-providers to cooperate to trigger an event upon satisfying a given context situation.

No common-knowledge or world-modeling context support: Because there isn't any explicit context information to model the real world. it will not be straightforward for the system to use the such knowledge, as for example, to use a person's height information to help user identification tasks

Surely, for the sensors in a context-aware computing system to be configured dynamically and intelligently to work in concert (even merely at device-level such as who should talk to whom, when should they update status, etc.), as well as for some higher-level contexts to be extracted from lower-level contexts, sensor fusion support is required. Sensor fusion is supported in a system architecture means that, inside such infrastructure, some middleware is assigned to perform some administrative functions such as, tracking the current available sensors and their specifications, integrating and maintaining the system information flow, collecting relevant data, consolidating similar contexts and resolving conflicts, etc..

The motivation of this proposed research is to provide a Context Toolkit system with the missing part — the sensor fusion support. In the long run, our goal is to provide necessary tools or architectural building blocks to easily build a flexible infrastructure that can incorporate various context information to enable compelling user applications in smart office, home automation, or car information center.

To achieve this goal, we will specify a systematic sensor fusion methodology. We will illustrate that synergistic interactions between sensor fusion and context information can greatly facilitate the sensor fusion processes, which in turn can provide more context information and more accurate context information.

1.3. Context versus Sensor Fusion

"Sensor fusion" here is to produce context information, meanwhile, since "context" has such a broad definition, there is surely much information that can enhance the sensor fusion processes.

1.3.1. Sensor fusion frustration and opportunity

As described in last section, context can be anything from low-level parameters such time and temperature to high-abstract concept such as intention and social relationship. Using a human user centered approach, the context information can expand roughly along three dimensions as of:

- "**space**": the user's outside environment, the user's own activity in react to the environment, and the user's internal physical and mental status;
- "time": current time that is conventionally assumed appropriate for some activities, personal and related group's activity schedule, personal activity history and preferences;
- "social relationship": who is who, whom the user will likely care about.

For given environment constraints and target physical phenomenon, we may find ways to measure the target parameters, and work out some sensor fusion algorithms to optimize the measurement results according to some criteria. But context include so much contents and for some information such as mood and social relationship we don't know how to describe them in a computer-understandable way, here we are facing the sensor fusion frustration of a two-fold problem: (1) context representation (2) mapping sensor outputs into such context representation.

The most promising methodology to confront such frustration is perhaps to "divide and conquer": to decompose complex and abstract context information into less complex and lower-level discrete facts and events. For example, the "comfortable-ness to work" of an office room environment to a specific user may be represented by several "comfortable-ness grade number" depends on environment parameters (temperature, noise level, etc.) as well as the user preferences.

There may be more than one way to decompose complex context into more simple discrete facts and events, in doing this, along with the frustration there is a huge opportunity for sensor fusion to success too. That is, compared to the traditional parameter estimation sensor fusion, the pieces of context information are often correlated, — means that there is often "extra" context information that can be used to verify the correctness of the estimated hypotheses. For example, in user-identification applications, many sensors' observation can contribute strengthening or weakening such a hypothesis as of "this is user A": fingerprint reader, image pattern from a camera, infrared feature from an infrared camera, the user's speaking sound, the user's height estimation from a motion detector or a stereo camera pair etc..

1.3.2. Interactions can help

The purpose of sensor fusion is to provide context information at various abstraction levels, usually deriving higher-level context from lower-level contexts. The very basic function of sensor fusion, in traditional or classical sense, is to consolidate (verify, reduce uncertainty, and get rid of redundancy) information and to resolve conflicts. Different sensors will have different performances and will behave differently according to environmental change, and since the estimation of sensors' working environment is often included in our context information pool too, it is naturally desirable that we can use the context information to tune up the sensor fusion processes.

Using context information to adjust sensor fusion processes is considered to implement in two different ways. One way of such implementation is to adjust a sensor fusion algorithm's parameters, as when the conditions are changing in favor of one particular sensor we can raise that sensor's relative weight in voting mechanism. For example, when the environment gets dark around a car, we would then trust laser scanner sensors more than video cameras in detecting objects.

The other way to use context information to tune up sensor fusion processes is through intelligently specifying constrains to affect sensor fusion algorithms' behavior, or to appropriately choose one of the many sensor fusion algorithm candidates. For example, from vehicle location and velocity context information (e.g., downtown versus highway) we can know how fast objects can possibly move between two successive frames in object tracking. Further, in crowded downtown environment where the vehicle moves slowly, the system would be able to run more advanced object recognition algorithms to get more detailed information about its environment; whereas at highway when the vehicle runs at high speed, it has to do with its simpler counterparts to satisfy the requirement that every thing must be done very fast.

1.3.3. Layered and modularized architecture desired

At current context-aware computing research stage, as the number of sensors being used is not very large, the sensor fusion computation at all different levels is still manageable in an ad hoc manner. In a such system, for a given specific set of sensors, some system middleware or some applications will detect the validity of the sensed data and perform sensor fusion or context extraction in their own way — in whatever way it is regarded convenient: different sets of sensors would have different sensor fusion and context extraction framework.

With the number of sensors increases and with the number of levels of contextual information being incorporated into a context-aware computing system increases, it would be cost-prohibitive to maintain and further develop such a system. It seems that using layered and modularized architecture is the only way to develop flexible and complicated context sensing systems. We will use layered and modularized system architecture and will later show how it can enhance the interactions between sensor fusion and the sensed context information.

2. Contexts and Context-Sensing

As previously pointed out (section 1.3.1), to sense context for context-aware computing, we are facing a two-fold problem (1) how to properly represent context in a computer understandable way, and (2) how to map sensors' output into the context representation. To deal with context sensing, we will first discuss context representation and its uncertainty management problem, then, we will address the system architecture issues in presenting context information.

Context sensing technology can advance in two major directions: individual sensor's performance improvement with new sensor models and/or with new computational algorithms, or, systematic performance improvement based on intelligently connecting existing sensors. The later seems now playing a more important role with today's ever-growing computer networking development. This conclusion is made based on the survey of relevant sensor technology status and development, included in the appendix "Sensing and Sensors for Context-Aware Computing" (Fall 2000, by Huadong Wu).

2.1. Context Contents and Representation

To simplify context sensing and context usage, it is desirable that a context-aware system would have a modular and layered architecture. Such a system ought be built in such a way that applications should be insulated from the context extraction, and in parallel, the sensor-implementation should be transparent to context extraction & interpretation. This process should naturally begin with a context classification research.

2.1.1. Context classification

To completely describe or represent our colorful real world in a computer-understandable way seems an impossible task; we could possibly manage to deal with only very small part of the knowledge about a user's environment — if we have a very simplified model. To build such a context model of very simplified world and make it yet useful, we need first somehow classify the relevant information and sort out the useful contents.

As mentioned before, one systematic way to classify a user's context information is to explore in physical, spiritual, and time dimension. However, a pragmatic approach to get start on modeling a user's environment is to avoid those more abstract information that we don't know how to present and measure, and limit or condense the information in time dimension as of only about history experiences, schedules, and expectations. Table 1 shows an example of a user's environment context information with the pragmatic approach. More work on context classification can be found in the appendix "Towards Context Modeling: Context Classification".

After context being classified, the next step is to build a context model to present the context set, thus it naturally leads to a systematic way of context description and presentation. With such a context description methodology, we can define our context information architecture; and then under guidance of the context information architecture, we can in turn work out the necessary components to implement the desired system architecture.

outside (environment)					
location	proximity	time	people	audiovisual	computing & connectivity
city, altidude, weather (cloudyness, rain/snow, temperature, humidity, barometer pressure, forecast), location and orientation (absolute, relative to *proximity*)	close to: building (name, structure, facilities, etc. knowledge), room, car, devices (function, states, etc.), …, vicinity temperature, humidity, vibration, oxygen richness, smell	day, date	individuals or group (e.g. audience of a show, attendees in a cock-tail party): people interaction, casual chatting, formal meeting, eye contact, attention arousing; non face-to-face interaction	human talking (information collection), music, etc.; in-sight objects, surrounding scenery	computing environment (processing, memory, I/O, etc., hardware/software resource & cost), network connectivity, communication bandwidth, communication cost
change: travelling, speed, heading,	change: walking/running speed, heading	time of the day: office hour, lunch time, …, season of a year, etc.	interruption source: imcoming calls, encounting, etc., …	noise-level, brightness	
		history, schedule, expectation	social relationship		

Table 1. A pragmatic way to model a user's environment context information

2.1.2. Context representation

From the our discussion thus far, it is becomes obvious that by the term "context", we are more concerning discrete environmental facts and events instead of parametric numbers usually dealt with in scientific measurements. Though not excluding parametric numbers, which is simpler in representation format, our context model should allow the discrete context facts and events information to be described in a clear and efficient way. This means that the context should be decomposed to the extent that it can be represented in a format of some simple numbers, string descriptions or indices.

As an example, the format of context description in Table 1 may look like:

```
Context.Outside.Location.CityAddress =
   {name=”Pittsburgh”, e=(1.0, 1.0), t=LastUpdatedTime,
   $Description, weather=WeatherLink , GSP=sth, … }
Context.Outside.Location.BuildingAddress =
   {name=“Newell-Simon Hall”, e=(1.0, 1.0),
   t=LastUpdatedTime, $FunctionDescription,
   furtherInfo=InforLink, … }
Context.Outside.Location.RoomAddress =
   {name=”A415”, e=(1.0, 1.0), t=LastUpdatedTime,
   $Description=“Siegel’s Lab”, facility=FacilityLink, … }
Context.Outside.Location.Point =
   {PointOfStandingA, e=(0.6,0.9), t=LastUpdatedTime,
   PointerToNearbyDevices, …;
   PointOfStandingB, e=(0.3,0.9), t=LastUpdatedTime,
   PointerToNearbyDevices, …; }
Context.Outside.People = {”Tom”+”Carol”+”Alan”,
   furtherInfo=query(location.people), …}
```

```
    Context.User.Activity.Work = {TaskID, e=(0.6, 0.9),
        PointersToResource, ImportanceScale=0.5, Status, , … }
    Context.User.Inside.Concerntration =
        {FocusOfAttention, e=0.4, … }
```

Since context information has different sources originated from distributed sensors, it is easily understandable that there should be some context aggregation mechanism to collect context information about important entities such as the system users and the instrumented rooms and buildings etc.. The context aggregation will also promote layered structure in information gathering, so that context information usage is separated from context sensing implementation concerns.

To realize the idea of layered information architecture, each important entity in the system needs to have a dynamic context database storing all information about that entity and needs a context server to collect and serve the information. For example, each instrumented location center should have a server, running to collect and maintain the context information data about situations around this location; and each user should have a mobile computer going with him/her to collect and maintain his/her activity and physiological status data. Figure 3 illustrates context information gathering and storage in a context-aware system.[3]

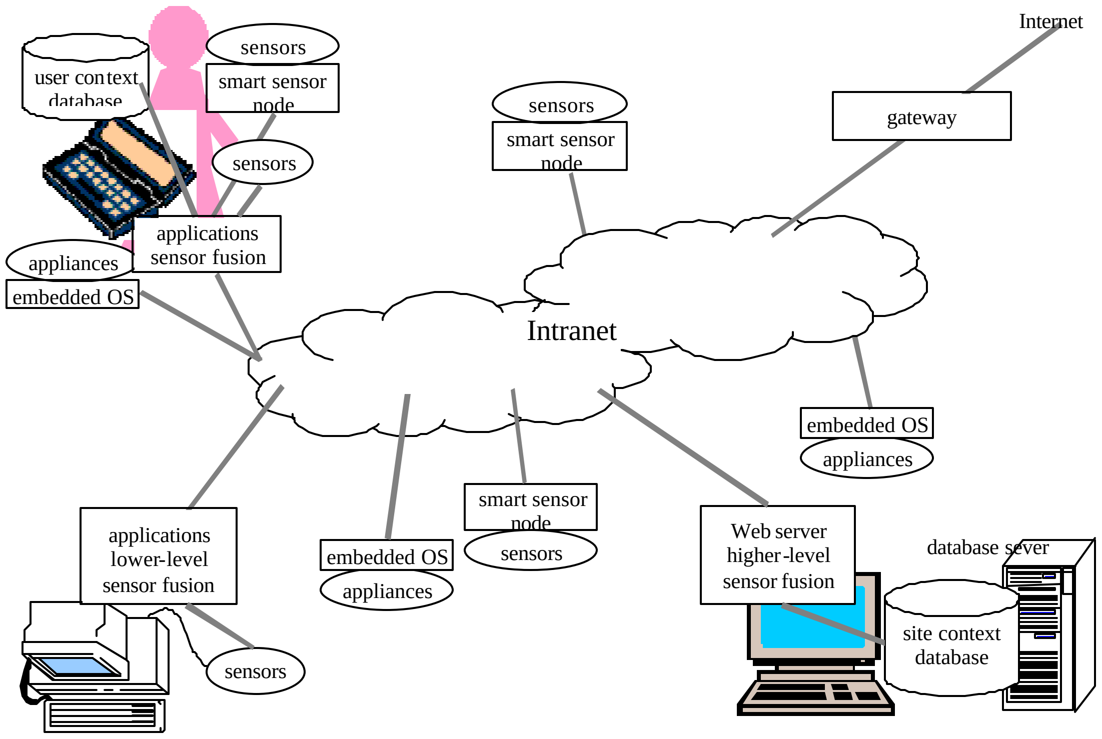


Figure 3. Dynamic context database supported context-aware computation architecture

[3] For general purposes, most application programs might be hosted in the users' mobile computer, accessing context information in both site context-server computer and user mobile computer databases, and sending out action requests to appropriate devices. However, for home automation applications, most applications would also be hosted in a local machine — as people are assumed in the most relaxed status in home and they may be reluctant to carry any such computational devices after all.

With this information architecture, all the communication and interactions inside the system can happen in a less chaotic way. Sensor fusion can be facilitated at all levels because the active agents (widgets, interpreters, and aggregators) have guidance as where to search for their necessary or interested resources. In addition, with the dynamic context database being built, it would also be easier for applications and sensor fusion agents to find their interested context cues because they now have the prior knowledge about what the data structure will be and what key words should be used in searching.

2.2. Context Uncertainty Management

As discussed in section 1.2.2, uncertainty is an intrinsic property coexisting with all sensing processes. Sensor fusion, in short, is mainly to deal with uncertainties — consolidating similar information and resolving conflicts. This section gives sensor fusion definition and classification, and discusses basic information combination methods in sensor fusion.

2.2.1. Sensor fusion and its methods

Different sensors have different resolutions and accuracies as well as different formats and update-rates; the sensed data may have overlaps or conflicts. “Sensor fusion”, often interchangeably called “data fusion”, is supposed to deal with these concerns, to meaningfully combine relevant information and resolve conflicts. U.S. Department of Defense Joint Directors of Laboratories Data Fusion Sub-panel recommended the definition of data fusion as of “data fusion is a multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from multiple sources” ([23], page 48).

Based on this recommendation and discussion in ([23] Chapter 1), sensor fusion can be defined as: the information processing that collects sensory data from multiple sensors, or from the same sources over a period of time, or both, to produce knowledge that is otherwise not obtainable, or that is more accurate or more reliable than information gathered from single systems.

From mathematics point of view, sensor fusion for context-aware computing is an operator to map sensor output into context information. The simplest format of information representation is a scalar number (for example, temperature measurement), which may be too simple to express many sensor output or useful context. Usually a vector is an efficient way to express quite rich of information content, so suppose we can represent our sensor output data in a vector $[s_1, s_2, \cdots, s_m]^T$, and represent the being measured context as discrete facts and events in a vector format as $[x_1, x_2, \cdots, x_n]^T$, then the sensor fusion processor is an operator $sf(\cdot)$ to fulfill the mapping.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \xleftarrow{sf(\cdot)} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}, \text{ or } \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = sf(\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}) \qquad \text{EQ.1}$$

The simplest operator is a linear combination of input to output, in which case the operator can be expressed in a matrix-multiplying format:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} sf_{11}(\cdot) & sf_{12}(\cdot) & \cdots & sf_{1m}(\cdot) \\ sf_{21}(\cdot) & sf_{22}(\cdot) & \cdots & sf_{2m}(\cdot) \\ \vdots & \vdots & \ddots & \vdots \\ sf_{n1}(\cdot) & sf_{n2}(\cdot) & \cdots & sf_{nm}(\cdot) \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix} \qquad \text{EQ.2}$$

For most our context sensing application, this linear mapping turns out to be an oversimplification. In next section, we will discuss some commonly used information combination methods such as Dempster-Shafer theory, which goes beyond linear mapping relationship.

Sensor fusion can be classified according to different criteria such as information level, implementation architecture, algorithms being used etc.. Frans Groen suggests classifying sensor fusion according to the nature of information input-output relationships, as of "complementary", "competitive", and "cooperative" three types. [24]

The **complementary type sensor fusion** fuses incomplete sensor data to create more complete model. One example of the complementary type sensor fusion is to match sensor data according a predefined physical model to estimate the state of being measured physical process, as in measuring pressure and airflow to estimate the pushing force of a jet nozzle.

The **competitive type sensor fusion** fuses sensor data that represent the same measurement types to reduce uncertainty and resolve conflicts. This is the basic sensor fusion type be used most extensively in various situations, and it is often regarded in the sense of "traditional" or "classical" sensor fusion technique.

The third type sensor fusion, **cooperative type sensor fusion**, fuses sensor data, where observations of one sensor depend upon that of another sensor, for example, image components depend on each other in pairs to estimate the corresponding object distances in stereovision.

The three sensor fusion types are not exclusive as many sensor fusion processes can belong to more than one type. Moreover, of the three sensor fusion types, the first and third types are generally more domain and problem specific, which means that their methods are often valid only when some specific knowledge and artificial intelligent inferences or reasoning techniques can apply.

Many well-developed algorithms can be generally applied to all sensor fusion types, some of them especially suitable for the competitive type. The commonly used sensor fusion methods are classical inference, Bayesian inference, the Dempster-Shafer theory of evidence, voting method, fuzzy logic etc. Of these methods, we are in favor of Dempster-Shafer because of its capability to solve general problems with reasonable computational complexity.

2.2.2. Dempster-Shafer theory of evidence method

In a Dempster-Shafer reasoning system, all the possible context facts (or events) of the same kind but being mutually exclusive are counted in as a part of "the frame of discernment T". For example, if we know that there is person in an instrumented room and we want to recognize whether this is the already registered user A, user B or user C, then, our "frame of discernment about this person is:

$$\Theta = \{A,\ B,\ C,\ \{A, B\},\ \{B, C\},\ \{A, C\},\ \{A, B, C\},\ \{\text{somebody else}\}\}$$

Each sensor, sensor s_i for example, will contribute its observation by assigning its believes over Θ. This assignment function is called the "probability mass function" of the sensor s_i, denoted by m_i. So, according to sensor s_i's observation, the probability that "the detected person is user A" is indicated by a "confidence interval"

$$[\ Belief_i(A) = \sum_{E_k \subseteq A} m_i(E_k)\ ,\ Plausibility_i(A) = 1 - Belief_i(A) = 1 - \sum_{E_k \cap A = f} m_i(E_k)\ ]$$

The lower bound of the confidence interval is the belief confidence, which accounts all evidences E_k that support the given proposition "user A". The upper bound of the confidence interval is the plausibility confidence, which accounts all the observations that does not against the given proposition.

Dempster-Shafer theory provides the rule to combine observations (denoted by its corresponding probability mass functions m_i, and m_j) from multiple sensors as:

$$m_i \oplus m_j(A) = \frac{\sum_{A_k \cap A_{k'} = A} m_i(A_k) m_j(A_{k'})}{1 - \sum_{A_l \cap A_{l'} = f} m_i(A_l) m_j(A_{l'})}$$

For each possible proposition (e.g., user A), the rule calculates all possible combinations of sensor s_i's observations m_i and sensor s_j's observations m_j that will lead to that specified proposition. The results are then normalized via being divided by a denominator, which accounts for all the impossible proposition combinations that have been assigned with some non-zero probability mass function.

The Dempster-Shafer decision theory is considered a generalized Bayesian theory, the canonical method to deal with statistical problems, in that it allows distributing support for proposition (e.g., this is user A) not only to proposition itself but also to the union of propositions that include it (e.g., this is likely either user A or user B). Compared with Bayesian theory, the Dempster-Shafer's capability to assign uncertainty or ignorance to propositions is very powerful to deal with a large range of problems that otherwise would be difficult to handle. For example, in the case that we use a camera to track people, after two people getting together and then separating, though the tracking system often get confused about "who is who", it is still valuable to keep the information that "this is either user A or user B". Another example providing a good reason to use Dempster-Shafer theory is that, it is very convenient to use information from sensors — information such as "it is highly probable (e.g. with a confidence of 0.9) that this is not user A though I don't know who this person is".

2.2.3. Other alternative methods

Classical inference and Bayesian inference

Given assumed hypotheses, classical inference can give the probability that an observation can be attributed to the presence of an object and event.

P(phenomena E_k observed | object or event H_i occurred)

Many decision rules can be used with the classical inference theory. For example, the Maximum Likelihood rule suggests accepting the hypothesis the *H_i* (the context fact or event *H_i* did occur) if the probability regarding the phenomena *E_k* observed satisfy:

$$P(E_k|H_i)\ P(H_i) > P(E_k|\neg H_i)P(|\neg H_i)$$

Otherwise, take it true that the context fact or event *H_i* did not happened.

Bayesian inference overcomes the limitations that only two simple hypotheses can be assessed at a time, the probability of hypotheses being true is not assessed, and likelihood of "hypotheses is true" is not updated from newly gained evidences.

Given the observed phenomena or evidence *E*, Bayesian inference calculates the likelihood of context fact or event *H_i* should have occurred *$P(H_i|E)$* in the form of:

$$P(H_i \mid E) = \frac{P(E \mid H_i)P(H_i)}{\sum_j P(E \mid H_j)P(H_j)}$$

Where *$P(H_i)$* is the priori probability that the context fact or event *H_i* would occurred; *$P(E|H_i)$* is the probability that the phenomena or evidence should be observed given the context fact or event *H_i* has occurred.

In practice, since it is often not easy to obtain the priori probability of hypotheses occurrences, they are often substitute with subjective probabilities. However, the output of such process is only as good as the input a priori probability data ([23], page 85).

Another difficulty makes Bayesian inference rule very hard to apply to combine output from multiple sensors with multiple possible hypotheses also. It is when there are more than two possible context facts or events, the combination of possible phenomena observed over multiple sensors is enormously large as it is exponential to bother the hypothesis number and to the status number of each sensors.

Voting fusion

Voting methods combine detection and classification declarations from multiple sensors by treating each sensor's declaration as a vote in which majority, plurality, or decision-tree rules are used. The general used voting architecture is Boolean combination of outputs from multiple sensors, though additional discrimination can be introduced via weighting each sensor's specific declaration.

The principle underlying voting fusion is the combining of logical values representing sensor confidence levels, which in turn are based on predetermined detection probabilities for an object. For a proposition of context fact or event occurring *H_k*, the inputs of voting fusion is the sensor *s_i* and *s_j*'s detection probability *$P_i(H_k)$* and *$P_j(H_k)$*, and their false alarm probability *$P_{fai}(H_k)$* and *$P_{faj}(H_k)$*, and the output is the detection probability *$P(H_k)$* and false alarm probability *$P_{fa}(H_k)$*.

$$P(H_k) = P_i(H_k) + P_j(H_k) - P_{i\cap j}(H_k)$$
$$P_{fa}(H_k) = P_{fai}(H_k) + P_{faj}(H_k) - P_{fa_{i\cap j}}(H_k)$$

Voting methods greatly simplify the sensor fusion process, and it can provide a prediction of object detection probability as well as false alarm probability. However, voting fusion, generally speaking, is more suitable to deal with "yes/no" problems like the classical inference method because its granularity of reasoning is usually not enough for multiple state context discrimination. For example, in the case of user identification, if the features of user A and use B are similar, it usually does help very much by just indicating probabilities of user A or user B being detected. Another disadvantage, when compared with Dempster-Shafer theory, is that voting fusion cannot properly use the information of negation such as “this must not be user A to the confidence of 80%”.

Fuzzy logic method

Fuzzy logic method accommodates imprecise states or variables. It provides tools to deal with the context information that is not easily separated into discrete segments and is difficult to model with conventional mathematical or rule-based paradigms. One example of such information kind is room temperature: though it is commonly referred with descriptions as “cold, warm, or hot”, it does not have hard boundaries to make decisions with regarding temperature status.

There are three primary elements in a fuzzy logic system, namely, “fuzzy sets”, “membership functions”, and “production rules”.

Fuzzy sets consist of the imprecisely labeled groups of the input and output variables that characterize the fuzzy system, as the “cold”, “warm” and “hot” status in the above room temperature example.

Each fuzzy set has an associated membership function to provide a representation of its boundaries. A variable of a fuzzy set has a membership value between the limits of 0 and 1, with 0 indicating the variable is not in that status and 1 indicating it is completely in that status. An intermediate membership means somewhat being regarded as in that status (e.g., being referred by some body but not others), and a variable may belongs to more than one fuzzy set, for example the room temperature of 90°F may be assigned as 0.25 “being warm” and 0.65 “being hot”.

Production rules specify logic inference processes in the form of IF-THEN statements, which are also often referred to as fuzzy associative memory. The output fuzzy set is usually defuzzified to convert the fuzzy values, represented by the logical products and consequent membership functions, into a fixed and discrete output that can be used by target applications.

There is a broad range of context information in our daily life, in which the boundaries between sets of values are not sharply defined, the events occur only partially, or the specific mathematical equations that govern a process are not known. With the capability to deal with this kind of information, and with its cheap computation to solve very complicated problems, the fuzzy logic method is expected to develop extensively in various context-aware computing applications.

Fuzzy logic sensor fusion would be a good choice to provide architectural context sensing support for further research and development. For now, we are in favor of Dempster-Shafer method only because fuzzy logic sensor fusion methods are more difficult to implement in a generalize-able frame, and it is doesn’t make sense to use membership function for the user-identification applications.

Neural network method

Neural network method opened a new door to intelligently fuse outputs from multiple sensors to produce desired behavior for various application situations. It can be though of as a trainable non-algorithmic black-box suitable for solving problems that are generally ill defined and otherwise require large amounts of computation power to solve.

The very good characteristic of neural network methods is that it can work in a high-dimensional problem space and generate high-order nonlinear mapping. Meanwhile, its very conspicuous disadvantage is that the mapping mechanism is not well understood even if it provides the desired behavior thus the solution cannot be generalized. In addition, to realize neural network sensor fusion, we have to overcome the local minim problem in training process.

Another very important factor affected us to choose supporting Dempster-Shafer sensor fusion instead of the potentially more powerful neural network methods is that neural network methods are generally not suitable to work in a dynamic configuration.

2.3. Context and System Architecture

The efficiency that a context-aware system senses and uses context information largely depends the way it handle the context information. In this section, we discuss the system architecture requirements to efficiently deal with context information.

2.3.1. The role of context in system

To simplify the maintenance task as well as to facilitate system development research, as discussed before, a context-aware system naturally requires a layered and modularized architecture.

There are many system architectures proposed and implemented to support context-aware computing ([22] section 3.4 Existing Support for the Architectural Features). From context acquisition and consumption point of view, they can be roughly classified as "context component architecture" and "context blackboard architecture" two categories.

The "**context component architecture**" thinks of the world as a set of context components corresponding to real world objects (such as users, facilities, devices, etc.); these components can interact with each other as the agents of real world objects. Most of research projects and developed systems use this architecture, perhaps because of its intuitively reflecting the real world. The main drawbacks of this architecture are its complexity in parallel programming and its higher communication bandwidth requirement.

Alternatively, the "**context blackboard architecture**" treats the world as a blackboard, where different types of context can be filled in and taken off. For small-scale context-aware systems, this is a very efficient architecture to realize less distributed computation requirements.

The context component architecture is believed to be the future of context-aware computing development because it is suitable for highly distributed ubiquitous computing applications, which are in turn believed to be the future to realize intelligent human-computer interactions. The Georgia Tech Context Toolkit system architecture belongs to this category, and this thesis research proposed context-sensing supporting system will develop along this direction although the sensor fusion mediator will use the backboard mechanism to some extent. The details are described in next chapter.

2.3.2. IEEE 1451 “smart sensor” standard concerns

In a traditional centralized measurement and control system, the applications’ programming code must know the specific information about the sensor devices’ address, registers, variables and units etc. This makes it difficult to add or change sensors in the system. With the dramatic performance advancement and price dropping of electronics and computer network systems in recent years, the concept of “distributed measurement and control” is quickly being accepted and implemented in industry, especially with the thrust of IEEE 1451 standards.

As shown in Figure 4., IEEE 1451 standard promotes the concept of separating sensor specific implementation details from application software by placing the sensed output data right at the sensors (http://www.motion.aptd.nist.gov/). The IEEE 1451.2 standard functionally splits a sensor node in a system into two modules as illustrated in Figure 5 (http://www.sensorsmag.com/articles/1097/ieee1097/main.shtml). The first module, containing a network capable application processor (NCAP), runs the network protocol stack and the application firmware. The second module, containing the sensor and a transducer electronic data sheet (TEDS), is called a smart transducer interface module (STIM). The two modules are connected through a standardized 10-pin transducer independent interface (TII).

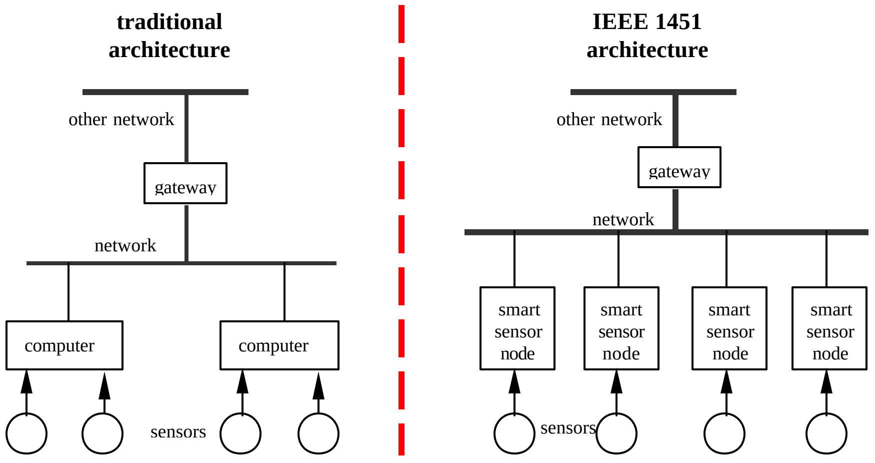


Figure 4. IEEE 1451 standards promote information abstraction by doing “signal to symbol translation” right at the sensors

As the sensors themselves (or more generally speaking, transducers, which refer to sensors and actuators) are considered to be an inseparable part of STIM, the included format-standardized TEDS can always provide the sensor information such as their manufacturer’s name, model series number, functional type, calibration data, and measurement specifications, etc.. Therefore, at the process connection level, this standardized system architecture can provide standard ways of creating totally self-describing measurement sensors and control devices. This mechanism allows the system to easily change its sensor configuration in a plug-and-play fashion.

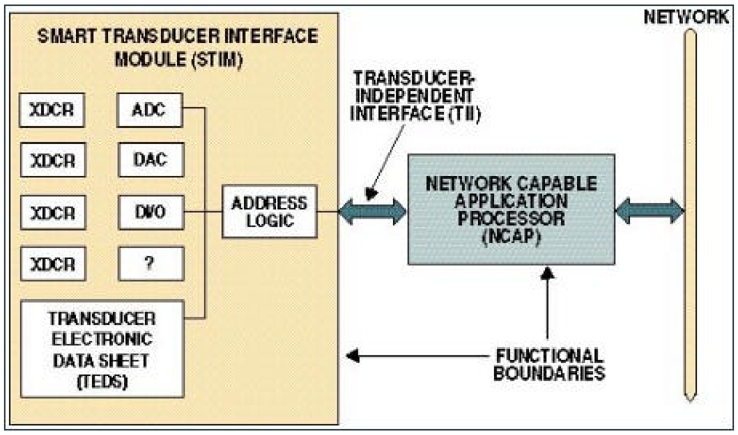

SMART TRANSDUCER INTERFACE MODULE (STIM)
XDCR
XDCR
XDCR
XDCR
ADC
DAC
DI/O
?
ADDRESS LOGIC
TRANSDUCER ELECTRONIC DATA SHEET (TEDS)
TRANSDUCER-INDEPENDENT INTERFACE (TII)
NETWORK CAPABLE APPLICATION PROCESSOR (NCAP)
FUNCTIONAL BOUNDARIES
NETWORK

Figure 5. IEEE 1451.2 Standard Smart Transducer Architecture

The IEEE 1451 standards have not been finalized yet[4]. So far, IEEE 1451.1 and IEEE 1451.2 have been approved in September 1997 and September 1999 respectively, and some pilot sensor products are emerging in market.

As a context-aware computing system usually needs to work in changing environments, by doing signal-to-symbol translation at the sensors instead of transmitting analog data and digitizing elsewhere, and by using a device self-identification mechanism as specified by the IEEE 1451.2 standards, the system integration and sensor fusion may become easier to implement.

However, this conclusion could be drawn only after a serious study with some experiments, because both context-aware computing and IEEE 1451 standards are in its early research and development stages and there is not much interaction across the two fields.

This study will be part of thesis work, and since sensors that have their own embedded signal preprocessor according to the IEEE 1451.2 Smart Sensor specifications are not commercially available yet, the necessary experiments will be done via software simulation, in which we will have to use a PC computer's general I/O ports to interface some of the sensors.

[4] The IEEE 1451 standards consist of four parts. Four technical working groups are formed in NIST (National Institute of Standards and Technology) to address different aspects of the interface standard. The P1451.1 working group aims at defining a common object model for smart transducers along with interface specifications for the components of the model. The P1451.2 working group aims at defining a smart transducer interface module (STIM), a transducer electronic data sheet (TEDS), and a digital interface to access the data. The P1451.3 working group aims at defining a digital communication interface for distributed multidrop systems. The P1451.4 working group aims at defining a mixed-mode communication protocol for smart transducers.

3. Supporting Sensor Fusion

This chapter discusses how we can provide architectural support to facilitate sensor fusion through synergistic interactions between the sensor fusion process and the sensed context. I am going to demonstrate the idea in two application scenarios: a user-identification system in an instrumented room in office or home environment, and an object tracking system around a moving vehicle.

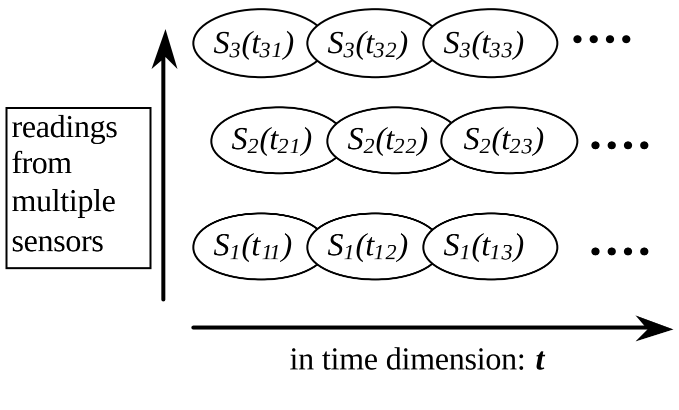
It was well perceived that context-aware computing research would inevitably incorporate heterogeneous equipment and sensors working over various computational environments. Almost all context-aware research projects have been built as computationally distributed systems, and perhaps it is from portability and scalability concerns the most commonly used communication languages are html compatible (Java, XML etc.). We are going to follow this convention also in building our primary concept-demonstration system. The secondary demonstration system is subject to project requirements to adopt Jay Gowdy's "symbolic sensor fusion" framework. [25]

3.1. Sensor Fusion Methodology

3.1.1. Fuse along two dimensions

In a traditional measurement and control system, to measure a system parameter, a sensor (or a set of sensors) is assigned to take the measurements. We can always develop and deploy high-quality highly reliable sensors to measure important parameters, however, from cost-effectiveness consideration, an alternate way is to improve measurement reliability or accuracy by adding measurement redundancy.

As illustrated, there are two possible ways to make redundant measurements: either we can take multiple measurements with one sensor or we can use multiple sensors to measure the same parameter. Therefore, sensor fusion can realize either along time dimension to fuse multiple measurements from the same sensor or fuse readings from multiple sensors.



$S_3(t_{31})$ $S_3(t_{32})$ $S_3(t_{33})$ ....
$S_2(t_{21})$ $S_2(t_{22})$ $S_2(t_{23})$ ....
$S_1(t_{11})$ $S_1(t_{12})$ $S_1(t_{13})$ ....
readings from multiple sensors
in time dimension: ***t***

Figure 6. Two possible ways to make redundant measurements, sensor fusion can along time dimension or cross multiple sensor outputs

In practices, it is often implemented in both dimensions. Moreover, typical practices are to filter some noises along time dimension, and then only the "cleaned" data are processed as "sensor fusion" in traditional senses. The most commonly used sensor fusion technology is the statistically weighted averaging with the smaller standard deviation measurements proportionally gaining heavier weights.

The traditional sensor fusion scheme works efficiently under the generally assumed conditions that the sensor set is predefined and their configuration will not change in multiple measurements over extended period of time.

To build a platform with its architecture generally supporting sensor fusion, we face two challenges:

A. The sensor set's configuration in a typical context-aware computing system is usually not static: the users with wearable mobile computing devices can constantly come and go and, different places would have very different instrumentation and computing environments;

B. Sensor fusion is no longer confined in the traditional sense of "weighted averaging": the sensor output measurements may come from very different perspectives, some higher-level context information is derived from intelligently combining lower-level sensor outputs and contexts as well as from artificial intelligent inference algorithms.

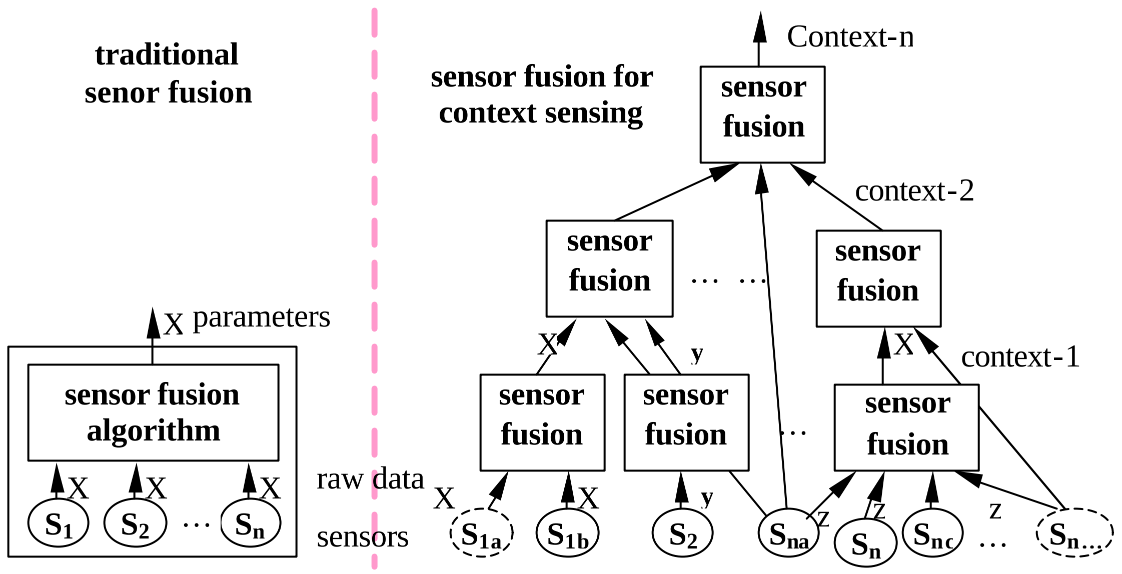


Figure 7. Traditional sensor fusion versis sensor fusion for context sensing

The promising way to take the two challenges is wrap sensor's low-level outputs as early as possible, converting the raw sensory data into more symbolic context representation right at sensor' interfaces. The implementation will be clarified in next section.

3.1.2. Sensor fusion procedures

In traditional sensor fusion methodology, there are three steps to proceed:

I. All sensor's output data are converted into an internal representation, this internal representation is common to all sensors

II. Sensor fusion is realized through combining these sensor data in the internal representation format in an meaningful way, the most commonly used is weighted averaging

III. The result of sensor fusion is converted to required representation if necessary.

Though at higher level to combine context information, an architecture that can generally support sensor fusion (or context sensing) needs also following the similar methodology as:

I. Define context information architecture and representation format, which should be discrete contextual facts or events so that the sensor's observation can easily represented

II. Combine sensor observations to generate context information, this is done with the help of a sensor fusion mediator

III. If necessary, convert context information representations or extract higher-level context from multiple lower level contexts

The realization of this methodology is discussed in the next section.

3.2. Sensor Fusion Realization

3.2.1. Information layer and function modularization

As discussed before, the key element that is missing in the Context Toolkit system is the capability to deal with the intrinsic errors and uncertainties in sensing and sensors. This incapability makes it generally treat context information as of either "yes" or "no" two states. To move from the so-called "pure symbolic" field into the "statistical reasoning" realm, the first step is to carefully build up an information architecture for given an application scenario. This is as if we prepare ourselves with a good blueprint to build a house.

Suppose we have successfully designed the context information architecture such that the interesting context information has been decomposed into discrete facts and events, with each sensor's widgets generating such context fact and/or event observations according to the specified templates. The proposed system architecture will have its modular components as shown in Figure 8.

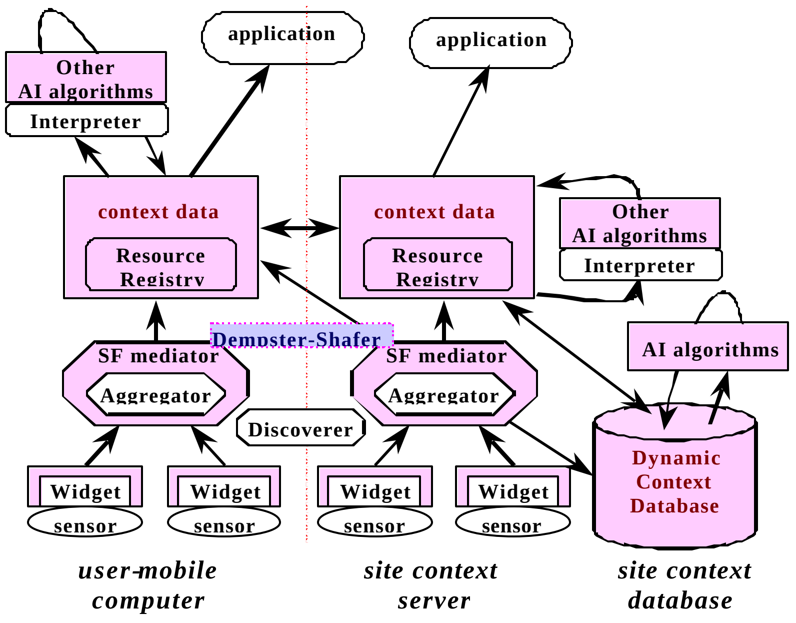


Figure 8. System architere to support sensor fusion in context-aware computing

Now, from the very low level — sensor widgets output — and up, any piece of measured or interpreted context has an associated number indicating its estimation confidence. Sensor fusion then is automatically realized when combining context observations: overlaps and cross-verifying information will increase the estimation confidence, whereas conflicting signals will decrease the associated estimation confidence.

The way we provide generalize-able architectural sensor fusion support is this: for each kind of context information, there is a sensor fusion mediator to coordinate the process. The sensor fusion mediator behaves like a specially function-enhanced Context Aggregator in a corresponding Context Toolkit system.

We propose to standardize the interfaces of the sensor fusion mediator modules, but their internal design is up to the specific nature of being sensed context as well as the sensors to be used. As discussed before, we choose to first support competitive sensor fusion type using Dempster-Shafer theory to statistically combine multiple sensors' output.

The system performance is expected to benefit in the following aspects:

- To directly support competitive type of sensor fusion in information mapping: for most cases the much more difficult sensor fusion task becomes an easier task of recalculating confidence. At various level of abstraction, the Dempster-Shafer theory will allow us to freely combine all sensors' observation with any confidence level, -- even the information such as ignorance of about a proposition.
- The complementary sensor fusion type and the cooperative sensor fusion type are indirectly supported through the adopting centralized context aggregation backed up with dynamic context databases. Here, context interpreters are no longer working with specific context widgets. Hence, it will be easier for artificial

intelligent agents to access the context data to derive more abstract, higher-level context information.

- The context information usage is further separated from context acquisition, making applications more resilient to the influence of system hardware configuration change, and the system is easier to scale up.
- The dynamic context database service will make the complex context situational abstraction easier to be implemented.

3.2.2. Information propagating and updating

The information layered structure and the sensor fusion supporting mechanism is illustrated in Figure 9.

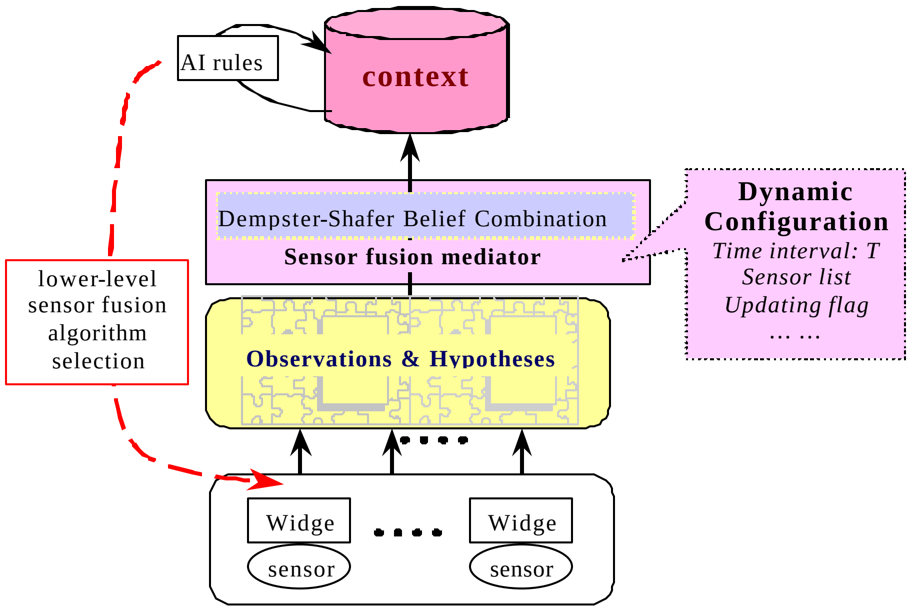


Figure 9. Information layered structure and sensor fusion support

To regulate sensor fusion system information updating, for every piece of context information from low-level sensors' output and up, there is always an associated time stamp to indicate when that piece of information was sensed or deduced. Each sensor fusion mediator keeps the information of: a list of all sensors that generate its corresponding kind of context information observations, the normal time interval needed to update the context from the list of sensors' observation, and an updating flag to indicate that the system is busy to update its context information.

A sensor fusion mediator gets the relevant sensor list from the system Resource Discoverer at initialization process, and subsequently, all newly coming sensors will report its availability to the relevant sensor fusion mediators and will try to announce its unavailability if possible. Any sensor in the list that first reports an observation may trigger the information updating process, but in general, the sensor fusion mediator will decide when and how to update the information and appoint some sensors to act as active triggering source.

An updating process begins with the sensor fusion mediator flapping the updating flag, and it then starts to query all available sources according to the sensor list. In the case that the updating process detects its sensor configuration changes, it will update the sensor list and adjust the estimated time interval needed to update context information next time.

3.3. A User-Identification Implementation

The user identification implementation will be the primary demonstration system to demonstrate the feasibility of generalize-able context-sensing support.

3.3.1. Context in the system

User identification is one of the most important features needed for all kinds of context-aware computing: even merely with this sensing capability, lots of functionality in home automation can be implemented. Some such application examples are: secured keyless entry, adjust facilities (lights, music, temperature, and other utilities) to the user's taste when the system recognizes that user has come home, etc.

Though people-identification is one of the most basic problems in context-aware computing application, in practice, it is still not very much well solved in terms of reliability and un-intrusiveness. Besides the multiple input modalities' dynamic configuration challenge, this sensor fusion application scenario by itself will be an interesting research topic.

At its first stage, the dynamic context databases include the following contents:

- Environment of the instrumented room: room name, building name, the room is divided into two areas: entrance area and inside area, equipment in the room, temperature, light and noise level, etc.
- Registered user's basic information: name, email addresses, age, height, preferences (music, temperature, light etc.), experiences, other information that the user would like to let the system know for potential applications, etc.
- User activities: current position in the room (entrance or inside area), activity history in this room etc.

Most of the background information about the instrumented room and the users is manually entered. As we are more interested in user identification and tracking, so those users that are currently detected in the room with associated confidence level above some criterion are listed in the user table.

The context information is served with a dynamic context database. For each record, the contexts that describe static facts (such as name, age etc.) will have fixed fields in the database, and the rest that describe dynamic contexts usually will be a link to another context table.

The output of the demonstration system is dynamic context information about the current status of room occupants — who is at where in the room, how much is the confidence interval about the facts, when the user was first detected, etc.. For those detected users, their background information and possible interests are loaded to the system, or the information links are checked to be ready to load the information.

3.3.2. System configuration

The instrumented room has the following sensors and facilities:

- A motion detector mounted at the entrance of the room

- A regular camera mounted near the room entrance pointing to door, mainly for user identification and moving object detection
- A wide-angle infrared camera mounted at a back corner of the room, pointing at the front door entrance, mainly for user tracking and scene monitoring
- Several microphones distributed in the room
- A fingerprint reader mounted beside the door-bell push button
- The system will have active badge system infrastructure installed, some registered users will have active badges or some physiological sensors that support wireless LAN connection

The system configuration is illustrated in Figure 10.

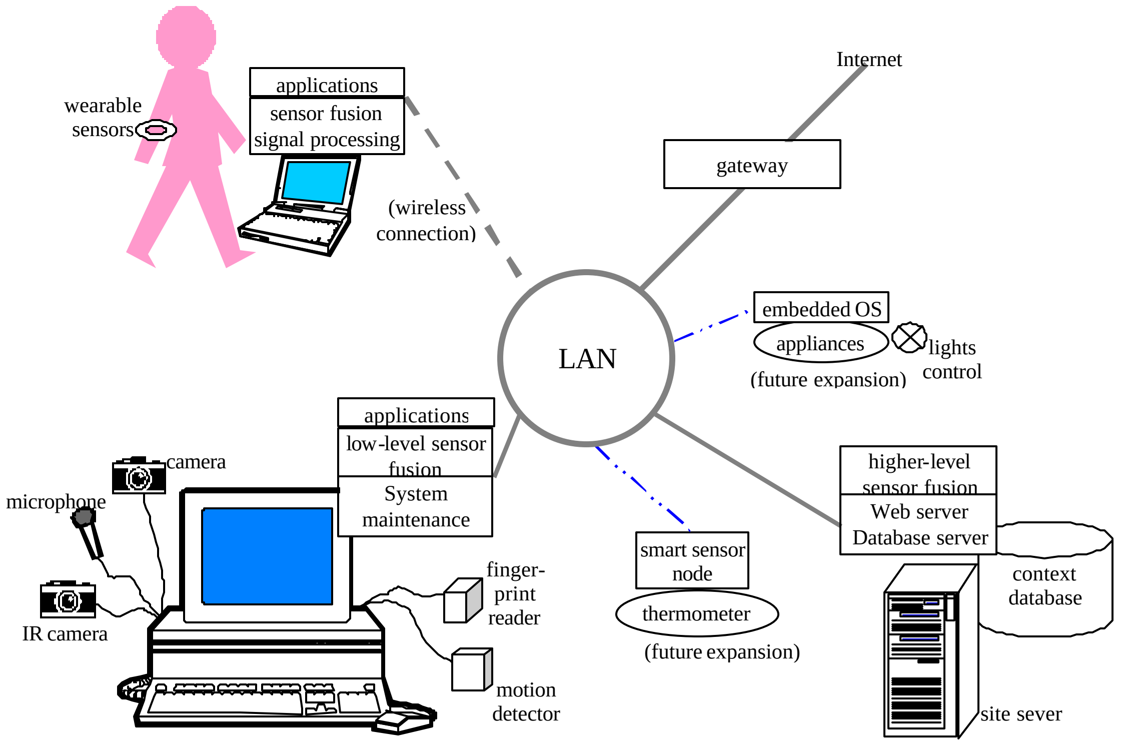


Figure 10. Concept demonstration system configuration

For the demonstration system, the room site machine will host both the dynamic context database to provide services and the sensor fusion mediators to coordinate the user identification processes. Another one or two site PC machine will be assigned mainly to host sensor widgets or to provide simulated widgets.

Some users may carry a mobile computer. In future the user wearable computer will be able to interact with the site context server all the time, to perform intensive sensor fusion computation in order to sense the user’s activity and intention, and maintain his/her personal context database. However, to ease the workload requirements so that the research will not lose its focus, for our demonstration purposes, it mainly serves interfacing the wearable physiological sensors.

3.3.3. Demonstration, software package, and documentation

This primary concept demonstration system will have all the components described in previous section. The developed software package is to be used in context-aware research projects in the Motorola Research Lab, Schaumburg, Illinois. Besides the software package and its installation instructions, the deliverable will also include description about the system architecture, instructions to use the software to incorporate new sensors, and guidelines to develop new sensor fusion mediators.

We can imagine a demonstration scenario showing how the idea works.

- A group of people are in the instrumented room discussing some sensitive topic as a presentation slide is shown on the screen
- When an authorized user query, he/she can know who are in this room, how long a particular user has been there, the current environment conditions, or what facilities are available there
- A registered user, John, approaches the room entrance, his carried active badge interacts with the system, and with other sensors' cooperation, the system does not have any trouble to recognize him with a reasonably high confidence level, this is shown on the site context server
- Depends on the his identity being recognized, the confidence level, and the sensitivity level of the material, the presentation may be kept on the screen according to regulations while John comes into the room
- Another user, Tom, approaches the room entrance; the system has some trouble to recognize him as he doesn't wear a badge, so the context information looks something like: a person of average height (99% of confidence), Tom (40% of confidence), and Jim (30% of confidence)
- Since the regulations only allow Tom to have access to the material, so the presentation slide is substituted with a wallpaper on the screen;
- Tom notices the indication that he was not well recognized, so he claims his identity to the system by pushing one of his fingers against the fingerprint reader. Now, the context server shows user information as: Tom (with 90% of confidence) and Jim (with 5% of confidences), and the presentation slide goes back on the screen.

This demonstration scenario only illustrates how we can build context-aware computing system with sensor fusion support, facilitated with dynamic context databases and under the guidance of the proposed information architecture. It shows how the system components cooperate to perform the desired sensor fusion functionalities. The full utilization of the available contexts (e.g., temperature, some activity patterns, etc.) is up to the applications — facilitated by layered information architecture and the modularized component infrastructure.

3.4. An Object-tracking Implementation

This will be the secondary concept-demonstration implementation system to show that interactions between context information and sensor fusion processes, or to be more specific, relevant context information can facilitate to tune up the sensor fusion processes.

This research is to be used by Robert Bosch Corporation Research Center North America, Pittsburgh, Pennsylvania.

3.4.1. Application and system description

This application is to enable a rich sensor instrumented car (shown in Figure 11) to have awareness about its environment (within 360° around, 7 meter side and back coverage, and 150 meter ahead coverage), classifying and tracking moving objects such as curbs, trees, pedestrians, other vehicles, and lanes etc..

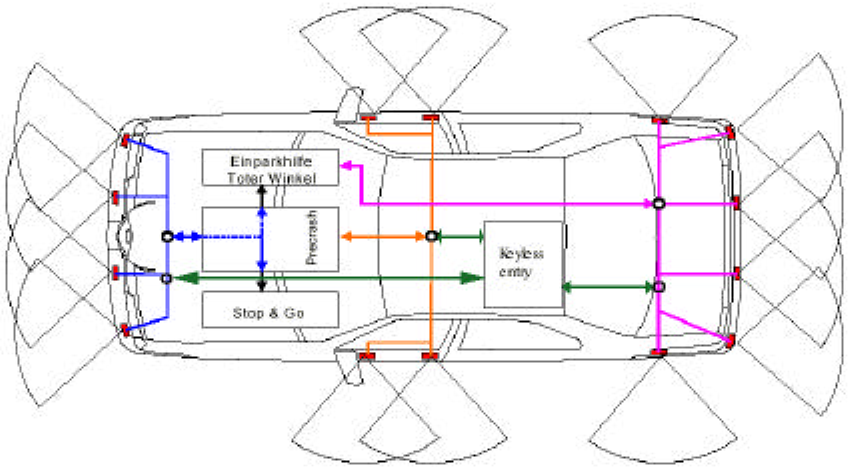


Figure 11. An instrumented car for Local Vehicle world Map (LVM): object-tracking and environment understanding

Contrast to a system for user identification and tracking to be used in home automation or smart office applications, the sensor fusion for Local Vehicle World Map (LVM) task has a static sensor configuration with the sensors type and number as well as their relative positions fixed.

The system may include sensors of short range radar, far range radar, Lidar, laser scanner, laser line striper, optical flow from omni camera, lane tracker, stereo video cameras, and possibly ultrasonic distance detector. Actually, we are not so concerned about the low-level sensors' implementation because what we are going to deal with are preprocessed data transferred from the so-called CAN bus (or Control Area Network bus of a vehicle), our inputs are data sequences of the clustered observed points of objects.

The required context information to be sensed is just the clustered points, representing objects, which are better defined with clearer constraints than those possible context contents in other context-aware computing systems. So, a non-distributed architecture is adopted for sensor fusion in the project. The system architecture framework is so-called "Symbolic Sensor Fusion" as shown in Figure 12 developed by Jaw Gowdy. [25]

From context sensing and consuming point of view as discussed in section 2.3.1, this sensor fusion should be categorized as the "context blackboard architecture".

The way the architecture works can be summarized in four steps:

1. Sensor modules produce observations: as shown in Figure 12, radar sensors can generate object distance readings, a lane tracker can generate lane observation

with location information, and a laser striper sensor can generate object observations with boundary information;

2. These observations are fed into a hypotheses pool;
3. Hypotheses in the pool will strengthen with new matched data and decay without further support over time;
4. The unused observations will generate new hypotheses in the hypothesis pool.

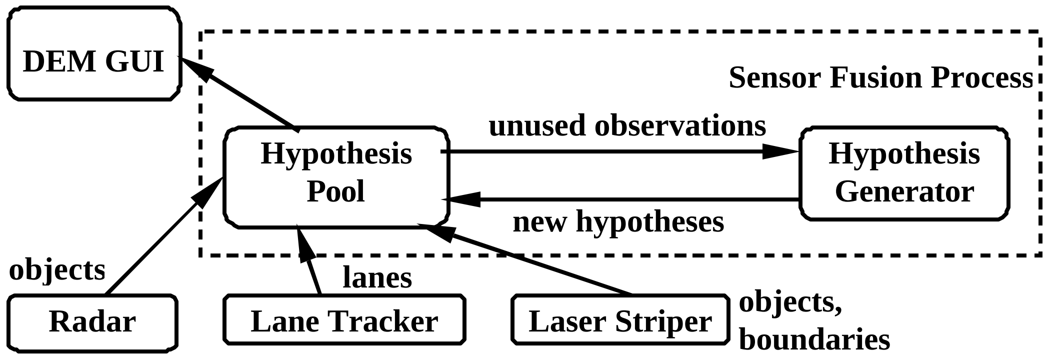


Figure 12. The "Symbolic Sensor Fusion" framework (Jay Gowdy)

The sensor fusion results are the hypotheses with associated uncertainty indications (confidence for discrete facts, or covariance for parameter estimation) in the hypothesis pool, which can be queried by or actively update context the consumption components.

From sensor fusion point of view, as discussed in section 2.2.1, this architecture is suitable to provide competitive type of sensor fusion support.

3.4.2. Using context to tune up sensor fusion processes

Following our proposed context analysis methodology, a list of interesting context information may include:

- Object hypotheses from the sensor fusion hypothesis pool, possibly classified objects and their location and velocity information
- Vehicle's own driving status: pose and orientation, speed, turning radius, etc.
- Nearby driving conditions: road quality (roughness, traction), other vehicles' traveling speed, traffic congestion, etc.
- General environment condition: weather, visibility, location (downtown versus highway), outside temperature etc.
- The driver's physiological and spiritual status etc.

Of course, we won't be able to get and use all these context information. This list suggests what we aspects we should look in considering using context information.

At the first stage, with regarding to object tracking task, we can start with using the vehicle's own speed and local speed limit information.

Because of the economic constraints, the sensors to be used are cheap; naturally, their output data are very noisy and unreliable. For each sensor module, for all detected object points in time sequence, an attempt was made to assign a number to each detected object (a point in observation pool) in order to track objects and facilitate the object recognition tasks. Therefore, to fuse measurements from different sensors, it is necessary to correlate detected points cross all sensor outputs at proximately the same time.

By now, the only suggested method to find matching points is by checking distances of point pairs against a preset number. In the object number matching processes, if the preset criterion number is too small we will easily lose track of objects, on the other hand if the criterion number is too large, we will risk cross-numbering objects.

The above described "symbolic sensor fusion" framework in a sense is actually a very vague outline, as it doesn't provide advices on how sensor observations should correlate the hypotheses in the hypothesis pool, nor does it provide any supporting tools to help the operation of strengthening hypotheses with observations or decaying hypotheses over time.

In general, the idea of using the context information to facilitate sensor fusion is to be implemented in two ways:

- The first way is to use the context information to help the matching processes of context hypotheses versus context observations. For example, the vehicle's own driving speed information, together with the nearby vehicles' maxim driving speed information from its environment, can be used to set object-tracking constraints to a finer degree thus to improve sensor fusion algorithm's reliability.
- The other way is to use the generated context information to choose sensor fusion algorithms, for example, to switch between sensor fusion algorithms used in highway high-speed driving conditions and those used in urban downtown slow-driving conditions.

The first stage of implementation is to use the vehicle's own velocity and local speed limit information. By using the vehicle velocity and environment context information, we will be able to calculate constraints regarding moving objects dynamically, thus enhance object tracking. In addition, the current joint-effort research is to use Kalman Filter algorithm to eliminate noises to get better object position and velocity estimations.

Further research should take other more context information into account, such as how weather and lighting condition will affect specific sensor's tracking abilities, thus provide more accurate error estimation for sensor fusion weighting processes.

4. Working Plan and Expectations

4.1. Schedule and Expectation

The research work actually began since my last year's summer internship job in Motorola Research Lab, Schaumburg, Illinois. I work for Bosch Research Center North America, Pittsburgh, Pennsylvania this summer, and is now work on the secondary concept-demonstrate system.

So far, the pre-thesis proposal works as of literature search and system preparation have made reasonable progress. The secondary concept-demonstration research is expected to have a software demonstration, which has all the necessary components working with some prerecorded data by end of August 2001.

The proposed first-stage research is planned to follow the schedule described below:

- By May 15, 2001: building system computational architecture
    - Digest Georgia Tech's Context-aware Toolkit, set up a context-aware system architecture using the Context-aware Toolkit, make sure system components communicate properly
    - Set up database server and web server services, preliminary experiments on context information repository mechanism
    - PhD thesis proposal[5]
- By December 15, 2001: sensor network development
    - Choose sensors and do experiments on simple-case sensory row data to context information mapping — programming enhanced sensor widgets, some of work will be software simulation
    - Adopt or evaluate the IEEE 1451 smart sensing network standards on some of the given sensors, this is done mainly through software simulation
    - Create or adjust context information repository mechanism using dynamic context database services
- By May 15, 2002: sensor fusion study
    - Context taxonomy and knowledge presentation study, sensor fusion mediator programming
    - Implement context information structure model for given application scenario, stipulate and program sensor-to-context mapping mechanism in the given situation — test and improve sensor widgets and the sensor fusion mediator
    - Programming to implement Dempster-Shafer evidence combination mechanism for sensor fusion mediator
- By December 15, 2002: system refinement and evaluation

[5] The first attempt turned out to be a disaster, this is revised version to clarify points.

- System architecture refinement and evaluation study, given the application scenario, evaluate performance improvement with respect to sensor fusion and system scalability
- System documentation, PhD thesis and defense.

4.2. Concept-demonstration Expectations

The focus of the proposed research is to demonstrate a top-down methodology, where under the guidance of a predefined context information architecture model the competitive type of sensor fusion can be automatically realized in the information mapping process, and sensor fusion in general can be greatly improved by context provided that we have a proper software system architecture support.

As illustrated in Figure 13, put it in a big picture of sensor fusion, the proposed architecture can support competitive type sensor fusion directly.

- At lower level, generally multiple measurements are fused so that the sensor's Widget reports its observation to the system sensor fusion mediator at a slower frequency and at a higher level of semantic abstraction.
- At higher level, the same type of observations from all the Sensor Widgets are fused using statistical combination methods such as Dempster-Shafer Theory of Evidence algorithm.

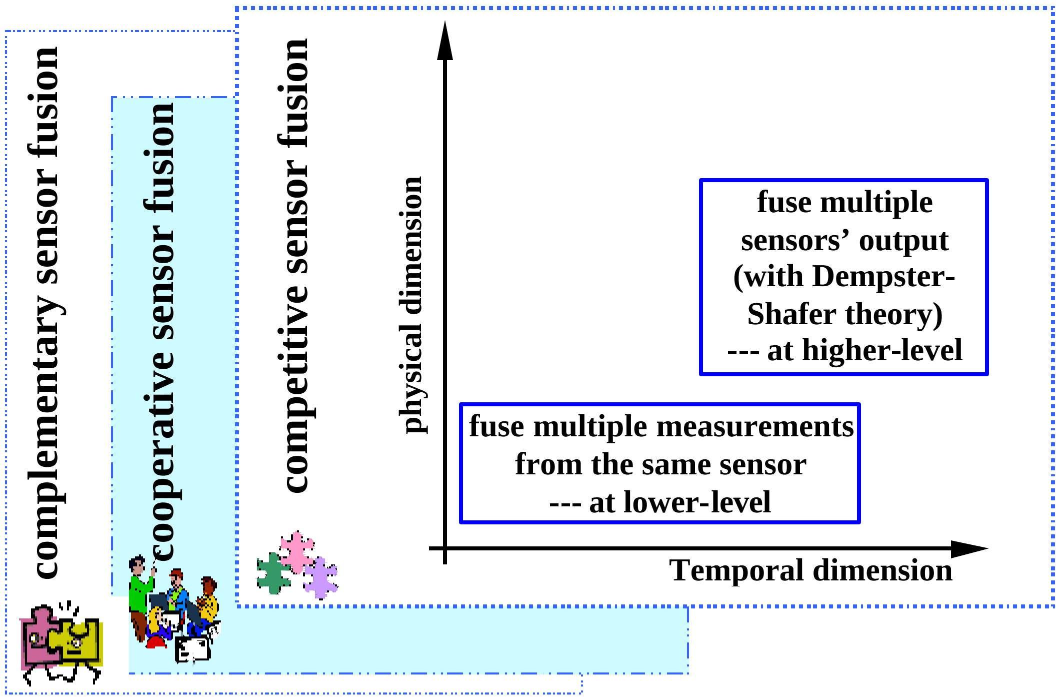

complementary sensor fusion
cooperative sensor fusion
competitive sensor fusion
physical dimension
fuse multiple sensors' output (with Dempster-Shafer theory) --- at higher-level
fuse multiple measurements from the same sensor --- at lower-level
Temporal dimension

Figure 13. Sensor fusion supporting in the big picture

As each specific sensor's performance improvement will be a PhD thesis research topic by itself, here, I plan to use available existing resources as much as possible, or to use software to simulate as virtual sensors, so that I can focus my research on enabling interactions between sensors.

To evaluate the success of this thesis work, three parts of the work will be examined:

A. A demonstration of working system with the above specified elements;

B. A software package with corresponding system description documents; and

C. Thesis to explain why this methodology is chosen, to articulate the assumption it takes, the benefits and limitations of this architectural support scheme, and to defend that though with these limitations this is a generalize-able methodology.

4.3. Expected Contribution, Future Work

This proposed thesis research work is expected to contribute to knowledge base of the whole world of science and engineering in the context-aware computing and sensor fusion domains as:

- Demonstrate the idea that, under the guidance of well-defined context information architecture, synergistic interactions between sensor fusion and context information can greatly support and promote each other.

  Currently most context-aware computing systems are built in an ad hoc approach, i.e., their system architectures are heavily influenced by the specific applications they run. This thesis research work reverses to the correct order of "from information architecture to system architecture implementation".

- Propose and demonstrate the idea of supporting generalize-able sensor fusion processes in translating data to semantic representations in an information structure.

  Sensor fusion method, in the traditional sense of parameter evaluation to seek a statistically optimized estimation, cannot be generalized. However, by extracting higher-level context from sensor output data, the process can be realized in consolidating symbolic information through statistically combining uncertainty representations.

- Conceptually improve the Georgia Institute of Technology's Context Toolkit system architecture in system performance and usage adaptability.

  The biggest contribution of GIT Context Toolkit is to promote the separation of context from its acquisition. This proposed thesis research work overcomes its limitations pushes its capability envelop with sensor fusion support in 1) providing uncertainty and ambiguity information to user applications; 2) consolidating information and resolving conflicts for the users; 3) using context information to improve sensor fusion process; 4) taking care of sensors' dynamic configuration; and 5) supporting complex context specification services.

The proposed context-aware research project is not meant to claim a complete solution, but rather to test an idea of system architectural improvement. There are many opportunities to be fully explored when further application researches are specified, such as researches regarding sensor uncertainties, learning from history information etc..

This research is partly funded by Motorola through the University Partnership Program, cooperating with Applications Research Lab, Schaumburg, Illinois.

5. References

[1]. Anind K. Dey and Gregory D. Abowd, “Towards a Better Understanding of Context and Context-Awareness”, Proceedings of the CHI 2000 Workshop on “The What, Who, Where, When, and How of Context-Awareness”, The Hague, Netherlands, April 1-6, 2000. ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf

[2]. Bill N. Schilit and Marvin M. Theimer, “Disseminating Active Map Information to Mobile Hosts”, IEEE Network, 8(5) 1994, 22-32, http://citeseer.nj.nec.com/schilit94disseminating.html

[3]. Schilit, B.N., Adams, N.L., Want, R., “ContextAware Computing Applications”, Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994. IEEE Computer Society. http://citeseer.nj.nec.com/schilit94contextaware.html

[4]. William Noah Schilit, “A System Architecture for Context-Aware Mobile Computing”, Ph.D. thesis, Columbia University, 1995, http://citeseer.nj.nec.com/schilit95system.html

[5]. Mari Korkea-aho, “Context-Aware Applications Survey”, http://www.hut.fi/~mkorkeaa/doc/context-aware.html

[6]. Roy Want, Andy Hopper, Veronica Falcao, Jonathon Gibbons, “The Active Badge Location System” (ftp.uk.research.att.com:/pub/docs/att/tr.92.1.pdf), http://www.cam-orl.co.uk/ab.html

[7]. Want, R.; Schilit, B.; Adams, N.; Gold, R.; Petersen, K.; Goldberg, D.; Ellis, J.; and Weiser, M., “The ParcTab Ubiquitous Computing Experiment”, Xerox Parc technical report, http://citeseer.nj.nec.com/want-parctab.html

[8]. Sue Long, Rob Kooper, Gregory D. Abowd, and Christopher G. Atkeson, “Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study”, Proc. 2nd ACM International Conference on Mobile Computing (MOBICOM), Rye, New York, U.S. http://www.cc.gatech.edu/fce/cyberguide/pubs/mobicom96-cyberguide.ps

[9]. J. Pascoe, “Adding Generic Contextual Capabilities to Wearable Computers”, 2nd International Symposium on Wearable Computers, Pittsburgh, Pennsylvania USA, October 19-20, 1998, page 92-99. http://www.cs.ukc.ac.uk/pubs/1998/676/index.html

[10]. Ryan, Nick, Pascoe, Jason, and Morse, David R., “FieldNote : a Handheld Information System for the Field”, First International Workshop on TeloGeoProcessing (Telegeo'99), Lyon, 1999, http://citeseer.nj.nec.com/nick99fieldnote.html

[11]. Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven Shafer, “EasyLiving: Technologies for Intelligent Environments”, Handheld and Ubiquitous Computing, September 2000. http://www.research.microsoft.com/easyliving/Documents/2000%2009%20Barry%20HUC.pdf

[12]. Steve Shafer, “Ten Dimensions of Ubiquitous Computing”, Keynote presentation at Conference on Managing Interactions in Smart Environments, December 1999.

http://www.research.microsoft.com/easyliving/Documents/1999%2012%20Ten%20Dimensions.doc

[13]. Kidd, Cory D., Robert J. Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E. Starner and Wendy Newstetter, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research", Proceedings of the Second International Workshop on Cooperative Buildings - CoBuild'99. Position paper, October 1999. http://www.cc.gatech.edu/fce/house/cobuild99_final.doc

[14]. Robert N. Johnson, "Building Plug-and-Play Networked Smart Transducer", Sensors Magazine, October 1997, http://www.smartsensor.com/doc/sensors.pdf

[15]. Stan P. Woods, "The IEEE-P1451.2 Draft Standard for Smart Transducer Interface Modules", Hewlett-Packard Company presented at Sensor Expo Boston, May 1997.

[16]. Anind K. Dey, Jennifer Mankoff and Gregory D. Abowd, "Distributed Mediation of Imperfectly Sensed Context in Aware Environments", GVU Technical Report GIT-GVU-00-14. September 2000. ftp://ftp.cc.gatech.edu/pub/gvu/tr/2000/00-14.pdf

[17]. Anind K. Key, Daniel Salber, Gregory D. Abowd, and Masayasu Futakawa, "An Architecture To Support Context-Aware Applications", GVU Technical Report GIT-GVU-99-23. June 1999, ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-23.pdf.

[18]. Martin Grötsch and Joachim Lügger, "Scientific Information System and Metadata",

[19]. Mark Stefik, "Introduction to Knowledge Systems", Morgan Kaufman Publishers, Inc., 1995, ISBN 1-55860-166-X

[20]. Mongi A. Abidi and Rafael C. Gonzalez (editors), "Data Fusion in Robotics and Machine Intelligence", Academic Press, Inc. 1992, (ISBN 0-12-042120-8, CMU E&S 629.892 D232)

[21]. Gregory D. Abowd and Elizabeth D. Mynatt, "Charting Past, Present, and Future Research in Ubiquitous Computing", ACM Transactions on Computer-Human Interaction, Vol. 7, No. 1, March 2000, Page 29-58. http://www.cc.gatech.edu/fce/pubs/tochi-millenium.pdf

[22]. Anind K. Dey, "Providing Architecture Support for Building Context-Aware Applications", PhD thesis, November 2000, http://www.cc.gatech.edu/fce/ctk/pubs/dey-thesis.pdf

[23]. Lawrence A. Klein, "Sensor and Data Fusion Concepts and Applications" (second edition), SPIE Optical Engineering Press, 1999, ISBN 0-8194-3231-8

[24]. Frans Groen, "Sensor Data Fusion" presentation, 11/3/99, http://www.science.uva.nl/~arnoud/OOAS/Presentation9fus/

[25]. Jay Gowdy, "Emergent Architecture: A Case Study for Outdoor Mobile Robots", PhD thesis (CMU-RI-TR-00-27), November 1, 2000, http://www.ri.cmu.edu/pub_files/pub2/gowdy_jay_2000_2/gowdy_jay_2000_2.pdf

[26].