The title of my PhD thesis research proposal is: supporting sensor fusion for context-aware computing. Briefly, I am going to study the sensor fusion supporting problem in context-aware computing. I want to 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.

I am going to demonstrate the idea in two application scenarios: one is to identify and track people/users in an office or home room; the other is to track objects around a moving vehicle. The developed high-level sensor fusion for context-aware computing software tool package will be used in Motorola Research Lab in Schaumburg; and the research of lower level sensor fusion algorithm tuning up with context information to track objects around a moving vehicle will be used in Robert Bosch Research Center North America.
As showing here, a traditional context-sensing system usually has its sensors and the sensed context information highly coupled. Typically, it can only use very limited context information, and since its architecture largely depends on the sensors it uses, it is very difficult to substitute existing sensors and to add new sensors to incorporate more context information.

The Context Toolkit system developed in the Georgia Institute of Technology promotes separating the sensed context information from its sensors through using sensor widgets, modularizing and standardizing software interfaces. This makes replacing existing sensors or adding new sensors easier, thus it facilitates adopting more and complex context information. However, it lacks the capability or mechanism to address the measurement uncertainty problems essentially existing in sensing and sensors, thus it cannot properly handle information overlapping and resolve conflicts from multiple sensors.

And my proposed research is to provide generalize-able sensor fusion architectural support in the context sensing process. My proposed system will be able to cope with the intrinsic uncertainty problem in sensing process, thus can better handle information overlapping issues and resolve conflicts from multiple sensors.

Surely, I am not claiming that I can solve all sensor fusion problem in context sensing, like Context Toolkit, I am only addressing higher-level context information, and my approach is mainly using the statistical combination methods in the information mapping processes. My work is based on the assumption that we can build a context model such that more complex context can be decomposed as discrete facts and events.
Expectations of the Proposed System

Expected Performance Boost

1. Uncertainty & ambiguity representation to user applications
2. Information consolidation & conflict resolving for users
3. Adaptive sensor fusion support switch to suitable algorithms
4. Robust to configuration change — and for some to die gracefully
5. Situational description support — using more & complex context

The proposed sensor fusion mainframe architecture looks like this. Here, based on the assumption, according to a simplified context information model sensors’ output can be preprocessed into more abstract but discrete context facts or events; and away from the Context Toolkit’s general entity abstraction, a sensor widget may generate multiple distinct observations or hypotheses, behaving like several sensor-widgets in a corresponding Context Toolkit system. With each kind of predefined context fact or event presentation, we build a sensor fusion mediator in the system to statistically combine beliefs from different sensors. For the first stage research in the user-identification demonstration system, I propose and will later adjustify using the Dempster-Shafer theory to manage uncertainty properties.

This sensor fusion mediator will also coordinate all the sensors that can generate its corresponding kind of context, appointing some sensors to behave as active sensors, arbitrating when to trigger then actually performing the information propagating and updating processes.

Further, by implementing entities’ central context information aggregation through dynamic context databases, it would be easier to apply artificial intelligent algorithms to extract more and more complex context information, and adaptive sensor fusion schemes such as using different sensor fusion algorithms according to the change of context sensing conditions will be naturally realized. As indicated here.

So I expect to improve a Context Toolkit system performance in the following respects as shown here: 1) provide uncertainty and ambiguity information about all sensed context to user applications so that they have an idea about how much they should trust the sensed context hence provide references about how to use the information; 2) all information sources of the same kind are consolidated and conflicts hopefully being resolved; 3) relevant context information is used to adjust sensor widget behavior so that the sensing process can adopt to environment change more easily; 4) the sensor fusion mediator can coordinate the sensor pool’s dynamic configuration; and 5) a context information server with centralized context aggregation can support using more complex context.

O.K. Before step into details of the implementation plan, lets start with a brief introduction and terminology clarification.
The foremost, what is sensor fusion? Sensor fusion, or often interchangeably called data fusion, usually refers to 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 sensor systems.

Different people may have a different interpretation when referring the term “sensor fusion”. From mathematics point of view, if we describe sensors’ measurement output in the form of a sensory data vector, and describe the desired outcome, in our case the so-called context facts and events, in the form of another vector -- context status vector, then sensor fusion is an operator to transfer, or to map, sensory data vector into context status vector. The simplest operator is linear transformation, whereas the mapping function can be described in the format of matrix.

By “providing sensor fusion architectural support”, here I mean to facilitate system software implementation which includes three steps: 1) to define the context status vector according to the particular application’s requirements, to define sensory data vector according to available sensors, the sensors’ output format, resolution, and updating rate, etc.; 2) to seek suitable mapping function that can possibly realize the mapping process; and 3) to write software modules to facilitate implementing such an information mapping.
The idea of “ubiquitous computing” and “context-aware computing” is to have computers understand our real world so that human-computer interactions can happen at much higher abstract level --- like human-to-human interactions. Context-aware computing research, or more accurately, context-aware human-computer interaction research, was started in the early 1990’s, and it is still at its preliminary stage. Currently most research activities address the areas of “smart office”, “home automation” and “car information center”. For continuity and as a practical matter, I too will use office or home room environment as well as automotive application as my test bed.
Context-Sensing: the Heart of Context-Aware Computing

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

• 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


Context-aware computing is a very broad research field, according to the researchers in Georgia Tech, it can be classified as three categories: 1) presentation of information and service to a user; 2) automatic execution of a service; and 3) tagging of context to information for later retrieval. To enable context-aware computing, context-sensing is of course at the heart of such a system.

However, a systematic sensor fusion research under the context of context-aware computing is hardly up to the challenge yet, although there are many individual sensor fusion research activities going on. This situation is perhaps partly due to the fact that, context is still a “divergent” concept --- that is, people still don’t know how to model our real world into computer-understandable “context”.
Reflecting our real world, a user’s situational context information can be anything about: roughly, in physical space dimension, his outside environment, his own activity, and his physiological status; in temporal dimension, his past experiences and his planned activity schedule; and in spiritual dimension, his feeling and intention, desire, social relationship etc.

Because we don’t know how to describe various abstract context information, it seems impossible for us to measure it. So, to enable context-sensing, we are facing a twofold problem: 1) context representation and 2) the mapping of sensor outputs into such context representation.

Anyway, perhaps the most promising approach to challenge such a seemly insurmountable task is to use the “divide and conquer” methodology. I expect to explore both side of the problem and focus on the interaction between them, with which I will discuss the limitation of my approach and suggest future work.

Just as an illustration here, this math equation format assumes a linear combination between sensor readings and a context state vector supposed somehow representing context information, which I believe is an oversimplification that should be replaced by more complex nonlinear relationship such as Dempster-Shafer theory of evidence.

One thing I would like to point out is that, along with this kind of context sensing frustration and difficulties, there also exists a huge opportunity there, that is, we often have various additional "context" information to support the sensor fusion process.
The proposed context-sensing methodology is: first, to classify the commonly used context information that we are interested and can now possibly measure; second, for a given application scenario, to carefully build context information architecture that can model this subset of the real world; and third, sensor fusion then becomes a mapping from sensor’s raw output data into the more abstract context information model.

The situation is much like this: now, the Context Toolkit has contributed building blocks, but for building a good house, we also need a blueprint of detailed architectural specifications. This advocated methodology is analogous to using an information architectural blueprint for building your context-aware system.

With such a context information model, sensor fusion in context-aware computing will be much more manageable and relevant information will be more readily used. Of course, the real world is not that simple to model. So let’s begin the process with a simplified context classification.
For example, a user’s environmental context may be classified as: location (city and street name etc., including links to local news, weather and other information), proximity (close to: a building, a room, a car, etc.), people around the user, audiovisual communication activities, computing facilities and connectivity status etc. etc..

Context classification and real world modeling is an open-end problem, I will only touch the surface of it, such that, for a simplified environment (an instrumented room, and a vehicle surrounding environment, for example), **pragmatically**, we have a consistent context information description, and the context information pool should be scalable --- that is, it should be very easy to add new context information.

We need software infrastructure support to realize the sensory data to context information mapping. Before we discuss building supporting infrastructure, lets first check what is special in sensor fusion for context-aware computing.
Sensor fusion for context-aware computing is unique as listed here, because of its dynamic characteristics and because of its for human-computer interaction purposes. In this area, as I mentioned before, there is also a big opportunity there. A very important point is that how we should present and aggregate context information to provide extra support to the sensor fusion process, which in turn provides more and better deduced context information.

Due to the difficulties in context-sensing, most context-aware computing research projects now only use very limited context information, such as user identification, location, and time. And till now, the context-aware computing systems largely have been developed in ad hoc manners: that is, their components are highly coupled and their system architectures are heavily influenced by the the way the context information is derived.
For a context-aware computing system to easily scale up, that is, for more sensors and applications to be easily added, it is highly desirable that the sensed context information be separated from the sensors’ measurement implementation. This way, the application software will more concentrate on better using contexts, instead of struggling how to deploy sensors to implement context sensing.

Towards this modularized system design, there are many system architectures proposed and implemented. 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. Alternatively, the “context blackboard architecture” treats the world as a blackboard, where different types of context can be filled in and taken off.

There is a tradeoff between the two architectures, what we want is to build a hybrid system that can best support the intelligent sensor fusion process, and the Context-aware Toolkit developed in Georgia Tech can provide a good software toolkit to help building such a system.
Georgia Tech’s Context Toolkit provides the basic architectural building blocks to build context-aware systems, as analogous to using standard GUI API software components to develop user interfaces. As illustrated, the architectural building blocks are: “Context Widget”, “Context Interpreter”, “Context Aggregator”, “Context Service”, and a “Context Discoverer”.

Context widgets are responsible for collecting information from sensors. They provide an abstraction that allows the sensed information to be passed without applications and widgets having to know the details about each other. Context aggregators collect related context together, each aggregator is usually about a particular entity in real world. Context interpreters convert context information from one format into another, for example, from GPS’s latitudinal and longitudinal parameter pairs into city and street location names. Context services provide reusable context-aware behaviors or services to applications, they can be part of highly-coupled sensor-actuator pairs.

The “context discoverer” provides support for locating context components. When widgets, aggregators and interpreters are instantiated, they register themselves with a discover, thus allows the system to track all components’ working status.

Communications among context components can happen in either of the two methods: one component can query another component about the specified attributes, or, one component can set up a conditional callback from another component.
The Context Toolkit was developed most in Java. Its component abstraction framework is shown here.

The BaseObject class provides the basic communication functionality: sending, receiving, initiating, and responding to communications. To handle simultaneous incoming communications, the BaseObject uses a multithreaded server. When it receives information, the server’s current thread handles the incoming request and it forks a new thread to listen for future communications.

User applications can use the BaseObject instances to communicate with the system infrastructure; and components within the infrastructure can sub-class from the BaseObject to communicate with other infrastructure components.

Context widgets, interpreters, and discovers subclass from the BaseObject and inherit its communication functionality.

Service is part of the Widget object.

Aggregators subclass from widgets, inheriting all the functionality of widgets.
Georgia Tech Context Toolkit System:

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Context specification</td>
<td>1. No intrinsic support for sensing uncertainty</td>
</tr>
<tr>
<td>2. Separation of concerns and context handling</td>
<td>indication</td>
</tr>
<tr>
<td>3. Context interpretation</td>
<td>2. No sensor fusion support</td>
</tr>
<tr>
<td>4. Transparent distributed communications</td>
<td>3. Unnecessary difficult to develop further when</td>
</tr>
<tr>
<td>5. Constant availability of context acquisition</td>
<td>sensors’ pool is large</td>
</tr>
<tr>
<td>6. Context storage</td>
<td>4. No common-knowledge or world-modeling context support</td>
</tr>
<tr>
<td>7. Resource discovery</td>
<td></td>
</tr>
</tbody>
</table>

The Context Toolkit author listed 7 benefits to use the toolkit, three of which, I think, are much more significantly appreciable: separation of concerns and context handling; transparent distributed communications; and resource discovery. From context-sensing point of view, however, it has certain limitations too.

The first limitation is that, by default, any pieces of sensed context information are regarded as correct --- there isn’t a convenient way to indicate the uncertainty or inaccuracy concerns in sensing and sensors.

The second limitation is that it has no 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.

Limitation 3, 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. We will discuss this problem later.

And limitation 4, 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.
Sensor fusion types roughly can be classified as three categories: the category 1 --- complementary sensor fusion --- is to piece information fragments together to get a more complete picture of the target; the category 2 --- competitive sensor fusion --- is to reduce uncertainties and resolve conflict information through combining multiple reading resources; and the third category --- cooperative sensor fusion --- is to derive hidden information using artificial intelligent techniques to reason or infer facts. An ideal context-aware context sensing system should have them all.

Because of the above mentioned limitations, a simple Context Toolkit system just cannot properly handle sensor fusion problem. For example, in the “in-out board” application system which is supposed to dynamically show who has entered and left a building and when, in the case that the sensors mounted at the entrance of the building separately reported that user A, user B, and an unidentified user was detected in short time, considering that a person being detected can either mean his entering or leaving the building, or just greeting to another person, then it would be very difficult for the system to figure out whether user A, or user B, or both, or three people have entered the building.

Of all the listed limitations of a Context Toolkit system, the key element that is missing is that it did not take the essential characteristics --- the intrinsic error and uncertainty property in sensing and sensors --- into account in system design.
Actually, the context toolkit author suggested an extension to the Context Toolkit to overcome some of the mentioned limitations. It was suggested to introduce “Organized Option Pruning System” mechanism and a mediator to solve the inaccurate context problem. The basic idea is that the ambiguous context that cannot be used by the application is first fed to a mediator, which requires the user’s further clarification and allows the original widget to have the additional new information, this process goes on until the ambiguity is resolved. This method of using hierarchical graph representing possible interpretations may be suitable to resolve ambiguity in speech recognition, but it is obviously not a generalizable solution for context-sensing systems.

For the third limitation, the scalability limitation, Georgia Tech proposed “Situation Abstraction” solution to deal with complex context situation. An example of this kind of complex context situation is to arrange to make a call to a remote family which has two adults and their several children when: at least one parent is at home and have more than one hour free time in schedule, nobody is sleeping, the family have already had their dinner. This kind of complex context specification requires lots of communication with various context widgets and aggregators in the remote site. The suggested extension of context toolkit, or situation abstraction, could simplify the application development process by specifying the required context situation to the context-sensing infrastructure as a whole. But, unfortunately, it was realized through modifying the system BaseObject class’ function capability as shown here, making the BaseObject unnecessary too complicated. Which I don’t think is a good solution.
My proposed architectural sensor fusion supporting system can overcome the above mentioned limitation 3 and 4. The idea is just to use dynamic situational context databases to help aggregate context information centrally.

To implement this, each significant entity (a user, a instrumented room, a house etc.) has its own dynamic context repository --- backed with a database server. With the database server support, the dynamic context repository virtually maintains and serves to applications all the commonly used context information about this entity: including the information model of the entity as well as the sensed and interpreted context.

The very basic characteristics of sensor fusion is to convert all sensors’ outputs into an information representation such that it is common to all sensors, thus make combining the outcome together possible. Here, obviously using dynamic context database can not directly provide sensor fusion tools to the system. But it can indirectly support sensor fusion category 1 and category 3, --- as it makes sensor fusion easier: for the complementary sensor fusion and the cooperative sensor fusion categories, it makes relevant information more conveniently available.
My proposed research is to directly support sensor fusion category 2 --- the competitive sensor fusion type.

As I mentioned 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, as we move towards having computers understand our real world, the very first step as being advocated before (slide 8) 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 now have a well designed 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. 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 done when sensors’ raw output data are mapped into the context information architecture: overlaps and cross-verifying information will increase the estimation confidence, whereas conflicting signals will decrease the associated estimation confidence.

This idea is realized through implementing Dempster-Shafer evidence theory: we are going to program the proposition belief function combination rule algorithms in the information mapping process.
Before addressing how to use the Dempster-Shafer theory of evidence in our system, let's first take a step back to check why we had chosen to support the competitive type of sensor fusion, and what could we get if we choose other kind of sensor fusion techniques rather than Dempster-Shafer theory of evidence.

Of the three previously introduced sensor fusion types, methods and processes in both complementary and cooperative types can not be easily generalized or standardized because they are usually application-specific or domain-specific, where priori knowledge is necessary.

The competitive type is the most basic, most commonly used type, and it includes many well-developed effective methods to various applications and they are usually easiest to be standardized or generalized. We choose to support competitive type of sensor fusion because we think this is appropriate start point and it is suitable for our user identification applications.

Under the umbrella of competitive sensor fusion type, there are lots of techniques suitable for various application situations. Here we list the most commonly used methods with highlights of advantages and shortcomings with comparison with Dempster-Shafer theory, in light of our user-identification applications.

We chose Dempster-Shafer theory of evidence as our statistical reasoning workhorse because it has some advantages over other methods. The very important advantage that is suitable for general context sensing applications as well as for this user identification task is that, it enables us to easily incorporate uncertainty or partial ignorance information about propositions from multiple sources of sensor observations, combine and reason them as a whole with estimated measure confidences, meanwhile without resorting to more rigorous real probability density functions which are hard to obtain in our applications.

As a matter of fact, depending on information architecture, Dempster-Shafer can often actually support both complementary and competitive types of sensor fusion.
**Competitive Sensor Fusion with Dempster-Shafer Theory**

- Frame of discernment $\Theta$: 
  \{ [user A], [user B], [user A or B], [neither user A or B] \}

- **Belief** $(A) = \sum_{E \in \Theta} m(E)$

- **Plausibility**$(A) = 1 - \text{Belief}(\overline{A}) = 1 - \sum_{E \cap A = \emptyset} m(E)$

- Updated belief $m_1 \oplus m_2(A) = \frac{\sum_{A \cap E_j = A} m_1(A) m_2(E_j)}{1 - \sum_{A \cap E_j = \emptyset} m_1(A) m_2(E_j)}$

In a Dempster-Shafer reasoning system, all the possible context facts or events of the same kind but being mutually exclusive are counted as the frame of discernment $\Theta$. For example if we need to recognize two users in an instrumented room, we will have the 4 possible states of being #1 user A, #2 user B, #3 its believed to be either user A or user B but don’t know exactly which one it is, and #4 its neither user A nor user B, it must be somebody else.

The system will use confidence interval to measure its uncertainty. The lower bound of the confidence interval is the belief confidence, which accounts all evidences that support the given proposition. The upper bound of the confidence interval is the plausibility confidence, which accounts all the observations that does not against the given proposition.

In our user identification example, for each sensor that can detect people, it will report its observation with an associated confidence estimation number. Say, a camera may report that it believes the detected person is user A with 40% of confidence, well, meanwhile it might be user B instead as well with 30% of confidence. The the system will then conclude that the likelihood the detected person is user A is 40 to 70%.

Meanwhile an infrared camera would report its observation, for example, as of with 60% of confidence in belief that its user A, and with 20% of confidence is might be user B as well. So the Dempster-Shafer rule will account the new observation support or contradict the previous proposition, adjusted by impossible combination situations, here, it is not possible that it is both user A and user B.
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 (side 18), we choose to first support competitive sensor fusion type using Dempster-Shafer theory to statistically combine multiple sensors’ output.

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.

A 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.
To summarize, the proposed context-aware computing system with the sensor fusion architectural support looks like this. The white blocks are inherited from the Context Toolkit context-aware building blocks.

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

Benefit 1, directly support competitive type of sensor fusion in information mapping: for most cases the much more difficult sensor fusion task becomes a easier task of recalculating confidence. The sensor fusion scheme in our proposed demonstration system will be firstly implemented using Dempster-Shafer rule to combine sensors’ observation. 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.

Benefit 2, 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. With this, context validation is easier to check. For the example given before, the user’s basic information such as height information is more readily to be used.

Here, context interpreters are no longer working with specific context widgets. Using this architecture, it will be easier for artificial intelligent agents to access the context data to derive more abstract, higher-level context information.

Benefit 3, 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.

And the last, the dynamic context database service will make the previously described context situational abstraction easier to be implemented.
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.

The hypothesis that sensor fusion for context-aware computing can be greatly facilitated with a well-designed information architecture is going to be demonstrated in two application scenarios.

One of the planned demonstration is to identify and track users in an instrumented room. The room site context database will have the registered or learnt background information about the regular system-users as well as the room environments. The output of the system is a dynamic context information about the current status of room occupants – who is at where in the room. Such context information can provide support for user activity or information access control etc applications.

To detect and track registered users, the system will include the following sensors: a motion detector mounted at the room entrance; several microphones distributed in the room; a camera mounted near the room entrance pointing at the door, mainly for user identification; an infrared camera with wide-angle lens, mounted at the back corner of the room, pointing at the room entrance, mainly for monitoring the whole room to track users; a fingerprint reader mounted at the room entrance; and some users may have a wearable physiological sensing device, or an active badge that has wireless connection capabilities, so the corresponding infrastructure needs to be built also.

We can imagine to demonstrate the user identification and tracking system in the scenario of: a group of people are in the instrumented room discussion some sensitive topic showing on the screen, when an authorized user query, he/she can know who are in this room since when. When John come into the room, the system shows that a newcomer has shown and most likely this is either John or Tom. Since by default, only John is allowed to see this document, so the screen turns to wallpaper automatically. Noticing this change, John put his finger on the fingerprint reader, o.k., the document is back again.
Following the previous articulated methodology, given an application scenario, our first step is to carefully build up an context information architecture to model all possible situation.

The context information architecture model described in rough granularity is in the form of classification tables or corresponding graph trees to indicate “what belongs what” and “what has what” relationships. With that, then the final context model is implemented in the context database, the format is shown here.

In the user identification and tracking demonstration, the instrumented room table lists some basic information about the room and the current status as of temperature, noise level, light condition, devices available, number of people detected, areas that have been divided to describe the situation in more detail, etc..

Follow the area table, it can be found that the room has been divided into two areas: the entrance area, and the inside area. Again follow each area’s description table, the more detailed situational information or context facts can be found.

Most of the basic information about the instrumented room is manually entered. So is the basic information of registered users, though some items such a user’s preference or habits can be learned from some artificial intelligent algorithms, which may be yet another one’s thesis topic.

As we are more interested in user identification and tracking, so those users that are currently detected in the room with its associated confidence level above some criterion are listed in the user table. For each user record, besides the fields that describe facts such as which area in the room, when was first detected, etc., there are also links to other relevant information about that user, such as the basic background information, preferences, etc etc..
When a person or a group of people approach the room entrance, the system notices that there is a moving object in the scene and will try to exam whether the moving object is due to people’s activity? if so, how many people are there? which pre-registered user or users are there, etc. from all available sensor modalities.

To be more specific, this is kind of context information will be specified in the system: moving objects detected; whether there are some people there; if people are detected with confidence above certain criterion, how many people are here; if it is believed with confidence above some criterion that a single person is there, who is this person; if it is believed with confidence above some criterion that there are two people there, who are these two people, etc. etc..

Also shown in this table is a very rough indication about which sensor’s individual observation can contribute to the context information mapping. 0 means it can hardly make this kind of observation, 1 means that it may make this kind of observation with large uncertainty, 2 means it can make this observation with reasonable confidence, and 2+ means this sensor is good at make this kind of observation.

Of course, in our highly dynamic environment, a sensor’s capability to make some specific measurement also heavily depends on the configurations and conditions. For example, for using camera to identify a pre-registered user, the reliability of observation that this is user A will heavily depend on the user’s pose, the lighting condition etc. factors.

Here, our sensor fusion framework assumes that the sensor widgets will somehow generate a reasonable confidence estimation associate with each observation it reports to the system.
This picture shows the configuration of the planned demonstration system. To realize the idea of generalize able sensor fusion architectural support, there are many challenges ahead, one of which is the sensor fusion trigger and process synchronization problem.

With the sensors highly distributed, and sensors’ working status different in nature, for example, a video camera may continuously provide output series, a microphone may generate a burst of output sequences, and a fingerprint reader can only originate discrete output events, it is very important that we specify a cooperation scheme. This cooperating scheme should allow the sensors to fulfill their lower-level sensor fusion with their own multiple measurements within some time interval, synchronized by, and maybe adopting some suggestions about the time interval from, the relevant system sensor fusion mediators.

As described before (slide 21), my proposed approach is: for each type of context, there is a corresponding sensor fusion mediator. This sensor fusion mediator will specify some sensors to act as active triggers, and generally, based upon the trigger signals coming from the active sensors, it will decide when to flip a flag to indicate that the sensor fusion and context updating process is going on, with this the mediator will then query all relevant sensors to have them report their observations after a pre-specified time interval of delay.

In designing sensor widget interfaces and sensor fusion mediators, all sensor widgets will be designed as capable of keeping some length of their history data which maintains the lower-level sensor fusion result --- sensor fusion with its own history data. The sensors that can only generate discrete events such as fingerprint readers should usually be set as active triggers to proceed a new round of sensor fusion and context information updating process.
The other application scenario is to demonstrate the function of the left part of the picture shown before (Slide 21), illustrating that higher-level context information can facilitate lower-level sensor fusion.

This is for the sensor fusion research project in Robert Bosch Corporation Research Center North America. The long term goal is to detect, track and classify objects (pedestrians, curbs, lanes, other vehicles, etc.) around a sensors-instrumented car to enable the system to have awareness of the world within 360 degree and 7 meter coverage.

The system may include sensors of short range radar, far range radar, Lidar, laser scanner, laser line stripener, 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 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 object matching processes and to improve sensor fusion algorithm’s reliability, for example to use the vehicle speed information and local speed limit information to detect and correct erroneous object number assignments; the other way is to use higher-level context information to choose lower-level 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.
Secondary Demo System: Context and Jay Gowdy’s Sensor Fusion Architecture

Sensor Fusion Process
1. Sensors produce observations
2. Observations fed to “Hypothesis Pool”
3. Hypotheses strengthen with new data, weaken in absence of data
4. Unused observations generate new hypotheses

Context Information
- Hypotheses from sensor fusion (sensed objects)
- Vehicle’s status (pose, speed, etc.)
- Environment (downtown vs. highway, road conditions, etc.)
- Other, such as weather, time, driver’s physiology status, etc.

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, and the required being sensed context information is better defined with clearer constraints.

The project adopts Jay Gowdy’s so-called “Symbolic Sensor Fusion” framework. How the architecture works was summarized as four steps: 1) sensor modules produce observations; 2) observations are fed into a hypotheses pool; 3) hypotheses in the pool will strengthen with new data and decay with time; and 4) unused observations will generate new hypotheses.

This architecture actually should be categorized into the “blackboard” context sensing category as we discussed before. From sensor fusion point of view, this architecture is suitable to provide competitive type of sensor fusion support also as in our primary demonstration system.

Following our proposed methodology, the context information architecture is simpler also, here is a list of some context information we can think of.

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. So our context-aware sensor fusion will attempt to help in these two aspects.
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 at each point of a time sequence, an attempt was made to assign a number to each detected object (i.e., a point in the observation pool), in order to track objects and facilitate the object recognition tasks. Therefore, to fuse measurements from different sensors, the first necessary step is to correlate all those detected points at approximately same time point.

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

As Jay Gowdy’s sensor fusion framework doesn’t specify how sensor observation data should actually correlate existing hypotheses in the hypotheses pool. The first stage of implementation is to use the vehicle’s own velocity and local speed limit information to solve the problem. By using these context information, we will be able to calculate constraints regarding moving objects dynamically thus enhance object tracking. In addition, the current research joint effort 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.
Here is my working plan.

This work began since my last year’s summer job at Motorola Research Lab in Schaumburg. Till now, the pre-thesis proposal work as of literature search and system preparation have made reasonable progress. The proposed thesis research is planned to finish in another about one and a half years.

One thing worth mentioning is that IEEE 1451 standards also advocate the “plug and play” concept in dynamic sensor configuration systems, but there is not any IEEE 1451 standard-compatible products for us to use at current stage. To implement software simulation of IEEE 1451 standard sensors, and then to compare and evaluate it against the Context Toolkits’ “Sensor Widgets” concept, is an important part of the proposed research agenda.
After having carefully examined current sensing and sensor technology status for context-aware computing, the proposed thesis research challenges to explore context-sensing improvement methodology and its required system architectural support. The proposed top-down sensor fusion approach for context-aware computing can be summarized in two major steps: to define a context information architecture that can model the real world in the given simplified situation; and to map sensory data into the context model with the architectural support from a dynamic context database system.

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 --- if we have a proper software system architecture support.
So from the above discussion, it is quite clear that of the three sensor fusion categories, the proposed architecture supports competitive sensor fusion.

• 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.

As each specific sensor’s performance improvement will be a Ph.D 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: 1) a demonstration of working system with the above specified elements; 2) a software package with corresponding system description documents; 3) 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.
**Expected Contributions**

- Demonstrate 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.
- Propose and demonstrate the idea of supporting *generalize-able* sensor fusion processes in translating sensory data to semantic representations in an information structure.
- Improve the system performance and adaptability of the quite well acclaimed Georgia Tech Context Toolkit systems.

---

*Wednesday, July 11, 2001*

*Huadong Wu, 33*