2015-06

A framework for context-aware sensor fusion

Martí Muñoz, Enrique David

Universidad Carlos III de Madrid
Escuela Politécnica Superior

Departamento de Informática
Doctorado en Ciencia y Tecnología Informática

Tesis Doctoral

A Framework for Context-Aware Sensor Fusion

Enrique David Martí Muñoz

Dirigida por

Jesús García Herrero

José Manuel Molina López

19 de junio de 2015

E-mail: emarti@inf.uc3m.es

Telephone: +34 91 856 1320

Address:

Grupo de Inteligencia Artificial Aplicada
Departamento de Informática
Universidad Carlos III de Madrid
Av. de la Universidad Carlos III, 22
Colmenarejo 28270 — Spain

A Framework for Context-Aware Sensor Fusion

Autor: Enrique David Martí Muñoz
Directores: Jesús García Herrero
José Manuel Molina López

Firma del Tribunal Calificador:

	Nombre y Apellidos	Firma
Presidente: D.	.....	.....
Vocal: D.	.....	.....
Secretario: D.	.....	.....

Calificación: .....

Colmenarejo, ... de ..... de 2015.

A los que siempre me acompañáis,
aun estando muy lejos.

Contents

List of Figures

List of Tables

Abstract

SENSOR fusion is a mature but very active research field, included in the more general discipline of information fusion. It studies how to combine data coming from different sensors, in such way that the resulting information is better in some sense –more complete, accurate or stable– than any of the original sources used individually. Context is defined as everything that constraints or affects the process of solving a problem, without being part of the problem or the solution itself. Over the last years, the scientific community has shown a remarkable interest in the potential of exploiting this context information for building smarter systems that can make a better use of the available information.

Traditional sensor fusion systems are based in fixed processing schemes over a predefined set of sensors, where both the employed algorithms and domain are assumed to remain unchanged over time. Nowadays, affordable mobile and embedded systems have a high sensory, computational and communication capabilities, making them a perfect base for building sensor fusion applications. This fact represents an opportunity to explore fusion system that are bigger and more complex, but pose the challenge of offering optimal performance under changing and unexpected circumstances.

This thesis proposes a framework supporting the creation of sensor fusion systems with self-adaptive capabilities, where context information plays a crucial role. These two aspects have never been integrated in a common approach for solving the sensor fusion problem before. The proposal includes a preliminary theoretical analysis of both problem aspects, the design of a generic architecture capable for hosting any type of centralized sensor fusion application, and a description of the process to be followed for applying the architecture in order to solve a sensor fusion problem.

The experimental section shows how to apply this thesis' proposal, step by step, for creating a context-aware sensor fusion system with self-adaptive capabilities. This process is illustrated for two different domains: a maritime/coastal surveillance application, and ground vehicle navigation in urban environment. Obtained results demonstrate the viability and validity of the implemented prototypes, as well as the benefit of including context information to enhance sensor fusion processes.

Resumen

La fusión de sensores es un campo de investigación maduro pero no por ello menos activo, que se engloba dentro de la disciplina más amplia de la fusión de información. Su papel consiste en mezclar información de dispositivos sensores para proporcionar un resultado que mejora en algún aspecto –completitud, precisión, estabilidad– al que se puede obtener de las diversas fuentes por separado. Definimos contexto como todo aquello que restringe o afecta el proceso de resolución de un problema, sin ser parte del problema o de su solución. En los últimos años, la comunidad científica ha demostrado un gran interés en el potencial que ofrece el contexto para construir sistemas más inteligentes, capaces de hacer un mejor uso de la información disponible.

Por otro lado, el desarrollo de sistemas de fusión de sensores ha respondido tradicionalmente a esquemas de procesado poco flexibles sobre un conjunto prefijado de sensores, donde los algoritmos y el dominio de problema permanecen inalterados con el paso del tiempo. En la actualidad, el abaratamiento de dispositivos móviles y embebidos con gran capacidad sensorial, de comunicación y de procesado plantea nuevas oportunidades. La comunidad científica comienza a explorar la creación de sistemas con mayor grado de complejidad y autonomía, que sean capaces de adaptarse a circunstancias inesperadas y ofrecer un rendimiento óptimo en cada caso.

En esta tesis se propone un framework que permite crear sistemas de fusión de sensores con capacidad de auto-adaptación, donde la información contextual juega un papel fundamental. Hasta la fecha, ambos aspectos no han sido integrados en un enfoque conjunto. La propuesta incluye un análisis teórico de ambos aspectos del problema, el diseño de una arquitectura genérica capaz de dar cabida a cualquier aplicación de fusión de sensores centralizada, y la descripción del proceso a seguir para aplicar dicha arquitectura a cualquier problema de fusión de sensores.

En la sección experimental se demuestra cómo aplicar nuestra propuesta, paso por paso, para crear un sistema de fusión de sensores adaptable y sensible al contexto. Este proceso de diseño se ilustra sobre dos problemas pertenecientes a dominios tan distintos como la vigilancia costera y la navegación de vehículos en entornos urbanos. El análisis de resultados incluye experimentos concretos que demuestran la validez de los prototipos implementados, así como el beneficio de usar información contextual para mejorar los procesos de fusión de sensores.

Agradecimientos

ESTO va a ser largo, pero os vais a tener que aguantar. Cuando uno entra en una vida tan plenamente como lo habéis hecho vosotros en la mía, debe atenerse a las consecuencias.

Quiero abrir los agradecimientos con unas palabras para Jesús, mi director, por enseñarme a ser mejor en mi trabajo a través de su ejemplo impecable, por su atención al detalle que es la base de la auténtica excelencia. Por ser un pozo de paciencia frente a todos esos desórdenes, dudas, retrasos y desastres que le he ido poniendo delante de los pies, y meterme en el camino cuando me empezaba a desviar. Gracias a José Manuel, por su clarividencia y resolución para saber en medio minuto qué está mal, por qué, y cómo arreglarlo. No sólo me has ayudado en la realización de la tesis sino que me has enseñado a pensar de forma distinta y, creo, mejor. Gracias a los dos por ser de esos "jefes" que sabes que van a hacer lo que esté en su mano (y a veces también lo que no lo está) por cuidar de quienes tienen a su cargo.

Eterno agradecimiento a David "GPS master". Siempre recordaré los paseos en coche, con tu talante inmejorable y todo lleno de cables: has sido la mejor ayuda posible. También a Nando, por su colaboración en la recogida de datos. Esto habrá que celebrarlo: no todo van a ser conciertos de U2 en Chicago, pero algún bicicleteo podrá caer, digo yo. Gracias al resto del GIAA y habitantes del campus: Miguel Ángel (que me metió en este embolado), Antonio, Mar, Eli, Carmela, Iván... perdonad que no siga, ¡sois tantos! Os voy a echar de menos.

Un hueco especial para mi "familia adoptiva", mi hermano Javi, mi mamá María Elena y mi papá Miguel. Habéis logrado que vuestro hogar sea también el mío, y eso es mucho decir. Puede que sin vosotros también estuviese donde estoy, pero sin duda alguna sería mucho menos feliz. Javi, es imposible agradecer lo que hemos aprendido, reído y caminado juntos. Quizá la mejor manera sea seguir haciéndolo durante unos cuantos años más, pongamos 50 o 60.

Rebeca, contigo sobran las palabras, de modo que ni lo voy a intentar. Sólo *gracias*. Tú sigue saltando de sueño en ilusión y pintando vidas con tu magia. Sabes que siempre me tendrás a tu lado, por muchos kilómetros y meses que separen nuestros caminos. Esto se extiende a mi gente de Tramacastilla. A Sara, porque hemos compartido tanto que asomarme a sus ojos es como mirar al espejo... ¡y lo que nos queda!, a Fer, Ale, Tamy, Pedro, Marcos, Vicente, Sofi y todos los demás. A pesar de todo, hay rincones fuera del tiempo y el espacio en los que reímos todos juntos.

A mis compañeros de tatami y de vida. Javi, Isa (¡"sobri" Luna!), Dani y Miri, vosotros sois especiales a muchos niveles, y siempre constituiréis un motivo para regresar donde quiera

que os halléis. A Kp (¡y Olga y Luna!), sensei, maestro, amigo, sin todos esos años a tu lado no sería quien soy... ¡Os!

A la Asociación del Crimen Organizado (así, *sin nombres*): los delitos pueden prescribir para la Justicia, pero seguirán frescos en mi memoria. Gracias a vosotros el trabajo ha sido menos trabajo, y lo que no es trabajo... me ha dejado ratos de los que te llevas en el último paseo en barca. Espero seguir disfrutando de vosotros, por separado y en pelotón.

Elena, Raúl, Isa, Marta, Iván... nos hemos conocido de adultos, pero en el fondo siempre seremos una panda de niños jugando en la tierra. Quiero tener cerca vuestras conversaciones, consejos y abrazos, siempre. Emi, tú vas aparte porque contigo también me faltan (o me sobran) palabras. No sé si eres consciente de lo importante que ha sido tu apoyo durante estos años, de modo que quiero dejar aquí registrado que el 49 % de esta tesis lleva tu marca. Siento no llegar al 50, pero es que tiene que seguir siendo mía.

A los más importantes, mi familia. Papá, mamá: todavía me maravillo por la fortuna que supone ser hijo vuestro. Gracias por vuestros amor, esfuerzos y desvelos, por darme lo necesario para llegar hasta aquí, por haberme hecho quien soy. Alex, hermano: te tolero.
(¡es broma! no concibo la vida sin saber que estás ahí, tu ayuda incondicional, sentido del humor... *aunque tampoco te tolere tanto, no te vengas arriba*).
Tíos, primas, abuelas, gracias también a vosotros. Dicen que la familia no se elige, pero creo que a mí me dejaron hacer trampas.

Y para finalizar, a mis dos abuelos, donde quiera que estén. Especialmente a mi yayo Arsenio, que sé que habría dado cualquier cosa por ver este momento. Esta va por ti.

Os quiero.

Kike
Colmenarejo, junio de 2015.

1

Introduction

The main goal of this dissertation, as stated in the title, is to define a Framework supporting and guiding the construction of context-aware sensor fusion systems. This is, sensor fusion systems that can integrate information describing its context, with two potential benefits: (a) make the fusion products better in some sense (b) increase robustness through adaptation to changing circumstances.

Along this section we will introduce some basic concepts required to understand this work. Next, these concepts are put together so that we can identify the needs and problems motivating this dissertation. The chapter is closed with the goals set for our work.

1.1 Basic concepts

Two are the basic building blocks of this work: sensor fusion and context information. This section introduces them briefly, along with a discussion on how they are related.

1.1.1 Sensor fusion

Sensor fusion is a subdiscipline of Data Fusion (also known as Information Fusion). Let us explain in detail what is Data Fusion before entering on details about the sensor part. We will use the definition found in the preface of (Liggins et al., 2008):

> Multi-sensor data fusion seeks to combine information from multiple sources (including sensors, human reports, and data from the Internet) to achieve inferences that cannot be obtained from a single sensor or source, or whose quality exceeds that of an inference drawn from any single source.

Lacking an official definition, though, many authors have opted to include their own formulations on their works. The most relevant ones are compiled and analyzed in (Boström et al., 2007).

Data fusion discipline aims to obtain the best description of a situation, from information that can be partial, insufficient, imprecise or unreliable. With that purpose, it borrows techniques from numerous fields of computation from the most simple mathematical operations, to challenging computational problems as decision taking, planning or automatic knowledge extraction from text.

This work is limited to sensor fusion, a branch of data fusion concerning data acquired exclusively from sensors, as opposed to data generated by human beings (also called "soft sensors") or extracted from databases as the Internet —semantic analysis of text, vague or abstract descriptions of complex situations—. However, since we will include context data, we can refine the previous statement by limiting this thesis to "low level information fusion" as defined in (Waltz and Llinas, 1990): information related with individual entities, as opposed to the "high level information fusion" that has to deal with the relation between entities, its meaning and consequences.

Sensor data is directly related with the physical magnitudes of real world entities, such as speed, weight, location, color or luminosity, which is characterized for having a lower abstraction level. It is usually presented as numeric vectors or discrete categories that makes its interpretation easier, although subject to different problems: noise, incompleteness, being partial and represent huge volumes requiring special processing techniques.

In data fusion systems with multiple stages, sensor fusion occupies the first ones because of its relation with the real world. It prepares raw data to be used by fusion processes with a higher abstraction level.

1.1.2 Context

The Royal Academy of Spanish Language defines context as *the physical or situational environment in which a fact is considered*. However, the definition of this term is usually adapted to fit the specific problems of the field in which it is considered: context has to be defined in a particular context.

The survey (Bazire and Brézillon, 2005) gathers and analyzes over 150 definitions for "context" in different disciplines including sociology, psychology and computation. Authors searched for the common terms and concepts, reaching the following general description: *context is the set of restrictions affecting the behavior of a system embedded in a certain task*. For this work, we have selected a definition that suits better the problem of sensor fusion:

> Context is everything that constraints or affects the process of solving a problem, without being part of the problem or the solution itself.

This is, the context of a system is composed by a set of internal or external factors that are difficult to anticipate and model, so that they cannot be easily integrated in a traditional algorithmic solution to the problem. In some circumstances, nonetheless, the context has a significant impact in the behavior of the system can be important enough to invalidate solutions not taking it into account.

Context-aware computing represents the best example of the application of contextual information to computing systems, where an automated system try to infer the situational information (context) of human users, in order to provide them with relevant services without requiring explicit actions on their side. Context aware computing is a mature discipline with solid, well-founded theory on the modeling, representation, processing and use of context. Thus, in spite that the motivation of this dissertation differs from that of context-aware systems, we consider it a helpful starting point for developing some of the foundational concepts of our dissertation.

1.1.3 Relation between fusion and context

Information fusion systems are defined and operated in open environments, and are consequently immerse in a context that can have a great influence on them, as explained before. Detecting this context and reacting to changes on it is a key capability for fusion systems that can deliver consistent performance in the real world. According to (Steinberg and Rogova, 2008), context can be used in Information Fusion to:

- Refine ambiguous estimates
- Explain observations
- Constrain processing, either cueing/tipping-off or in managing fusion or management processes

In the last two decades, a great number of works have been published around the inclusion of context information in data fusion algorithms. Thanks to that knowledge, results are better (more precise, fast and/or adequate) and fusion systems have improved their robustness and flexibility because they can adapt themselves to the environment. It is possible to find groups of interest focused on handling context information to improve high level fusion processes. As an example, the 8th edition of the International Conference of Information Fusion (2007) hosted an "Special Session on Context Information in Data Fusion".

There is not a clear consensus, however, when dealing with sensor fusions systems and its lower level of abstraction. It is possible to find works where context is applied to sensor fusion processes, but the proposed solutions are valid only for very specific problems and they make a limited use of the context.

In the past, synergies between fusion and context have been used in the opposite direction: context-aware computing systems use sensor fusion techniques for extracting contextual data with higher accuracy and reliability, as argued in Huadong Wu dissertation (Wu, 2003). This same work postulates that the loop can be closed for further benefits: use inferred context information to improve the sensor fusion process, which is the proposal developed by this thesis.

As a side note, the distinction between what is a variable of the problem and what is context becomes fuzzier as systems grow in size, complexity and abstraction level. For example, meteorological information is clearly a contextual factor for a fusion process that tries to track the motion of a vehicle based on some sensors. It could indirectly constraint the mobility of the vehicle, but the basic sensor fusion processes are completely independent and still valid. However, weather forecast can be a fundamental piece of information for a high level fusion process that is considering the probability of a foreign army attacking a facility, while still being information about what "surrounds" the problem itself.

1.2 Motivation

The paper (Snidaro et al., 2013) reviews the most important elements on the way to creating adaptive fusion systems that made an extensive use of context information. It identifies two needs to be satisfied by the scientific community:

- Developing a methodology for the inclusion and exploitation of context information in fusion systems.
- Design a context-aware fusion architecture that integrates context knowledge for a better understanding of the observed entities and to adapt itself to changes in the environment.

Our thesis is that Sensor Fusion systems, even those dealing only with low level information, can benefit from the use of context information for two purposes: automatic adaptation to the environment and enhance the quality of the resulting fusion products. The efforts of the scientific community on this field have been sparse, focused on small and concrete problems, and insufficient until the present. The proposed solutions are so specific that they can sometimes lose their validity if one of the sensors is substituted with a different model with slightly different technical specifications, or under subtle changes in the environment.

Sensor fusion discipline needs a solid theoretical proposal for the application of context information. This proposal must account for all the involved factors: what can be considered

context information, how should it be acquired, represented and stored, how to use it for improving fusion processes, and the problems or difficulties derived from such use.

A second point of interest comes with a shifting/expanding market for data fusion systems. From its beginnings over 40 years ago, data fusion has been mostly applied in military environments. Over the last few years, the advent of smartphones as portable computing platforms with extensive sensing capabilities represents a whole new world of possibilities. This short period of time has been sufficient to develop thousands of applications for tracking physical activity, augmented reality or monitoring user health.

Developing this kind of applications in a mobile device poses several challenges. In first place, smartphones are equipped with sensors that have been designed to be small, power efficient and cheap. Such sensors provide low quality data (compared with higher end sensors). On the other hand, mobility means that some sensors can be subject to periods of degraded quality and outages: GPS does not work underground, and light sensors can be useless when the device is in a bag.

Sensor fusion can be used to improve poor quality data, and context information is useful to detect abnormal sensing conditions. Combining both disciplines can lead to adaptive fusion schemes that react against unexpected problems and adapt their strategy in order to provide the best data under a wide range of conditions.

1.3 Goals

Taking as starting point the needs identified in the motivation, we have determined the following list of goals for this work:

1. Analyze the different ways in which context information can be applied to sensor fusion processes. Establish a clear and solid theoretical proposal, supported by experiments.

2. Develop a framework that supports the creation of adaptive, context-aware sensor fusion systems. It will be composed of both theoretical tools (models and guides) and software modules with generic tools useful for developing this kind of systems.

3. Implement functional prototypes which demonstrate the validity of our proposal. This includes:

    - Select applications of interest. Analyze the related state of the art.
    - Study the set of sensors to be used. Implement data gathering processes.
    - Design one or several algorithmic proposals that can use sensor data for solving the problem.

- Identify context information useful for improving the fusion process. Integrate that information into the solution.
- Analyze the strengths of the context-aware solution with respect to a traditional sensor fusion scheme.

In order to satisfy these goals, it is necessary to make an in-depth study about architectures and models used in information fusion and context management. The acquired knowledge will serve as basis for our proposal, which should combine aspects of the two disciplines.

The basic building blocks of a sensor fusion system are the low level techniques in charge of doing the actual fusion: the fusion algorithms. The framework must define a proper interface (input, output) for optimal information flow. Also, fusion algorithms determine how context information can be used, a factor that will affect the design.

1.4 Structure of the document

Chapter 2 introduces the two principal topics of this dissertation, sensor fusion and context information, covering their most important concepts and conducting a review of the relevant state-of-the-art. We are interested in the architectural and design aspects of existing information fusion systems that can help building a framework with the characteristics enumerated before. The second part of the chapter reviews the use of context in computer sciences, following a similar structure. The focus here is put in the management of context information: architectures and mechanisms used to acquire, store and disseminate contextual data. We will also analyze the problems related with context information, namely the uncertainty, the relevance, its accuracy over time and other questions as its pedigree (origin of the data).

Chapter 3 describes our proposal: the design of a framework for context-aware sensor fusion. Chapters 4 and 5 Another chapter describes the different cases of study used for developing and testing the framework. The first one takes an already implemented fusion system, and explains how the proposal can be used to enhance the solution by including context information, and also to improve the flexibility and robustness of its design. The second scenario designs and implements an adaptive context-aware sensor fusion solution that is then tested and analyzed.

The last part of the thesis reviews the list of goals, checks that they have been satisfied and proved through the proposed experiments, and provides some concluding remarks about the advantages of including context information in the design of sensor fusion systems.

2

State of the art

This chapter contains a review of the relevant literature in the fields of data and sensor fusion and context-aware computing, including previous efforts to bind them in a meaningful way.

2.1 Sensor Fusion

In this thesis we are interested in two different aspects of sensor fusion: high level organizational aspects and low level techniques. Organizational aspects include conceptual models, architectures and frameworks that can be used to define and create sensor fusion systems. By analyzing the related literature we expect to find ideas and examples that can guide us on developing our own framework. We should also identify limitations in those approaches, primarily related with the incorporation of contextual knowledge in the fusion processes.

Our interest in low level algorithms is purely instrumental: we need to understand how information is used and transformed in order to create an effective tool. Furthermore, the implemented prototypes require a bottom layer in charge of performing the sensor fusion task.

2.1.1 Models, architectures and frameworks

2.1.1.1 Definitions

Let us define and clarify the differences among these three terms before starting the analysis of existing literature. This work follow the proposal of (Llinas, 2010) with minor modifications:

Model

A conceptual tool that provides a way to understand or approach a problem. Models are composed by definitions of the elements involved in the problem and its solution, descriptions or rules guiding the relationships between those elements, and criteria for categorizing relevant knowledge.

Models are useful because they provide a common framework for reasoning and exchanging information about the problem. It does not facilitate mechanisms for solving concrete problems. e.g. Joint Directors of Laboratories (JDL) Fusion model organize data and processes based on their abstraction level, describing their role in a general data fusion application created to support human beings in decision taking processes.

Architecture

Defines a (hardware and/or software) structure that is useful for creating solutions to a problem. An architecture can be considered as a model (previous definition) whose goal is to describe the parts composing the solution, and how those parts are organized and related.

Architectures can describe different organizational aspects, including physical distribution (centralized, distributed), physical/process topology (ring, hierarchical) or process organization (layered, Model-View-Controller).

Framework

Conceptual structure that can be used as support or guide in the construction of a system with a real application. It is defined in (Llinas, 2010) as "partial realization of an architecture, that can be used as the base for building concrete solutions". In spite of not constituting a solution to concrete problems, it differs from models and architectures in that frameworks include tools and generic functionality.

In the domain of software, a framework usually implies what is known as "control inversion": solutions are implemented by creating pieces of software that are plugged in the structure of the framework, but it is the framework itself who guides the processing. Control inversion is actually quite variable: it depends on the purpose of the framework and how specific it is –bound to a restricted type of problems.

Library

A collection of software tools and utilities, useful for solving a certain type of problems. A library is different from a framework in that its components do not imply any processing structure or stream. They are individual, independent tools.

2.1.1.2 Relevant works

This section presents models, architectures and frameworks together despite the differences among them, because it is difficult to classify some of the existing works in a single category. Moreover, most of the existing literature does not make such distinction and uses the three terms equally, so that we prefer to follow the trend to avoid confusions.

As stated in section 2.1.1.1, some models are strongly focused in concrete aspects of the problem to be solved, so that they complement each other and makes direct comparison more

difficult. It is important to keep this fact in mind, since our design will be driven by the use of context information but we shall keep other aspects in mind.

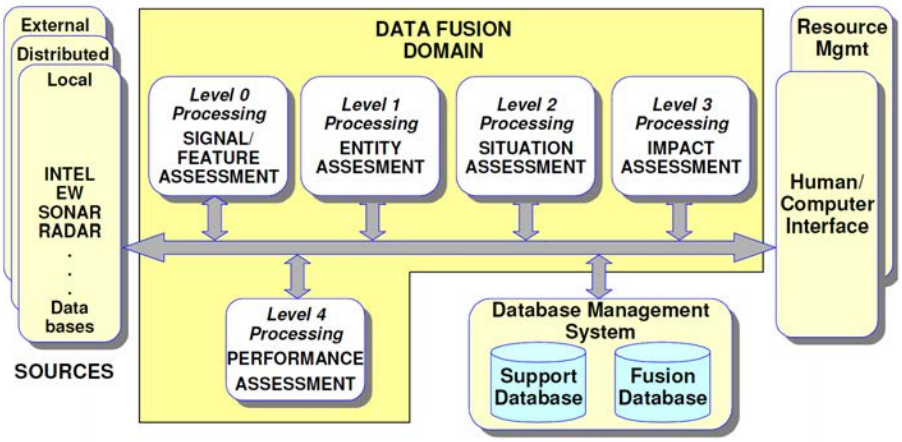
JDL Model The JDL Model (Llinas et al., 2000) was presented at the beginning of the 90s as a conceptual framework for developing data fusion systems. It is composed by layers representing processes of increasing abstraction level. This model aims to provide a common language for unambiguous communication between groups of interest. This is remarked in (Liggins et al., 2008), p.26, since the "layer" terminology can mislead the reader:

> The diagrammatic arrangement of the levels and the nomenclature used were not intended to imply a hierarchy, a linearity of processes, or even a process flow. [...] parallel processing of these subprocesses and level skipping can and does take place. The model is still useful, for the levels serve as a convenient categorization of data fusion functions and it does provide a framework for understanding technological capability, the role fusion plays, and the stages of support for human decision making

After its last review, the model defines the following levels:

- Level 0: Sub-object assessment. Raw signal process for getting refined sensor information. e.g. translate pixels to blobs in an image/video signal.
- Level 1: Object assessment. Use sensor information for extracting data about individual entities. e.g. identity, position, speed.
- Level 2: Situation assessment. Analysis of relationships between individual entities. e.g. clustering techniques for discerning groups.
- Level 3: Threat assessment. Project current situation into future, to identify threats, vulnerabilities and opportunities.
- Level 4: Process refinement. A metaprocess that monitors the whole system and optimizes its response.
- Level 5: Cognitive refinement. Interaction between fusion systems and human users. e.g. data visualization, decision support tools.

Although the model does not imply hierarchy or process flow, it is quite common to find data fusion applications following the order defined here. As a reference, sensor fusion implies levels 0 and 1 of JDL model.



Waterfall The waterfall process model (Markin et al., 1997) is derived from the software engineering methodology with the same name. It proposes a chain of processes, where each action is based in the products of its ancestor and will forward its results to the next action.

It is divided in 6 layers: the two first can be identified with level 0 in JDL model, next two with level 1. The last two layers are equivalent to levels 2 and 3 respectively. This model represent the most natural and direct approach for doing data fusion. Its main problem is that, because of its strict sequentiality, lacks feedback. Our proposal will be far from this model, since feedback is a powerful tool for building adaptive systems.

Intelligence Cycle Defined in (Shulsky and Schmitt, 2002), makes emphasis in the general steps common to all data fusion processes. It is composed by 5 steps repeated in cycle, closing a loop that makes feedback possible. These steps are:

1. Planning and direction: determines "information requirements" of the process.

2. Collection: acquisition of the required information.

3. Collation: align collected information. This step involves data transformation, temporal scale resizing, etc.

4. Evaluation: the proper data fusion processes, where information is analyzed and transformed. The result of this analysis is new data with a higher abstraction level (usually called "intelligence"), that can be directly applied for solving problems.

5. Dissemination: the created intelligence is distributed over the system, to be used in later cycles.

Boyd Control Loop (OODA) The origin of this control loop is a set of slides used by Boyd during an informal presentation in 1959. It is defined from a classical decision support scheme (OODA) used in military information systems (Angerman, 2004), but the affinity of those systems with data fusion makes it possible to extend its application to the latter.

The loop is composed by four steps:

- Observe: Data acquisition and pre-processing (level 0 JDL)
- Orient: Actual fusion processes(levels 1-3)
- Decide: Comparable to JDL level 4.
- Act: this part is out of the scope of JDL model. It is referred to actions over the real world, performed by entities that are not part of the data fusion system.

It is important to notice that this model is not focused in the data, but in the process. It is helpful to put data fusion systems in context, to identify the purpose they serve to.

LAAS LAAS is the acronym for "Laboratoire d'Analyse et d'Architecture des Systèmes", it was developed as an integrated architecture for design and implementation of mobile robots operating in real time conditions. It is intended to improve the amount of code that can be reused when developing this kind of software. It is included in many sensor/data fusion surveys because robotic systems do extensive use of fusion techniques.

Low- and mid-level sensor fusion processes (JDL 0-2) are located in the functional level on this proposal, while high level processes (JDL 3) take place in the decision level. LAAS defines a design guide for robotic systems, but it is considered an architecture and not a model because it includes utilities (reusable software modules) for the implementation.

It defines a layered architecture where information flows strictly between adjacent levels. This is too rigid for our purposes of feedback and context information dissemination, but we will take into account this reference for its ideas regarding code reusing.

Dasarathy Model The model proposed in (Dasarathy, 1997) approaches the creation of fusion systems from the perspective of data transformation. Information is classified into three abstraction levels. Authors state that this classification is repeated in many independent works over the years. The levels are:

- Data: raw information, as provided by sensors
- Features: intermediate level information

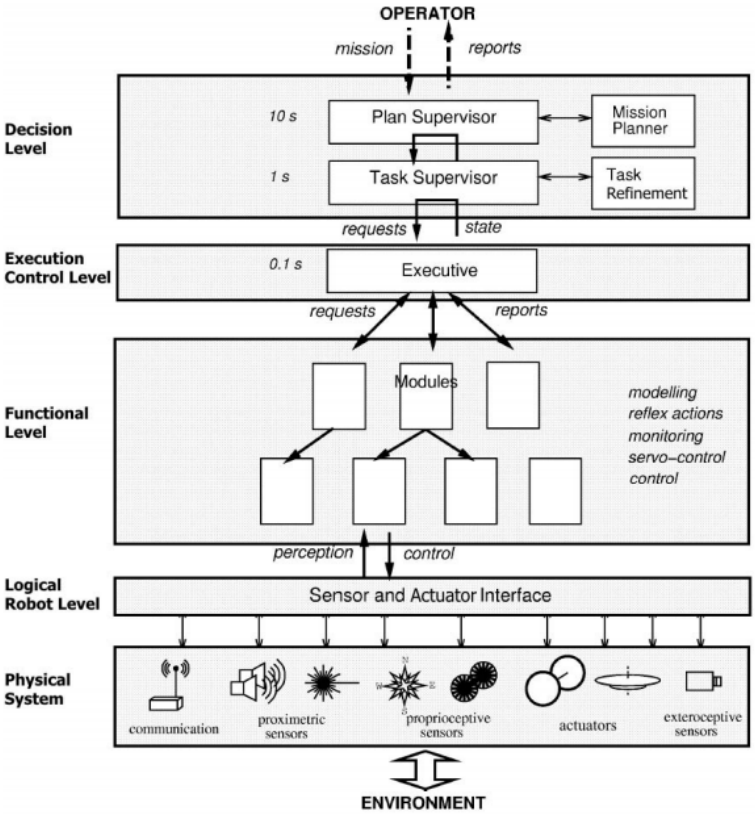


Figure 2.1: LAAS architecture for design and implementation of real time mobile robots

- Decisions: symbols, statements, beliefs.

Dasarathy points that fusion processes act both as a bridge that transform information between levels, and for creating new information with the same level of abstraction. This results in five different categories: 3 intra-level and 2 inter-level:

- DAI-DAO (Data Input, Data Output)
- DAI-FEO (Data Input, Feature Output)
- FEI-FEO (Feature Input, Feature Output)
- FEI-DEO (Feature Input, Decision Output)
- DEI-DEO (Data Input, Decision Output)

The original work applies this categorization in the design of several processing architectures with different features.

Omnibus Presented in 1999 (Bedworth and O'Brien, 2000) as an effort for creating a data fusion model with all the desirable features of previous proposals, getting rid of their problems.

It starts with a cyclic process inspired by Intelligence Cycle and Boyd control loop. Each level is refined by adding the definitions of Waterfall model, conveniently restructured so that each step can be identified with a level of JDL and Dasarathy models.

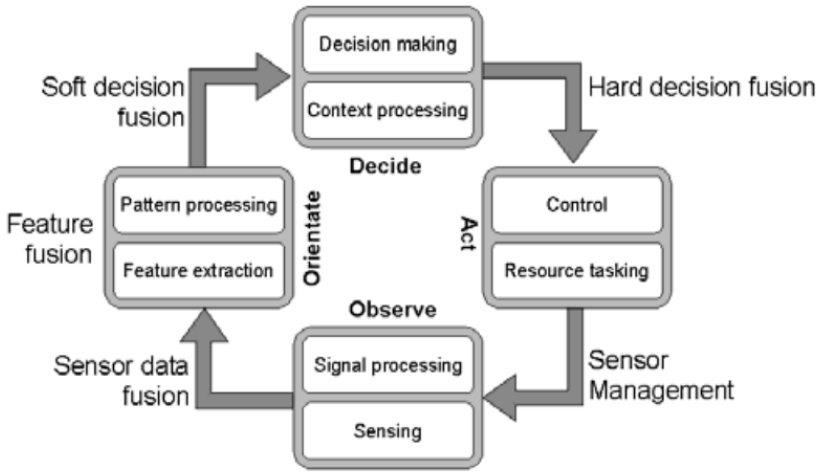


Figure 2.2: Omnibus Model

In words of the authors, feedback is explicit in this model, and what is more important: it tackles the problem of nested fusion cycles, where a step of the process can, internally, be another full fusion cycle. The result has a higher detail compared with previous works, since it tries to define the logical steps in the fusion process, how those steps are related, and the abstraction levels of the information.

The idea of nested cycles allows complex flows of information, in spite that feedback is limited to injecting the results of the last step into the first one of the next cycle. The way in which information flows inside the fusion process is a key aspect of our proposal. Because of this, Omnibus model constitutes a fundamental reference of this work.

DFuse DFuse is a framework for developing distributed sensor fusion systems (Kumar et al., 2003). It has been designed to work in heterogeneous wireless networks, assuming a fixed network topology and full time sensor availability.

The network is composed by source nodes, fusion points and sink nodes. When a fusion point does not get input data from a source on time, it can work over a partial subset to give an approximate result.

This framework includes two components:

- Fusion module: an API of fusion algorithms/procedures.
- Placement module: defines the location of the required functionalities (nodes and fusion points) inside the physical network (hardware elements).

DFuse facilitates the last part through a tool for automatic deployment of fusion applications in physical networks. It works from a graph that defines the fusion system, and the code for each functionality.

The principal drawback of DFuse is one of its requirements: the network must be static (not changing) and reliable. Other disadvantage is that the hardware in which the system is deployed shall be capable of timestamping messages in a consistent way. Last, the size of the software makes impossible its use in tiny devices as motes, although current technology is expected to make this inconvenient less important over time: the original work (Kumar et al., 2003) presents a case of study running on PDAs with at least 32MB RAM and a 200MHz ARM processor. This power is currently available in devices the size of a coin.

DFuse works in a different domain than our problem, both in the goals and in the set of base assumptions. However the idea of abstracting components as interrelated nodes that consume and/or produce information is very important for our proposal, to the point of constituting one of the cornerstones of the flexible architecture.

2.1.2 Fusion Systems with adaptation capabilities: JDL Level 4

Any fusion application with self-adaptive capabilities will include (by definition) processes belonging to Level 4 of the JDL model (Process Refinement). The article (Azimirad and Haddadnia, 2015) identifies the different aspects of process refinement, and cites some techniques used to implement them:

- Performance evaluation: measures of performance and utility theory.
- Process control: multi-objective optimization (through linear programming, goal programming or other methods).
- Source requirement determination: involves sensor modeling.
- Mission management: using knowledge-based reasoning techniques.

Level 4 is considered to close the loop of the fusion process. However, as stated by (Blasch and Plano, 2002), its functions usually do not involve actual information fusion processes but rather taking decisions from a set of indicators –decision intelligence (DECINT). In high level systems, this is translated into resource retasking to better support mission goals. Such

inference can be complex in multilevel information fusion systems, which is the reason why Level 4 is not present in many information fusion systems. These are the conclusions of (Liggins et al., 2008) (chapter 27, p. 692): it analyzes a total of 79 fusion systems, to find that only 9 provide Level 4 capabilities, although none of them was operational at the time of writing the book (just as part of R&D initiatives). Partial support for Level 4 is more common, understanding it as the extraction of performance measures that can be applied locally to some concrete processes of the fusion system, or interpreted by human operators to execute manual process refinement actions.

There are some theoretical frameworks for creating fusion systems with full-scale adaptive features. The article (Solano et al., 2012) defines the elements (primitives) of a fusion problem through semantic, temporal and geospatial features. These primitives are processed by a recombination workflow that maximizes the data exploitation value chain, in what is called a Recombinant Cognition System. The work presented by this thesis is, in some way, similar to previous proposal, in the sense that we will provide a formal definition of available primitives that enable automatic reasoning processes for enhancing the quality of the fusion.

Level 4 capabilities are easier to implement in sensor fusion applications. The gap between resources and output products is smaller considering the number of steps and the abstraction level. This statement is supported with arguments by our proposal (chapter 3) and demonstrated through examples (chapters 4 and 5).

2.1.3 Processing architectures in distributed systems

This work is proposed for centralized fusion systems, where all sensors and computing elements share location and are part of the same device. However, our design should be extensible to distributed systems, where the composing elements are in different locations and can also move around.

Compared with centralized schemes, distributed fusion systems bring new possibilities but its implementation pose several challenges regarding communication capabilities (availability, bandwidth), failure tolerance or adaptability to changes in the environment or in the goals to be accomplished.

The following list introduces generic distributed processing architectures that have been applied to data fusion (Benaskeur et al., 2007)(Hilal and Basir, 2014), sorted from more restrictive to more flexible:

Centralized

All fusion processes are performed in a unique central node. Input data is acquired by sensing nodes, that forward it (directly or through other nodes) to that central node. The strengths of centralized schemes include simplicity (easy to implement) and

minimum computational effort. Centralized processing makes possible an optimal use of the available information, and keep the total computational effort reduced to a minimum.

On the other hand, they are less tolerant to failures because of their dependence on the central node. Communication load is superior to other schemes, becoming a bottleneck in many cases.

Hierarchical

This architecture organizes nodes in a tree-like structure. Information flows from the leaves to the root. Intermediate nodes can process their input data to create intermediate results that are then transmitted to their ancestors.

Hierarchical architectures fit naturally in chain-of-command structures, and is thus frequently used in military environments. Root node gathers the final results and controls the whole process. Command controls follow the opposite direction, flowing from the root to the leaves.

The principal disadvantage of this scheme is its dependency on individual nodes: a blackout in an intermediate node means losing its subtree. Apart from that, communication between non-related nodes must pass through a common ancestor. If those are often required, the system can incur in a communication overload.

Holonic

Holonic architectures are halfway between hierarchical and heterarchical. They sacrifice the control capability of the first to get in exchange some of the flexibility and adaptability of heterarchies.

This architecture is composed of autonomous functional units called "holons", which cooperate to reach system goals. Each holon is composed by one or more nodes (processing or sensing), as well as other holons –in a recursive composition–. Ideally, a holon is a software agent with autonomous decision/operation capabilities.

Holonic architectures can be reconfigured online, giving them a great adaptability and failure tolerance. That flexibility comes at the cost of losing control over how the system works. Even tasks with a clear execution plan can be solved in unpredictable ways, because the result will depend on how holons negotiate and create coalitions.

Heterarchical

In heterarchical architectures, every pair of nodes can establish a direct communication link. They lack a predefined structure, and each node is fully responsible of its internal behavior and interaction with other elements.

This option is attractive for semi-collaborative complex systems where each node has independent goals, but can also collaborate with other elements to help them –or ask for assistance to fulfill its own. It is not adequate, however, for systems that have to

satisfy a global agenda: managing links between nodes in a big system can become a combinatorial problem, leading to chaotic behavior and a poor performance.

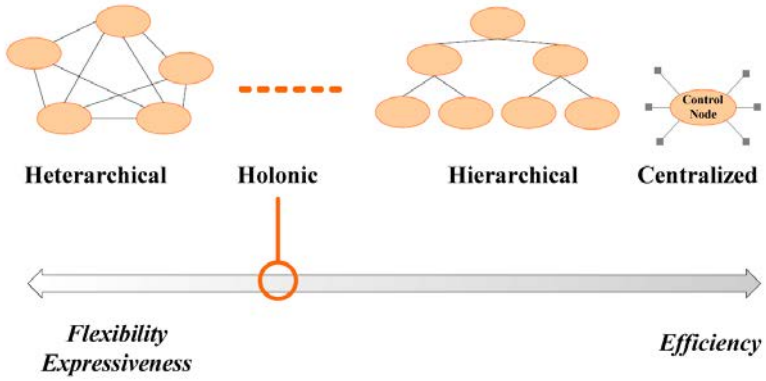


Figure 2.3: Processing architectures in distributed systems. Taken from (Benaskeur et al., 2007)

It is usually assumed, when a data fusion system is created, that the participating entities do contribute to satisfy a common goal agenda in a coordinate manner. This can be not true in some applications, where the entities are independent but can punctually collaborate.

Following this line, we find other "intermediate" proposals that have been extensively studied in the field of multi-agent software systems. We are talking about federated and market-based architectures.

A federated architecture can be described as a plain holonic organization, meaning that none of its holons contains another holon, where each holon contains an intermediary agent managing communications. This agent is responsible of both internal –between holon components– and external communications –with other holons. Intermediary agents help reducing communication complexity, at the cost of introducing a critical component that can represent a bottleneck. On the other hand, control is far more consistent than in holonic architecture in exchange for losing some of its scalability.

Market-based architecture adds management and organization features to the heterarchical architecture. Lets assume a scenario where coexisting independent entities require and offer services, resources or tasks, but the negotiation process is controlled by a number of "broker agents". These nodes gather information about the different nodes: what they offer and what they need, and put them in contact following a bid system. Compared with plain heterarchies, this partial centralization results in a higher performance at the cost of introducing a potential bottleneck in broker nodes. This performance will depend on keeping the bid system properly tuned.

This architecture is a great alternative for open systems where nodes can appear and disappear. However, it does not provide –without important modifications– natural mechanisms for controlling malicious behaviors.

2.1.4 Types of fusion

The work (Durrant-Whyte, 1988) establishes a classification for the types of sensor fusion. The first criterion is the number of sources, discerning between single sensor and multisensor applications.

Single sensor systems make use of successive observations of a variable over time, along with some knowledge about the dynamics of that variable. Combining both, it is possible to refine the estimated value of that variable –or infer some others. A very simple example is found in how a GNSS differential station estimate its location. It records position fixes over a time window –the larger, the more accurate the result, but typically 24 hours–. Those fixes have an error that is mostly caused by disturbances in some layers of the atmosphere. Then, it makes use of previous knowledge. First, the station has a fixed location; second, the error of the fixes can be approximated as a Gaussian distributed random variable. The best guess of the real station location is obtained by applying least squares algorithm to the recorded fixes.

Multisensor fusion is more complex. Durrant-Whyte identified three different types, according to the relation between input and output information:

Competitive
: Combines data from multiple sensors observing the same magnitude at the same time (redundancy). This type of fusion improves the accuracy of the obtained measure. For example, a voting process can detect biased or defective measures that should be discarded. The average value of several observations is another fusion method that can give a good result in this case.

Complementary
: Combine information of different features or aspects of a same entity, to achieve a more complete description. (Wu, 2003) gives the example of combining pressure and airflow to estimate the propulsive force of a jet nozzle.

Cooperative
: Stereoscopic vision is the paradigmatic cooperative fusion application: two cameras observe the same scene from slightly different angles to obtain depth information. Triangulation from range-only sensors can be included in this same category.

2.1.5 Sensor fusion techniques

After introducing the general aspects of sensor fusion system construction, it is time to describe the basic building blocks: the algorithms that do the actual fusion, mixing the information to get accurate, meaningful and relevant results.

We are classifying fusion algorithms in three big families: classic, statistical and computational intelligence algorithms. This section reviews the most important ones.

2.1.5.1 Classic algorithms

Under this denomination we group those simple fusion techniques that do not have a solid theoretical background, such as calculating average or median values for a set of numerical inputs. Voting processes are inside this category. Participating entities are asked to solve a problem or to take a decision, and each one casts a vote. The final output is calculated from the set of votes, either by majority or using more advanced processes as decision trees. We can find an example in (Franz et al., 2002). This work proposes an architecture for vote based decision taking in distributed environments.

2.1.5.2 Statistical methods

Statistical techniques try to overcome data uncertainty and incompleteness by using probabilistic descriptions of the real world.

Bayesian theory Bayesian theory of inference evaluates the probability of a hypothesis using some prior knowledge about it, together with evidences that provide new information about the truth of the hypotheses. In a typical sensor fusion problem, the hypothesis is related with the state or variable to be estimated. Sensor measures provide the evidence needed to refine the estimation.

Bayesian inference has several disadvantages (Klein, 1999). A common hurdle is the difficulty to determine prior probabilities, something that is worsened by the absence of mechanisms for working with uncertainty (variables for which there is no available data). Bayesian reasoning can be very complex in cases comprising multiple hypotheses, and it requires simultaneous hypotheses must be mutually exclusive. These problems are partially alleviated by other tools as Dempster-Shafer theory, a generalization of Bayesian theory that has been used for data fusion in works as (Wu, 2003).

Filtering Filtering techniques are also based in Bayesian inference, but we consider it apart due to its relevance in the field of sensor fusion. With a filtering algorithm, its is possible to obtain an estimation $\hat{x}_k$ of the unknown real state x_k of a system, based in a mathematical model $f(\cdot)$ of its predicted behavior and partial observations of the real state $Y_k = y_1, y_2, \ldots, y_{k-1}, y_k$ taken over time.

The problem can be expressed in formal terms as:

$$\hat{x}_k = E[x_k] = \int (x_k) p(x_k|Y_k) \mathrm{d}x_k \tag{2.1}$$

Assuming the behavior of the system satisfies the first order Markov property, a filter applies Bayesian inference recursively to refine its estimation of real state over time. Recursion is key to reduce the computational requirements of filters.

Kalman Filter (KF) (Kalman, 1960) represents the paradigmatic example of a filter. It provides a estimation of a system state described as a multivariate Gaussian probability distribution. This estimation is optimal if: *a* the behavior (temporal evolution) of the system is described by a linear system; *b* observations can be related with system state through a linear transformation; *c* the error of each observation is independent from previous observations, and is described by a zero-mean Gaussian multivariate distribution.

Authors have suggested numerous modifications and alternatives extending its application domain to non-linear systems, as the Extended Kalman Filter (EKF) or the Unscented Kalman Filter (UKF) (Uhlmann and Julier, 1997), or to non-Gaussian probability distributions as the case of Particle Filter (PF) (Gordon et al., 1993).

Filtering algorithms play an important role in the experimental part of this thesis. Appendix A explains in depth the Kalman, Unscented Kalman and Particle filters.

Dempster-Shafer Dempster-Shafer evidence theory is a generalization of Bayesian theory that overcomes some of its limitations: *a*) allow union of hypothesis; *b*) does not require prior knowledge *c*) include into computation events or probabilities that are not known.

The reasoning process starts with a set of basic, mutually exclusive hypotheses that cover the full probability space. It builds a new set containing all the possible combinations of basic events, and reasons over the elements of this set. Instead of using probability distributions, as in Bayesian reasoning, a "belief" functions is defined. Dempster-Shafer theory includes the operational mechanisms for working with belief functions, including the possibility of not having information about some hypotheses.

Thanks to its advantages, this technique has been the choice for real time systems operating in environments with a high uncertainty, as mobile robots (Murphy, 1998).

Fuzzy logic Fuzzy logic is an extension over classical logic that deals with variables defining imprecise (fuzzy) categories, this is, where the limit between values are not clear. Although there exist works on this topic since the decade of 1920, fuzzy logic is considered to be born with this denomination in 1965, when L. Zadeh formulated its theory of fuzzy sets.

Some authors have taken positions in favor of or against the use of fuzzy logic, using arguments about whether it is needed, or it is a subset/superset of other tools. Apart from these questions, fuzzy logic has proven to be a valuable asset, because variables, categories and rules are formulated closely to how they exist in human thought.

As an example, temperature can be represented as a fuzzy set with five categories: cold, cool, optimal, warm and hot. The correspondence between actual temperatures and the categories depends on the application domain –the concept of "optimal" is different in the controller of an air conditioning system and in a foundry oven. Imbuing the components –variables, rules– with knowledge about the problem to solve simplifies the creation expert systems. On the other hand, this complicates the creation of general purpose decisions tools based on fuzzy logic.

Because of its rule-based nature, fuzzy logic is easy to apply in high-level data fusion processes. (Hall and McMullen, 2004) contains a discussion about the applicability of expert systems based on fuzzy logic to data fusion. (Stover et al., 1996) presents a general purpose architecture based on fuzzy logic, that can be applied to full data fusion problems (including sensor fusion).

Fuzzy logic has also been successfully applied to low level data fusion problems. A first approach consists on applying the rules of an expert system to low level data, for obtaining fused information. (Solaiman, 1998) elaborates terrain maps from multisensor information. Terrain fragments (pixels) are classified according to a set of fuzzy rules that compose a context-sensitive reasoning process. In (Cohen and Edan, 2004) projects sensor measures over a grid map, and applies a fuzzy reasoning system to create occupancy maps for navigation.

Fuzzy logic can also be used as small reasoning components plugged into a traditional sensor fusion system. The fuzzy reasoning process works analyzes the data and affects the behavior of the system or the final products of the fusion. The paper (Sasiadek et al., 2000) takes a basic GPS/INS navigation system based on an EKF. It is augmented with a fuzzy reasoning component able to detect when GPS is reporting positioning errors below the real one. That information is used to modify the covariance matrix used to update the filter, leading to better results. Wang and Gao describe in (Wang and Gao, 2005) how a fuzzy system can be used to achieve high accuracy navigation using low cost sensors that have not been adequately calibrated. The input of inertial sensors are fed into the fuzzy reasoner, which can discern among types of motion (stopped, turn maneuvers, accelerations). This information is used to change the prediction model of a KF.

2.1.5.3 Computational intelligence

This category encompasses a heterogeneous set of techniques which were born in the field of Artificial Intelligence (AI), conveniently adapted to sensor fusion problems.

Artificial Neural Networks (ANN) ANN are a family of automatic machine learning techniques, inspired by the neural systems of animals. A network is composed by several interconnected "neurons", which receive an input and produce an output $f(a) = b, a \in \mathbb{R}^n, b \in \mathbb{R}^m$.

Generally speaking, we can consider ANN as a universal function approximator. Their capabilities extend to multidimensional non-linear functions. The information about the approximated function is coded in the network structure: which neurons are connected, and the "weight" of those connections.

In the field of Data Fusion (DF), ANN have been used in those problems that make direct use of their ability for pattern learning. Image fusion is an area with great potential for ANN: (Li et al., 2006)(Wang et al., 2010)(Carpenter et al., 2005) use an ARTMAP type network (Adaptive Resonance Theory (ART)) for fusing images that have been labeled using inconsistent semantic criteria, at the same time that learns the relation between labels and how their meanings overlap.

ANN have been used in robots to create grid occupancy maps from the fusion of sonar and infrared sensors (Barbera et al., 2000). The problem of navigation was approached with ANN in (Sharaf et al., 2005), a work proposing the use of Radial Basis Function Neural Networks (RBF-NN) for fusion Global Positioning System (GPS) with Inertial Navigation System (INS) sensors.

Evolutionary Algorithms (EA) Evolutionary algorithm encloses a family of metaheuristic optimization techniques inspired by the theory of Natural Selection. These algorithms start with a "population" (a set) of candidate solutions for the considered problem, whose "fitness" (how good a solution they are) is evaluated. Best individuals are likely to reproduce in a larger number than others with a low fitness, conforming a new "generation" of solutions. EA define processes for introducing variability in the population, such as random mutations. The whole process is cycled until a satisfactory solution is found.

EA can work in huge search spaces with a high dimensionality, even when the problem presents additional difficulties such as non-linearity, non-convexity, discontinuities, non-differentiability, multimodality and/or noise. Due to its stochastic and domain agnostic nature, EA are well suited to a wide variety of problems. They are difficult to apply to real time problems as sensor fusion ones, however, since they require a high number of iterations to reach good solutions.

The survey (Maslov and Gertner, 2006) presents a compilation of EA-based solutions to data fusion problems. Most solutions are oriented to high-level DF, as selecting the set of features to be extracted from a dataset to optimize the results obtained by a classification algorithm.

2.2 Context-aware computing

Context-aware computing is the discipline where the concept of context has been best analyzed and defined. It is also a mature discipline, making it makes a good starting point for acquiring the basic concepts relating computation and context information. This section reviews the existing literature in context-aware computing with special emphasis in two aspects: clarify the concept of context (already introduced in 1.1.2), and explore existing architectural solutions for managing context information (acquisition, representation, storage and dissemination).

In 1991 is published (Weiser, 1999) (we are referencing a reprint in a special issue dedicated to Prof. Weiser), which is considered the seminal article of Ubiquitous Computing. It predicts the birth of information systems integrated with the environment, so that users can benefit from the services they provide without requiring any explicit interaction. The idea is alternatively called Pervasive Computing.

The term "Context-Aware Computing" appears for the first time in (Schilit and Theimer, 1994). That work postulates that the information we can sense from our close environment (see, hear, touch) is useful for operating when non familiar conditions arise, as also to cooperate with other agents presents in that environment.

Both disciplines are related since, in Ubiquitous Computing, users context is the starting point to determine an agenda of actions, and how they must be carried out. However, context information is something difficult to deal with. (Dey, 2000) cites the following reasons:

- Context is acquired from non-traditional devices, different from classical computer peripherals.
- Getting meaningful and compact information may require complex processes for transforming or abstracting raw input data. E.g. from a sound record to a statement "John is talking".
- It may require mixing heterogeneous data from distributed sources.
- Context is dynamic: can vary over time, location and be influenced by the presence of other agents.

Sensor fusion already faces, at least, two of these difficulties: acquisition from non traditional devices, and mixing heterogeneous data from distributed sources.

2.2.1 Architectures for context-aware computing

We can find in (Baldauf et al., 2007) a detailed analysis of solutions for creating context-aware computation systems, including architectures, processing models and how context information

can be represented. Figure 2.4 reproduces a summary table included in the original work. An architecture for context-aware computing is determined by three aspects:

- How context information is acquired ("Sensing" column in the figure)
- How context is modeled and represented ("Context processing" column in the figure)
- How context is accessed by system components ("Context model" column in the figure)

Table 4 Summary of discussed approaches

Architecture	*Sensing*	*Context model*	*Context processing*	*Resource discovery*	*Historical context data*	*Security and privacy*	
CASS	Centralised middleware	Sensor nodes	Relational data model	Inference engine and knowledge base	n.a.	Available	n.a.
CoBra	Agent based	Context acquisition module	Ontologies (OWL)	Inference engine and knowledge base	n.a.	Available	Rei policy language
Context Management Framework	Blackboard based	Resource servers	Ontologies (RDF)	Context recognition service	Resource servers + subscription mechanism	n.a.	n.a.
Context Toolkit	Widget based	Context widgets	Attribute-value tuples	Context interpretation and aggregation	Discoverer component	Available	Context ownership
CORTEX	Sentient object model	Context component framework	Relational data model	Service discovery framework	Resource management component framework	Available	n.a.
Gaia	MVC (extended)	Context providers	4-ary predicates (DAML + OIL)	Context-service module (first-order logic)	Discovery service	Available	Supported (e.g., secure tracking, location privacy, access control)
Hydrogen	Three layered architecture	Adapters for various context types	Object-oriented	Interpretation and aggregation of raw data only	n.a.	n.a.	n.a.
SOCAM	Distributed with centralised server	Context providers	Ontologies (OWL)	Context reasoning engine	Service locating service	Available	n.a.

Figure 2.4: (Baldauf et al., 2007) Architectures for context-aware system development

Next sections explain the most relevant proposals regarding the three aspects above mentioned, which ones do we consider more appropriate for our purpose, and the arguments supporting the decision.

2.2.2 Context acquisition

The thesis (Chan, 2004) identifies three different approaches for acquisition of context information:

Direct sensor access

Used in devices with sensors built in. The software accesses information directly from source sensors, requiring some knowledge about drivers and APIs. This feature makes

it unsuitable for distributed systems. This method is also not scalable since it does not provide control mechanisms for concurrent access to sensors.

Middleware infrastructure

A middleware is a layer of software that bridges the abstraction level gap between two components, in this case the sensors and the software using it data. A sensor middleware hides the low level details of sensors (synchrony issues, irrelevant parameters, trivial data preprocess), allowing to create client software that is more extensible and reusable. Most middleware infrastructures are designed as a " translation layer" that changes the terms of the communication with sensors. This means that the access to data sources is still pretty direct, expected to have a low performance penalty, but does not provide mechanisms making it suitable for distributed systems.

Context server

Represents a further step over the middleware approach that grants clients the access to remote sources of data. Sensors are accessed by a central component, the context server, that gathers and stores the data. Clients communicate with the server for getting data. This approach is optimal for distributed systems, specially when clients are resource constrained elements. The context server can take care of the resource intensive operations related with accessing and pre-processing sensory data to extract the actual context information. It also simplifies client implementation, that only have to stick to a single communication protocol/interface for getting any kind of sensor data.

Figure 2.4 includes other paradigms for context access, although they can be considered variations over one of the three major styles described above. The context broker agent of CoBra architecture (Chan, 2004) is an extension of the context software for generating context knowledge that cannot be inferred from single sensors (intentions, roles, spatial and temporal relations). The author cite the potential bottleneck of a single context server, that in the case of broker agents can be solved by creating a team of them that keep a coherent model of context and a consistent knowledge base.

The Context Toolkit (Wu, 2003) propose using context widgets for acquiring context. We explain the widget concept in section 2.2.4, since it is extensively used in the toolkit for other purposes. Our proposal will gather context knowledge in a centralized repository that is also responsible of some additional tasks. We can describe it as a mix of context server and middleware infrastructure, because it is not (initially) prepared for working in a distributed environment but the access to context is more similar to the server proposal.

2.2.3 Context modeling and representation

Context information has to be transformed to a representation that allow to store and process it. The six most relevant approaches are described in (Strang and Linnhoff-Popien, 2004):

Key-Value pairs

Is one of the simplest models for storing information. Keys are unique identifiers of an entity or feature of interest, which are assigned a value. Key-value storage model has recently gained popularity in NoSQL databases and big data applications, because it allows to store unstructured data and searches are easy to make parallel. On the other hand, the application accessing the data is responsible for knowing how data is structured. Operations about how elements are related can be difficult to implement and have a costly execution.

Markup scheme models

A hierarchical data structure relating attributes with content. Any language defined over XML is a good example. Markup languages are usually well defined by a companion scheme, and have limited capacity to express constraints such as basic data types and ranges. One of its main drawbacks is that they lack a standard mechanism to merge models, so that it is not easy to have partial descriptions that can be finally used together.

Graphical models

Some authors have used the Unified Modeling Language (UML) for modeling context, as (Sheng and Benatallah, 2005). In (Henricksen et al., 2002), authors extend the Object-Role Modeling (ORM), which is a technique for designing and querying relational database models that represent a set of business rules using terms understandable for non-technical users. They add persistence and source features to ORM facts and a dependence indicator. While graphical models formality is poor, they are a strong option to describe the structure of contextual knowledge, and are easily understandable by humans.

Object Oriented approaches

Uses the principles of Object-Oriented design to model context. It has the advantages of this programming model, as encapsulation or inheritance, and separates processing and representation logics.

Logical Based Models

Using formal logic to model context requires defining facts (things that are known), and rules that relate facts. This approach allow to apply automatic reasoning techniques to infer new facts, as well as validate or remove existing knowledge.

Ontologies

An ontology is a tool that allows to define a conceptual schema describing a knowledge domain. Ontologies are characterized for having a high expressiveness, allowing to describe not only the concepts and how they are related, but also domain constraints and assumptions that other tools cannot. Since knowledge is well structured and formalized, it is possible to apply reasoning processes over ontologies. Finally, ontologies enhance knowledge reusability: they can be merged and imported.

The work (Strang and Linnhoff-Popien, 2004) also analyze appropriateness of each modeling approach, according to six different criteria: *a*) distributed composition, *b*) partial validation, *c*) richness and quality of information *d*) incompleteness and ambiguity, *e*) level of formality, and *f*) applicability to existing environments. They conclude that ontologies are the most promising tool, although this does not discard any of the other methods for its use in ubiquitous computing systems.

Not all the enumerated criteria are relevant for the construction of a centralized sensor fusion system. Richness and quality of information is an important feature because our proposal does not impose any restriction on the type of context that can be modeled. The representation should also enforce non-ambiguity, and a high level of formality to make possible applying automatic inference processes. According to the summary chart at the end of the survey, ontologies are the best option for this set of criteria, and we will use it in the design of our framework.

If we take a look at figure 2.4, we can see that ontologies are a popular choice for modeling context. An additional advantage of ontologies is that they can be transformed if necessary to other representations: the proof is that OWL ontologies can be physically stored in XML and tern-based formats. However, our proposal does not enforce a particular representation. The selected applications (chapters 4, 5) encourage mixing models when it has benefits for the implemented system. In the car example, context information is represented through an ontology, but it is processed by a rule-based reasoning system.

2.2.4 Context management and access

Another important aspect regards how context knowledge is disseminated across the system in charge of using it. The thesis (Winograd, 2001), a paradigmatic work in the area of Context-Aware Computing, cites three well differentiated styles for managing context information:

Blackboard

Blackboard architecture (Engelmore and Morgan, 1988) defines a centralized information repository that can be read and modified (insertions and deletions). All the components in the system access, thus, a shared information base.

Blackboard architectures offer a subscription service for consumers wanting to be notified when the information relevant to them appears or changes. While representing a popular choice because of its simplicity, they have the drawback of not scaling properly to large volumes of data or great number of consumers, as it happens with most centralized systems.

Service-oriented (client-server)

The client-server approach is located in the opposite side of the spectrum than blackboard architectures. Its components operate independently, without a global control facilitating or regulating how information flows: service consumers are responsible of discovering servers, connecting with them and managing how information is transmitted.

Among its disadvantages, we can cite the extra complexity added to each component, and the larger computational load associated to accessing information access. This makes service-oriented architectures a poor choice for systems where the low level components are tightly coupled (connections never change and exchange large volumes of data). On the other hand, they offer great robustness and flexibility: this is the reason why it become the standard approach for building the Internet, including web page serving and remote APIs.

(Hong and Landay, 2001) offers a good analysis of this architecture applied to context management. However, most of the research on this architectural style is focused in the complex aspects as automatic service discovering (Gribble et al., 1999).

Widget

Halfway between the two previous proposals, we can find the widget based architecture. (Dey, 2000) is one of the most remarkable works using it.

A Widget can be interpreted as an extension to software of device controllers: provides a common interface for accessing and controlling its underlying elements. This simplifies the process of connecting/disconnecting elements. The term "widget" was first used in the world of visual user interfaces. It makes possible creating applications with visual interface that can change their aspect or work in different platforms without having to modify the source code.

Widget based architectures require a global managing component, which must know all the existing modules and how they are connected. We can see this as a service-oriented approach where the discovery and negotiation parts reside in a central component. This solution is modulable, scalable, and keeps the computational cost within acceptable levels even for highly coupled systems.

Widget style is, among the three analyzed options, the most promising architecture for our proposal. The reasons include that the developed framework aims to centralized systems

with potential for tight coupling between low level components (high refresh rates, large data volumes), but we want to maximize the flexibility for changing how the composing elements are connected. As in the Context Toolkit (Wu, 2003), the widget concept will be used for more things apart from managing context.

2.3 Context in information fusion processes

The synergies between data fusion and context-aware computing have been explored with much more attention in the latter discipline. This is a natural consequence of the usefulness of low level data fusion techniques for generating the context information required by context-aware algorithms.

However, data fusion community is increasingly aware of the importance of context information as a fundamental tool for building systems able to operate in a real, open world. They need to adapt to unforeseen situations and can also benefit from the exploitation of unstructured or unexpected information.

Starting at the beginnings of data fusion discipline, it is relatively easy to find applications that apply specific context information to improve results. As an example, (Blasch et al., 2013) reviews some popular approaches for enhancing tracking processes by using context. In (Caron et al., 2006), authors describe a GPS/IMU fusion navigation system with that adapts the fusion solution depending on the "context", which is defined here as the validity status of GPS/inertial measures. The system monitors the quality of input data, switching between filtering solution depending on the available measures.

But it is the last decade when fusion community has intensified the efforts on that direction, showing an explicit interest on exploiting context information. It is possible to appreciate that trend by reviewing the proceedings of the International Conference on Information Fusion:

- 2007 (10th edition): special session on "Context information in Data Fusion".
- 2008 (11th edition): features (Ricquebourg and Delahoche, 2008), a paper about a sensor-based application that exploits sensor redundancy and fusion consistency to detect when a sensor is not working properly (providing degraded or invalid input).
- 2010 (13th edition): includes a special session "Intelligent systems for context-based information fusion", supported by the argument that "context-based information fusion has matured during the last decade and many effective applications of this technology are now deployed".
- 2011-2012 (editions 14th, 15th): special sessions "Context-based information fusion".

2.3.1 Context-aware architectures for information fusion

At the moment of writing this thesis, we could only find two architectures for data fusion systems that take context explicitly into account.

2.3.1.1 DAFNE project

The DAFNE project (Ditzel et al., 2011) is an European initiative that "aims to design an experimental distributed multi-sensor fusion engine that will combine data from heterogeneous sensors in urban war-fare scenarios to enhance situational awareness during military operations.

Authors propose a layered architecture according to the levels of the JDL model, where each layer manages a local base of context information that is used to configure the intra-level processes. While context is cited as a factor for achieving operational flexibility, it receives a very domain-specific definition in this work:

> [...] relevant parts of the environment, such as geographical information and databases or rules for classification and definition of threats.

The layered approach imposes an artificial restriction on not mixing information with different abstraction levels. In a system using DAFNE engine, a level 0 sensor preprocessing task could not use a level 2 contextual fact, and vice versa

We think that, in order to exploit the full potential of context information, fusion processes must be capable of using available facts with any level of abstraction. Apart from that, level 4 processes (adaptation, refinement) are transversal so that they need contextual and fusion information from all the different levels. These statements are adequately exemplified in the experimental sections.

Taking into account these considerations, we can argue that DAFNE architecture does not satisfy the motivation of this thesis.

2.3.1.2 Adaptive Architecture for Information Fusion

The system presented in (Llinas, 2010) defines a generic Information Fusion Framework strongly focused in the creation of adaptive fusion processes. These adaptive processes represent the 4th level of the JDL model and can affect every part of the fusion application, including algorithms internal parameterization, the interaction between them, and sensor/resource management.

The components in charge of doing the adaptation are spread over the whole architecture, and receive the name of "adaptive logic" blocks. They combine information about current system performance and relevant contextual data. This architecture is intended to serve fusion

systems covering all the levels of the JDL model, which requires solving complex problems as bridging semantic gaps between processes or defining mechanisms that can resolve the relevance of arbitrary context information for complex high level fusion processes.

Since this is an ongoing project that has not be fully defined, we can consider that it does not collides with this dissertation. We will use, though, some of its ideas and points of view in our proposal.

2.3.2 Previous works in the research group

This dissertation is written as part of the research conducted within the Applied Artificial Intelligence Group from University Carlos III of Madrid. This group also develops other research lines ranging from Ambient Intelligence (AmI) to multi-objective optimization or augmented reality. The next paragraphs summarize the most relevant works mixing data fusion and context information.

Automated video surveillance systems can use fusion techniques when they include multiple cameras. The article (Sanchez et al., 2007) describes a video tracker augmented with a symbolic reasoning layer that modifies its output, using specific context information. This proposal was refined in (Gomez-Romero et al., 2011), which develops a framework for fusing prior (static) context information coded according to an ontology with real time video data. The final product is a high level interpretation of the observed situation. Some results have been obtained in group activity recognition, as (Pozo et al., 2011).

The goal of (Cilla et al., 2011) is to improve the recognition of basic human activities (walk, run, sit) in a multicamera system. A probabilistic model is developed with this purpose, which describes the spatial context for each action, this is, in which camera and zone of the image is more likely to observe the actions. It results in a higher recognition rate.

According to (Gomez-Romero et al., 2009), ontologies are the choice for representing context information. This paper reasons why they are a flexible, powerful mechanisms that is well suited for the type of symbolic reasoning used in higher levels of data fusion. That conclusion was successfully proved in (Gomez-Romero et al., 2014), which describes a case of study where video information is used for suspicious activity recognition in a harbor surveillance scenario. The followed approach can be directly applied to other sensors such as radar.

The group has made efforts to consolidate the theoretical aspects of context based information fusion. For example, (Gomez-Romero et al., 2010) provides a formal description for how to use context information to improve high level data fusion processes. The work (Blasch et al., 2013) identifies different ways in which context information can be used to solve a target tracking problem (low level information fusion). Finally, (Garcia et al., 2011) discusses the procedure followed in the design of a hybrid surveillance system for harbor areas, which

combines context information two different reasoning processes: deductive reasoning about the expected behavior of a boat taking into account its features, and abductive reasoning under high uncertainty conditions.

In the line of AmI research, (Blázquez Gil et al., 2012a) presents a distributed architecture for recognizing the context of smartphone users. This work is an example of data fusion oriented to consumer market, although its results can be translated to other domains as monitoring people with health issues that live on their own (Blázquez Gil et al., 2012b).

3

Proposal: adaptive framework for context-aware sensor fusion

The introduction of this thesis (see section 1.3) establishes two goals that will be fulfilled in this chapter. The first one is an analysis of the different ways in which context information can be applied to sensor fusion processes. The second one is the development of a framework that supports the creation of context-aware sensor fusion systems. We will cover the theoretical parts first, since they will have a big impact in latter design decisions.

3.1 Preliminary concepts and analysis

3.1.1 Uses of Context Information in Sensor Fusion systems

Back in the introduction section, we specified that context information must be used in the proposed framework for two different purposes: enhance fusion products thanks to an improved understanding of the observed entities, and also creating systems that can adapt to changes in the environment. In this section, we are going to detail where and how context information can be used to achieve those goals.

The design and tools of the proposed framework shall facilitate the use of context information in all the fusion related processes. The first step is, thus, defining which these uses are. Our analysis starts from the conclusions of (Schilit and Theimer, 1994), which identifies four different ways of context information in user-centered context aware systems:

- *Proximate selection:* emphasize or make easier to select those objects that are in the same "locus" as the user. Locus is a term originally used in biology to refer to positions of interest over genetic sequences, and most context-aware systems identify the concept with the user physical location by default. An example of proximate selection: if a user

wants to share the screen of his/her smartphone screen, a context-aware system should present nearer displays first.

- *Automatic reconfiguration:* refers to the addition or removal of elements (drivers, devices, software modules) and modifying how they are connected, depending on the context. In a context-aware system may refer to user needs.

- *Contextual commands:* are those that can take into account context to modify their response. For example, a contextual "print" command can automatically deliver documents in the printer closest to the place where they will be used.

- *Context-triggered actions:* Can be defined as simple "IF-THEN" rules. This approach works well for systems with a high level of abstraction, because it permits to translate human knowledge in a very natural way. For example, Microsoft developed a research prototype for smart homes following this concept (Dixon et al., 2012), where users can program automatic actions such as "Turn on the lamp when room door is opened" using their smartphones.

We propose a modification of these uses to better fit the domain of sensor fusion applications: First change is changing "proximate selection" to "relevance based selection". This requires extending the concept of "locus" to the more generic idea of determining when a piece of information, a sensor or an algorithm is useful in a certain situation.

The relevance of an element depends on its inherent features and also on the context. Inherent features encompass factors that can be statically determined and are supposed to remain stable along time, such as the features of the information it produces/consumes, computational requirements or energy consumption. It is easy to exploit inherent features: they have a low level of abstraction, can be hardcoded in the fusion logic or assumed as a precondition for the design of the system.

Determining the relevance of context information is a much harder task, specially when it has a high level of abstraction. These kind of difficulties are cited in (Liggins et al., 2008) (p.447) as the main issues in Situation and Threat Assessment (levels 2 and 3 of the JDL model), but they also apply to our problem:

- Weak ontological constraints on relevant evidence: it is difficult to model which information is relevant on each situation, and how it is relevant.

- Weak spatio-temporal constraints on relevant evidence: it is difficult to establish for how long an evidence is relevant/valid on each situation.

Context exploitation present an additional challenge: the same piece of information can accept multiple representations differing on abstraction level or the point of view they are

looked from. Thus, in order to adequately determine the relevance of some element under a particular context, we need to express that context in a restricted and standardized format that minimizes ambiguities and duplicities. In section 3.3.2 we present a proposal to use ontologies for this purpose, together with a minimal language that describes both the domain of the sensor fusion application and its context. Another option is to create a logic that can transform information between equivalent representations, but this solution is completely domain-specific and we will not explore it in this document.

This work assumes that the relevance of context is determined beforehand, either hardcoded in the logics of the software, or expressed as part of the problem space description. Automatic relevance determination is out of the scope of this work.

Automatic reconfiguration can be applied straightforward to sensor fusion domain, referring to sensors, other information sources, and algorithms. A sensor fusion solution is a particular arrangement of such elements, so automatic reconfiguration affects which of them are present and how they are connected.

Contextual commands, in the sensor fusion domain, are identified with fusion algorithms that can use context information to improve their output in some way. As it is explained in section 2.3, this has been the classical use of context information to enhance fusion processes. Contextual algorithms are defined as any other algorithm, declaring which context information they can use so that the framework takes care of providing it.

The last considered used of context information, triggering actions, does not constitute a separate case in Sensor Fusion domain in the sense that its implementation does not require specific mechanisms. It can be considered a complementary aspect of the previous usages. The actions in a sensor fusion system are sensor/algorithm management (automatic reconfiguration), or something built into a fusion algorithm and, thus, part of a contextual command.

There are some additional aspects regarding how context is created, stored and used. In (da Rocha et al., 2008), authors discuss the requirements of a middleware for context-aware applications with ubiquity features. We are not interested in context-aware applications and do not pursue ubiquity, but the paper presents some interesting thoughts. It defines 12 requirements, some of which are relevant for our case:

- Support for context evolution: the framework must support the inclusion of new context types/concepts without affecting the execution of consumer processes.

- Extensible abstractions for accessing and using knowledge: the middleware should allow access to context information through mechanisms that are adequate for the level of abstraction of the target applications. It should allow the specification of new abstractions in top of the existing base of knowledge.

- Architectural independence: related with permitting access to context information from different platforms (hardware or software). Although this is clearly defined with a distributed scenario in mind —not our case—, we are interested in on-line integration of sensor sets that can use context information for enhanced results.
- Decoupling between context management and inference mechanisms: authors argue that the mechanisms for context inference must be decoupled from context management infrastructures, because it results in a good trade-off of expressiveness, consistency, computational efficiency and reusability.

These requirements will be at least partially integrated in the framework.

3.1.2 Ontologies as a tool for adaptive systems meta-description

Back in section 2.2.3, we identified ontologies as the right tool for describing problem space and relevant context information. Ontologies have a good number of benefits as a knowledge representation tool (Bürger and Simperl, 2008): they allow to describe concepts, domain assumptions and constraints, and it is possible to apply automatic logical inference processes over them. One of the most important and popular applications of ontologies is to define a common vocabulary that ensures some degree of interoperability between automated systems, and with this purpose they were chosen as the basic description tool for building the semantic web (Shadbolt et al., 2006). Another advantage is that ontologies can be easily interpreted by both humans and computer.

The paper (Lee and Zeigler, 2010) describes how the System Entity Structure (SES) ontology framework can be used to improve how information is exchanged. The idea is to enable centralized Data Fusion processes that acquire information from networked environments. It is used to transform raw data to different (high-level) representations that satisfy the needs of the various layers in a Data Fusion system.

Dockhorn-Costa dissertation (Costa, 2007) describes the fundamental concepts and structures for supporting the development of Context-aware applications. It contains a very detailed and solid work on modeling, representing and using context information. It defends the use of foundational ontologies for "representing conceptualizations that are truthful to reality".

These and other works as (Wu, 2003) contain nice and clear examples on using ontologies to describe context information and components of computational systems. The article (Chen et al., 2003) describes two tools: the COBRA-ONT ontology for supporting context-aware systems in an intelligent meeting room environment, and an inference engine for reasoning about context information. Our proposal takes some of the ideas and principles of these works.

This work proposes the use of ontologies for representation purposes, but not necessarily for inference or reasoning, because exploiting the knowledge in ontologies can require an excessive

computational effort. This problem can be mitigated through the use of complementary techniques as rule-based systems. In the experimental section 5 we demonstrate how to these two techniques can be hybridized into a solution that combines their respective advantages. Ontologies are also criticized for their high design cost, but we still encourage their use for representing knowledge when reusability or interoperability between collaborating systems are important features. This advantage is illustrated in the same experimental section: the navigation system, mounted in a vehicle, discovers a smartphone that can contribute with its sensors to the fusion process. Collaboration is possible because both platforms share a common language for exchanging the required information.

3.1.2.1 Considerations about redundancy, ambiguity and non-orthogonality

When context information is applied to enrich the solution of a problem, the first step consists in identifying which pieces of information are actually relevant. Secondly, it is necessary to select its best representation, since any particular piece of information admits a potentially infinite number of them.

As a rule of thumb, any representation of knowledge should be:

- Complete: there are not pieces of information that are relevant to the problem but cannot be expressed using the selected representation.
- Unambiguous: each part or aspect of the selected representation, and any piece of information expressed using it must have unequivocal interpretation.
- Non-redundant: a piece of information must have a unique representation in the system.
- Minimal: it has not superfluous elements.

When designing a representation for context knowledge, however, some of these requirements can be either very difficult to hold or not adequate. The most notable case is unambiguity, which can be solved by selecting a set of orthogonal features for representing data. Orthogonality means that the value of one feature is not related with the values of the others, because the features represent disjoint aspects of the reality. Several works have pointed that this is extremely complicated for context information because of the complexity of the data, the mix of abstraction levels and the implications it has on a problem.

The next list presents some of the difficulties associated or caused by how sources of context information provide their data:

- Heterogeneity in abstraction level and language: radio communications in natural language about a traffic accident, terrain altimetry as a matrix of numbers

- Incompleteness and uncertainty on some features.
- Overlapped sources: two or more sources provide the same information. However, since the data presented by the sources can have different abstraction level, and be partial and uncertain, it can be very difficult to detect and eliminate the redundancy.

Because of these factors, the task of selecting an unambiguous and non-redundant representation for context information can be far from trivial. Furthermore, such features can be not harmful for the entities that will consume/use the information. Each algorithm in a sensor fusion solution can require or prefer a representation that suits its internal abstraction level, granularity or point of view of the problem.

Thus, a sensor fusion system can ask for the same piece of information expressed in different ways simultaneously. As a conclusion, in spite that it is (in theory) possible to design a "perfect" representation of context information, in most cases it is not practical because it introduces technical difficulties and require actually more effort to exploit context knowledge. Designers are encouraged to identify representation ambiguities and redundancies, and make sure that the system implements mechanisms to avoid inconsistencies and contradictions in actual data, and/or the potential inconsistencies have a bounded impact in the system.

3.1.2.2 Introduction to OWL/RDFS ontologies and RDF language

The examples used in this section and the ontologies created for the selected applications are written using Web Ontology Language (OWL) (Lacy, 2005). OWL is constructed in top of Resource Description Framework Schema (RDFS), a more basic language for describing ontologies. In RDFS, a vocabulary (the terms vocabulary and ontology can be used indistinctly in this context) is a set of descriptions of classes and properties. The ontology is populated with individuals, that are elements defined and interrelated according to the rules of this vocabulary. Both RDFS and OWL can be serialized using the XML representation of Resource Description Framework (RDF) language (Brickley and Guha, 2004). We have chosen XML-RDF as vehicular language because it is accepted by the majority of ontology edition and visualization tools, and is easy to process with Jena library, written in Java language and part of the Apache tools.

In OWL and RDFS, a class is a category for individuals, quite similar to Object Oriented Programming (OOP) (Object Oriented Programming) classes stripped from its behavior (only data). Individuals can belong to several classes at the same time. For example, we can have an ontology with classes Person and Child (Child is a subclass of Person) that is filled with individuals John and Carl. John is an adult, so it has class Person, but Carl is a kid so it has classes Person and Child.

```
1 <!-- Class Person -->
2 <owl:Class rdf:about="#Person"/>
```

```
3
4  <!-- Class Child as a subclass of Person -->
5  <owl:Class rdf:about="#Child">
6    <rdfs:subClassOf rdf:resource="#Person"/>
7  </owl:Class>
8
9  <!-- Individuals John and Carl -->
10 <owl:NamedIndividual rdf:about="#John">
11   <rdf:type rdf:resource="#Person"/>
12 </owl:NamedIndividual>
13 <owl:NamedIndividual rdf:about="#Carl">
14   <rdf:type rdf:resource="#Child"/>
15   <rdf:type rdf:resource="#Person"/>
16 </owl:NamedIndividual>
```

The other element of a vocabulary are properties. We are using two types of properties in this work: object properties and datatype properties. Object properties are directed relations between two individuals of the ontology, this is, they have a subject and an object. It is possible to define a domain and a range for each property, which restrict the allowed classes for the subject and object individuals. In our example, we can define the sample object property "isParentOf" with domain Person and range Person, and use it to express that "John isParentOf Carl".

```
1  <!-- Define property. Domain and range are class Person -->
2  <owl:ObjectProperty rdf:about="#isParentOf">
3    <rdfs:domain rdf:resource="#Person"/>
4    <rdfs:range  rdf:resource="#Person"/>
5  </owl:ObjectProperty>
6
7  <!-- Add object property to individual John to state the relationship
8     with Carl (modifies previous declaration) -->
9  <owl:NamedIndividual rdf:about="#John">
10   <rdf:type rdf:resource="#Person"/>
11   <isParentOf rdf:resource="#Carl"/>
12 </owl:NamedIndividual>
```

Datatype properties relate individuals with literal values, and can be seen as the attributes/-fields of classes in OOP languages. They also have a domain over the classes of the ontology. An example is the property "hasAge", with range over the non-negative integers, so that we can express that "John hasAge 28" and "Carl hasAge 5".

```
1  <!-- Define datatype property.
2     Domain over class Person, range over non-negative integers -->
3  <owl:DatatypeProperty rdf:about="hasAge">
4    <rdfs:domain rdf:resource="#Person"/>
5    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
```

```
</owl:DatatypeProperty>

<!-- Add datatype property to individual declarations -->
<owl:NamedIndividual rdf:about="#John">
  <rdf:type rdf:resource="#Person"/>
  <isParentOf rdf:resource="#Carl"/>
  <hasAge rdf:datatype="&xsd;nonNegativeInteger">28</hasAge>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="#Carl">
  <rdf:type rdf:resource="#Child"/>
  <rdf:type rdf:resource="#Person"/>
  <hasAge rdf:datatype="&xsd;nonNegativeInteger">5</hasAge>
</owl:NamedIndividual>
```

OWL augments RDFS with additional semantic capabilities, including:

- Specify cardinalities of object relations and datatype properties.
- More interesting and complex ways to describe classes. Apply set operators, e.g. union or intersection of classes. Define two classes as disjoint to indicate that no individual can belong to both classes at the same time.
- Relationship between properties, that can be declared as transitive, symmetric, functional and also define one property as the inverse of other (in the above example, we could declare the object property "isSonOf" as inverse of "isParentOf").

3.1.3 Automatic sensor and algorithm selection

As of today, we have not seen the advent of automatic tools for generating sensor or data fusion solutions. The reason is that it is not a simple problem: the gains achieved by a fusion process come either from adding new knowledge to that provided by the information sources (e.g. the prediction model in a tracking filter, which gives information about how the target moves), from using some knowledge about the information sources (characterizing the performance of a GPS receiver under different circumstances and take that information into account during fusion), or how sources are related (using a complementary filter for fusing a biased but fast reacting gyroscope with an unbiased but slow magnetometer, and get an stabilized attitude estimation).

Although part of this information can be potentially translated into rules, data or models, most of it is implicitly integrated in the fusion solution as the selected architecture, algorithm parameterization or embedded rules. Furthermore, there is not a clear methodology for reaching this knowledge: it is determined through experimentation and testing, and is difficult to extract (make explicit) afterwards.

Regarding the use of sensors, some authors claim that most sensor fusion works need to use all the sensors continuously because the more information is available, the better will be the result. This is especially true when the proposed solutions exploit domain knowledge through the subtle relations between sensors or problem variables.

Following the traditional approach of tailored solutions using all the sensors leads, however, to highly coupled solutions that are less robust against (a) sensor failure/outage, (b) sensors showing uncharacterized negative effects (c) external conditions invalidating prior knowledge of the domain. While fully automatic algorithm selection is generally not an option, these concerns have been expressed in previous works. For example, (Liu and Gingras, 2007) describes a system that combines fault detection with data fusion to create a robust positioning navigation system.

The article (Cohen and Edan, 2008) presents a rule-based framework that selects the most reliable sensors and most suitable algorithm for fusing sensor data in a mobile robot platform. The framework does not require any preliminary knowledge about the sensors involved, although the presented solution is limited to sensors whose measures can be translated to a grid occupancy map –cameras and ultrasonic sensors in the experiment. That simplification provides a homogeneous view of the sensory information, making possible to calculate comparable quality metrics.

Another typical scenario consists on those systems that have severe performance constraints (small devices, embedded hardware) or working under real time requirements. A feasible solution consists in creating systems that count with a repository of preconfigured solutions, which can be exchanged depending on the external conditions.

3.1.4 A note on coupling, modularity and automatic adaptation

Data fusion consists on making the best possible use of available information. With this purpose, data sources are sometimes arranged in complex structures capable of exploiting subtle shared synergies that, however, make the solution more dependent on specific sensor features, domain assumptions and other prior information.

We can find an illustrative example in fusion algorithms for GPS/INS navigation, which are categorized in the literature as loosely, tightly and ultra-tightly coupled solutions. The loosely coupled solution (also called "cascaded solution") is the most simple one: a navigation filter (as the UKF used in section 5.2.3) integrates GPS calculated locations with the acceleration and turning rates measures generated by the inertial sensor –this is, sensors operate separately, and their measures are considered individually. This solution has several drawbacks (Wendel and Trommer, 2004): GPS device needs at least four visible satellites in order to calculate a valid solution, leaving the inertial sensor unaided when the available constellation falls below that number. Also, most recursive filters assume uncorrelated errors between consecutive measures, but this is not true for the GPS.

Tight- and ultra-tight- coupled solutions (Petovello et al., 2007; Ravindra and Wang, 2005) fuse kinematic information from the INS sensor with the low level (pre-measure) data available at GPS device: the filtering algorithm processes the pseudoranges of individual satellites. If the number of visible satellites is not enough to produce a fix, it is possible to predict recently lost signals thanks to the inertial data, and generate a degraded GPS fix that still contains the information of several real satellites. The difference between tight and ultra-tight versions is that the first uses a separate tracking loop (filter) for each satellite, while the second funnels all satellites down a single instance of a more complex filter. Coupled solutions have some potential downsides: they require a higher implementation effort and their performance can be more difficult to characterize (require extensive and intensive testing).

Loosely coupled algorithms are, in general, easier to use in adaptive solutions. They can be described with simpler and smaller formal models, which are in turn easier to exploit. The interaction between data sources take place at a more general or higher abstraction level compared with tightly coupled solutions, making easier to integrate context information in the internal logic to improve results.

Building a fusion system automatically from individual components (as the prototype designed in chapter 5) is not an option for tight coupling schemes, and can even be dangerous for loosely coupled ones. It is difficult to infer the performance of a fusion system from a formal description of its components. In fact, current development of sensor fusion solutions relies on heavy testing and experimentation phases. An adaptive system prepared to deliver the best performance in real conditions would probably require a fine characterization of individual configurations, and then be subject to strict constraints on how components can be connected. In the end, it is a matter of *granularity*: define components as the smaller arrangement of logical fusion operations that can be decoupled from other components while keeping some features considered useful for the developed system.

Taking this consideration into account, it is possible to integrate tight-coupling components in an adaptive solution by defining them as monolithic blocks whose internal logics are not exposed in the formal model of the system. Context information can be used to take structural decisions as activating or deactivating the block, and also to refine its output through an adjacent loosely coupled layer that do some simple task as constraining the produced values.

3.2 System scope and requirements

The framework must satisfy the following requirements:

- Capable to host any kind of centralized sensor fusion solution.

- Simplify the design and implementation of flexible/adaptable sensor fusion solutions. The flexibility of a solution is related with the following features:
  - Robust against sensor outage/loss.
  - Can incorporate new sensors.
  - Supports sensors with features subject to change, as refresh rate or quality.
  - Can adapt the produced solution to satisfy different requirements.
  - Can adapt the processing scheme/parameterization to maximize the fitness of the solution.
- Allow the collection, storage and dissemination of context information with no restrictions on its type or abstraction level.
- Context consumers shall use context information as a finished product. The specific mechanisms for acquisition and management will be kept hidden to fusion algorithms and adaptation logics.
- Facilitate usage of context information for:
  - Relevance-based selection: prioritize data sources, data elements and algorithms according to their effective and potential benefit/usefulness.
  - Automatic reconfiguration: use context to determine changes in the fusion solution leading to a better performance. This information can be used to add or remove elements present in the solution (sensors, algorithms), modify connections between elements, or change the internal parameterization of configurable elements..
  - Contextual fusion algorithms: provide relevant, up-to-date context to those algorithms that can use it to improve their products.
  - Contextual sensors: provide relevant, up-to-date context to those sensors or sensing sets that can use it to improve their products.

Some aspects and applications are left out of the scope of this proposal. The design will take them into account as possible as long as it does not interfere with any of the requirements enumerated above.

- *Critical and real time systems:* the framework is not oriented to guaranteeing continuity or quality of service. Real time functioning is not assured.
- *Human-in-the-loop:* the framework is oriented to pure automatic sensor fusion applications. No mechanisms have been created to allow neither human intervention or usage of soft data –humans as sensors.

- *Distributed systems:* it is possible, using the proposed framework, to design solutions where some sensors or parts of the fusion process take place in a distributed environment. Nonetheless, the framework does not include mechanisms oriented to distributed fusion, such as those related with resource discovery, communication load/bottlenecks or integrity/consistency checks.

3.3 Proposal

This section contains a detailed description of the proposed architecture, depicted in Figure 3.1. It represents a complete rework over the fundamental concepts explored in early works of this thesis (Martí et al., 2011a,c).

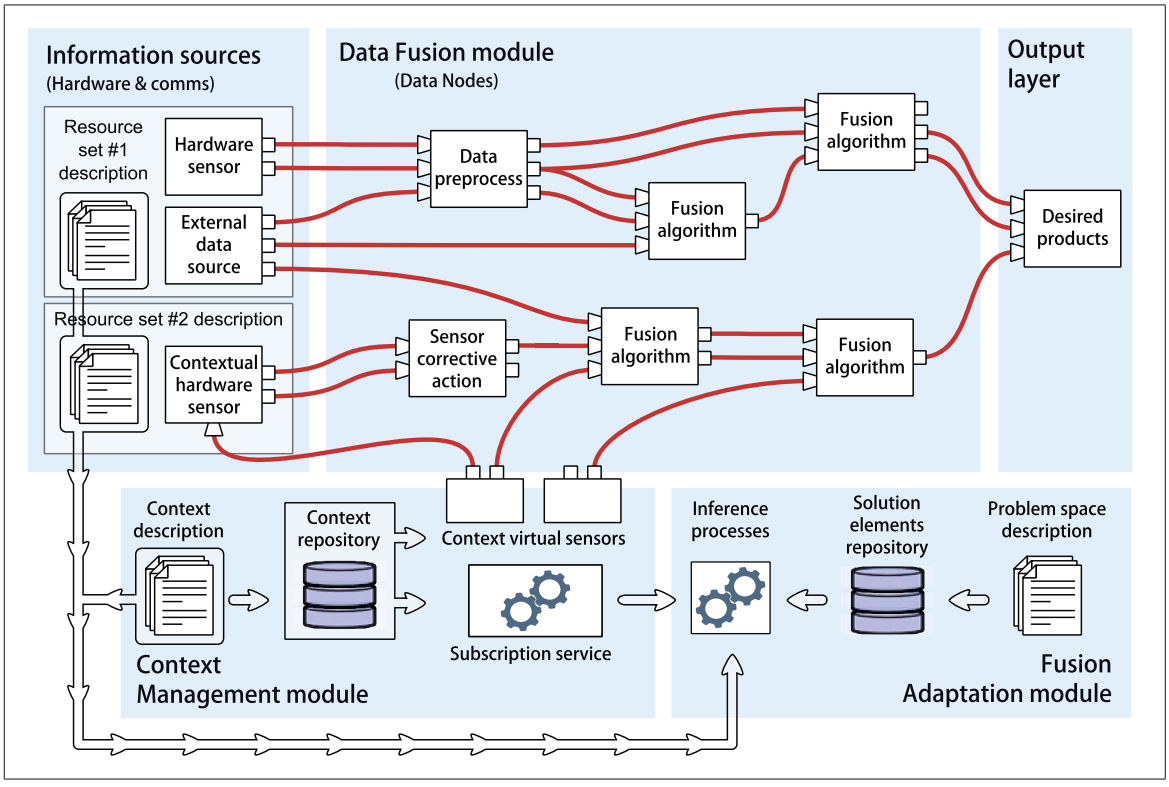


Figure 3.1: Architecture of the context-based adaptive sensor fusion system

The diagram shows four principal components. Two of them, Information Sources and Data Fusion modules, compose the core elements of a regular sensor fusion solution: the sensors that provide raw data to be fused, and the algorithms that combine and process that data to generate a fused output. The other two have been created to accomplish the goals of this work. The Fusion Adaptation Module configures the fusion solution that results in optimal results, based on the desired fusion products, the features of the involved components, and relevant

contextual information. It works over the components in the first two modules, selecting which are active, their configuration and how they are connected. Finally, the Context Adaptation Module is a centralized repository of contextual information. It is in charge of storing context in the correct format, and includes the mechanisms used to disseminate context to elements using it.

These modules have to be combined to create adaptable sensor fusion systems, where context information is extensively used. In order to make this possible, we had to make our way through the following steps:

- Define a formal system for characterizing the problem space: domain, constraints, data/information types, sensors and algorithms.
- Define a formal system for describing the relevant context information and how it affects the problem.
- Design a procedure that, given a problem specification, determines how to solve it using some available tools. The solution has to be the best possible one, according to some criteria expressed in the Problem Space Characterization.
- Combine the aforementioned elements in a generic framework for developing adaptive sensor fusion problems

The next subsections address these questions in detail.

3.3.1 Fusion solution components: Data Nodes

The main components of a fusion solution are the sources providing information and the algorithms that transform information (that we call "fusion nodes"). These components are combined in an arrangement that generates the desired fused output.

In the way of creating modular fusion solutions, coupling between components must be reduced to allow the composition of different functional combinations of sensors and algorithms. This does not mean renouncing to the benefits of specialized solutions over well characterized data, but designing components that can also work when their working conditions are not met in spite of showing a lower performance. With this purpose in mind, we decided to build the architecture around two important concepts of software design: virtual sensors and widgets.

- *Virtual Sensors:* described in (Indulska and Sutton, 2003) as software components that provide a certain information, instead of hardware devices. In this thesis we understand virtual sensors as a software component that provide data through an homogeneous interface, hiding the specific mechanisms through which it is produced.

We find an example in the Android OS API, that defines sensors for counting steps and provide the gravity direction vector (figure 3.2). These virtual sensors can be backed by hardware implementation (manufacturers are encouraged to implement step counting on hardware to save battery) or be derived/composed by other information, but this fact is transparent to the developer.

- *Widgets:* can be seen as reusable building blocks that encapsulate a functionality (Salber et al., 1999)(figure 3.3). A widget exposes a well defined interface that can be used to control it, feed the required input information and extract the produced outputs. Widgets foster modularity and reusability. Modularity because splits the functionality in independent blocks that can be replaced or modified, so that the behavior of the system is adapted without requiring further changes in any of its surrounding elements. Software encapsulated in a widget is more likely to be reused in different systems because it depends only on input data without further functional dependencies. In the proposed architecture, this last aspect can be changed when it is convenient for the purpose of the sensor fusion system. For more details about this, read sections 3.3.2.1 and 3.3.3.3.

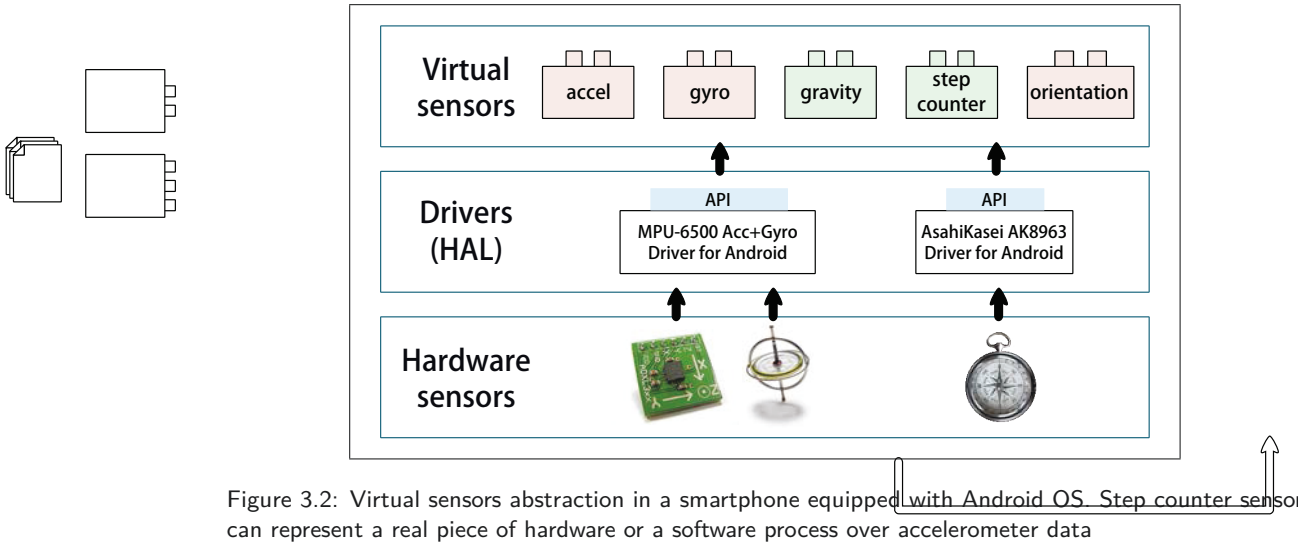


Figure 3.2: Virtual sensors abstraction in a smartphone equipped with Android OS. Step counter sensor can represent a real piece of hardware or a software process over accelerometer data

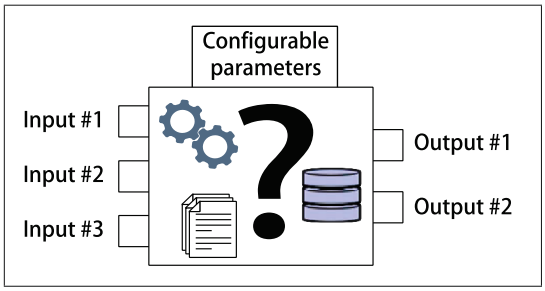


Figure 3.3: A widget encapsulates a functionality minimizing the parts of it exposed to the outer world.

Next sections combine these abstractions to define the Data Node entity. Data Nodes represent the basic components of a fusion solution: sources of information and fusion algorithms. It is important to notice that both categories, information sources and fusion nodes, are not exclusive in the sense that some elements can be in both at the same time. Contextual sensors –sensors that can receive additional information about the context to enhance their results– can be considered sources of information and fusion nodes at the same time.

These abstractions are fundamental for our proposal. They are used with two purposes: simplify the task of providing a formal and normalized description of the fusion system components, and enabling automatic adaptation of the sensor fusion solution. Both aspects are explored in next subsections and cases of use. Widget-based architectures are usually managed by a centralized control component that acts as repository of available widgets and existing links between components. In our case, the central component is the Fusion Adaptation Module, described later in section 3.3.3.3.

3.3.1.1 Abstract information sources

A Sensor Fusion process, as defined in this thesis, makes use of sensory information and other static or dynamic sources of data. This work propose to combine widget and virtual sensor abstractions to achieve a unified description of information sources that can offer homogeneous access to data. The combination of these two concepts have the following advantages:

- Consumers do not need to implement specific access methods for each type of source. Virtual sensors share a common and simple interface.
- The fusion logic is isolated from sensor management details that are out of its scope, but data sensors still expose a configuration interface that can be used by the fusion system when it has an impact on performance or quality. This configuration interface can be adapted to show virtual parameters directly related with the domain of the problem.
- Equivalent data sources are exchangeable at runtime. The change is transparent for consumers, that receive an uninterrupted flow of incoming data.
- The same fusion system can be used with simulated, recorded and real time data. It only requires implementing equivalent virtual sensors that adapt data from each source –real sensors, recording file or simulation software.
- Basic signal pre-process can be built into the virtual sensor.

The process to abstract each type of information source can be different, although they are basically a translation process. Let us give a brief description of the most relevant considerations for each of them:

Hardware sensors

The virtualization of a hardware sensor can be as simple as writing a wrapper that translates the API exposed by the driver to the common interface of virtual sensors. Depending on the sensor, it could take care of details as synchronous/asynchronous access to devices or communication channels. It can also implement more complex processes, as a combination of buffering and markers for granting several consumers simultaneous access to stream-like information sources.

Recorded and simulated sensors

Using information that is stored in a file or has been synthesized requires additional logic regarding time simulation. This aspect is responsibility of the implementation, which must decide how to synchronize components, simulation speed and other details. An example can be found in the last selected application, chapter 4, where all time related operations rely on a custom timing library that keeps processes synchronized and allows to simulate accelerated time frames for agile experimentation.

Context

In this proposal, context information is a mix of static and dynamic knowledge with different abstraction levels. It will be difficult or even impossible to use the virtual sensor approach to give access to some pieces of context knowledge, depending on how they are represented. The Context Management module (section 3.3.3.2) is responsible of deciding, based on the description of context, which variables will be accessible as virtual sensors. As a general rule, this will be possible when the value of the variable can be contextualized implicitly. For example, mapped information does not meet this requirement, since it is necessary to specify the location whose value we are interested in. The current activity of a device user or the weather at a known location, however, can be abstracted as a virtual sensor.

3.3.1.2 Fusion nodes

We define as "fusion node" any (stateless or stateful) operation that receives an input and produces an output, and can be used as part of the fusion system. This applies to actual fusion algorithms, but also to other operations as unit conversion, external calibration tools for the sensors and even logging/displaying tools. The difference between a fusion node and a source of information is that the latter does not require any input to produce a real, meaningful output, while the first defines its functionality based on the received input.

Most popular sensor fusion algorithms are easy to translate into the widget abstraction, since they fit naturally the concept of a black box that takes some input data and produces and output. Implementations need to care about synchronization and timing issues regarding

the data entering and exiting the widget. This is the same consideration exposed in the previous section 3.3.1.1. Virtual sensors do also follow the widget style, allowing to apply an homogeneous management for all the components of the fusion system. Furthermore, virtual sensors that feed on other processed data or that need a contextual input can define their inputs just as any fusion algorithm.

3.3.2 Modeling the components of the problem and the solution

A fusion systems with automatic adaptation features needs full knowledge of the problem to be solved. This knowledge is composed by the problem space –a concept including the domain of application, the components of the solution (explored in the previous section) and the relationships between these elements– and, in our case, contextual information. Specific sensor fusion systems tend to embed adaptation-related knowledge in the structure of the algorithms, but this has some has drawbacks: implicit logics can be difficult to interpret for a human reader, its modification requires changing the code, and reduces the reusability of the code in other solutions.

This theses defends explicit above embedded knowledge. We have proposed the use of ontologies to describe the information required by Context Management (section 3.3.3.2) and Fusion Adaptation (section 3.3.3.3) modules. According to the proposed architecture (see figure 3.1), a context-aware sensor fusion system needs the following elements described:

- Problem-space (in the Fusion Adaptation module): basic notions about problem domain, components of the solution.
- Resource sets: sources of information, generated data, features, requirements.
- Context: description of relevant context variables. Constraints, relationships.

This section identifies the information that has to be modeled and some important considerations. We also propose, as an example, a minimal ontology that can be used to describe the basic aspects of almost any adaptive sensor fusion system.

Figure 3.1 shows that the three descriptions are interpreted by the inference processes of the Fusion Adaptation module. Our design presents them as separate components in an attempt to increase system modularity, but they are just different aspects of a single language. At the time of designing a sensor fusion solution within the proposed framework, it is necessary to ensure that the different components are consistent, with special attention to the parts they may have in common. For more details on this question, check section 3.3.4.

3.3.2.1 Problem-space description

The problem space description, highlighted in figure 3.4, describes the domain of the problem to be solved, the basic components that can be used in the solution, how solution components are related between them and with the domain of the problem. It can also include other useful semantics.

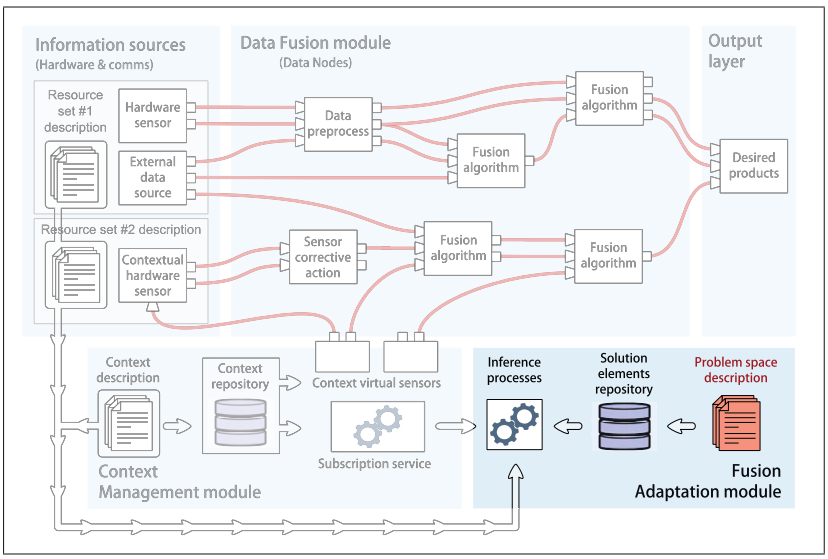


Figure 3.4: Problem-Space description is a basic component of the Fusion Adaptation Module.

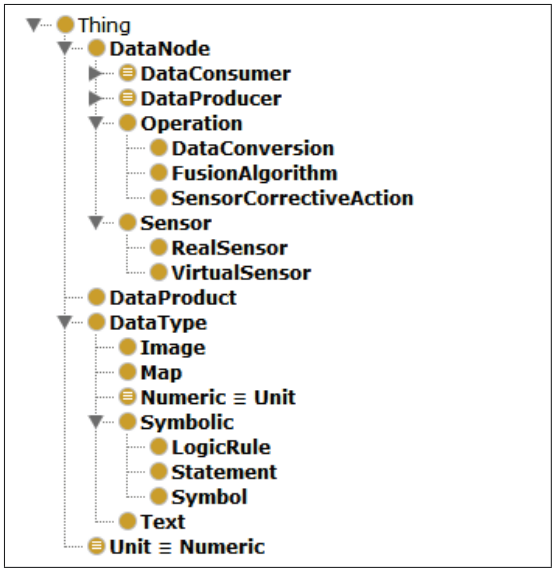


Figure 3.5: Hierarchy of classes in the problem-space description ontology

In its most basic form, the domain of the problem can be defined as the types of information (physical magnitudes, declarative knowledge) that are necessary for its formulation and for its

solution. This definition can include features of interest as the accuracy, or the relationship between units. Also, this ontology has to describe the available fusion/data nodes in the central repository of elements. A proper description of data nodes require at least defining the data types managed by the application, their features, and the additional features of data nodes.

Next, we will develop a basic Problem-space description ontology using OWL language that is applicable to most generic sensor fusion problems. This ontology restricts problem description to the types of data involved in the problem. Solution elements will be related with the domain of the problem through the types of their input and output data –this is, this problem is seen just as a mere transformation of data. Designers are responsible to use ontologies to define the problem from the most useful point of view. In a problem like (Cohen and Edan, 2008) where several algorithms for multisensor fusion are evaluated in parallel and the best one is selected for the solution, the description of the problem space would include the criteria used to evaluate performance. The work is focused in sensors whose measures can be assimilated to grid-like maps, so this concept should appear in the description of the problem, along with other issues as the size of the grid.

According to the introduction to the language in section 3.1.2.2, an OWL ontology defines relevant concepts as classes, uses object properties for expressing relations between elements, and individuals are attributed through datatype properties. Let us define these three types of elements.

Classes: figure3.5 shows the hierarchy of classes used for this work. The top elements are the most important concepts, while the subcategories are defined for further inference processes. Let us describe the top classes:

- DataNode class are the basic building blocks of a fusion system. Since we are approaching the problem from the point of view of transforming information, including information sources as a subclass of DataNode helps to homogenize automatic reasoning processes. The presented figure show several subcategories: Sensors –only information sources considered here–, Data Conversion functions, Sensor Corrective actions and Fusion Algorithms. These subcategories can help defining constraints or inference processes, as "the roots of a fusion process have to be Sensors" or "a Fusion Process graph cannot contain loops, except when the loop involves a sensor corrective action".

- DataType parent class categorizes the type of information managed by the system. The figure show a further decomposition in Numeric, Image, Map, Symbolic and Text classes, that can be used to express additional facts or constraints, or just to help human users that have to interpret or edit the ontology. These classes will be populated with individuals that represents the actual data types used in the system.

- DataProduct are auxiliary entities for representing attributed data production or con-

sumption. In OWL ontologies, it is possible to use an Object Property to express that a Data Node "produces" information of a certain Data Type. However, this relation does not allow features as "Sensor Y produces data type Z at 5 Hz with high accuracy". Data Product is an instrumental entity that can be placed between both classes and be attributed with data properties that express the desired features. Thus, the above statement can be reflected as the composition of several statements as shown in table 3.1.

Table 3.1: Set of OWL statements equivalent to *Sensor Y produces data type Z at 5 Hz with high accuracy*

Domain	**Property**	**Range**
Sensor Y	produces	DataProduct Z
DataProduct Z	hasType	DataType Y
DataProduct Z	hasType	DataType Y
DataProduct Z	updateFrequencyInHz	Integer "5"
DataProduct Z	quality	Literal "High"

Object Properties relate pairs of individuals of the ontology. In the example, we are interested in two types of relations. in figure3.6, object properties have been classified according to their domain class: properties of data nodes and properties of data products.

The first category links the processing (data) nodes with the data products needed at the input and produced at the output. Remember that a data product is just a data type with additional attributes. It includes the relations "produces" and "consumes", expressing which Data Products are the inputs and outputs of Data Nodes. The property "preconditionedTo" is intended to describe arbitrary requirements for a Data Node being applicable, such as a certain Data Node being active.

The second category, properties of data products, contains the property "hasType" that relates a data product with a data type. Using OWL tools, we have defined the inverse relations "producedBy" and "consumedBy", that can be applied to data products.

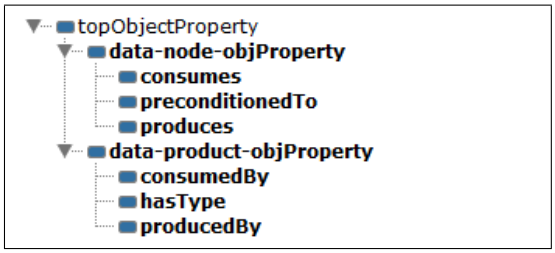


Figure 3.6: Object properties in the problem-space description ontology

Data Properties allow to add literal values to individuals, which are somehow equivalent to

the attributes in OOP. Data properties are very dependent of the application domain, so we do not provide generic contents in the proposal. Figure 3.7 shows an example based in the ground vehicle navigation scenario (see section 5.2.6), organized in three categories according to the domain class.

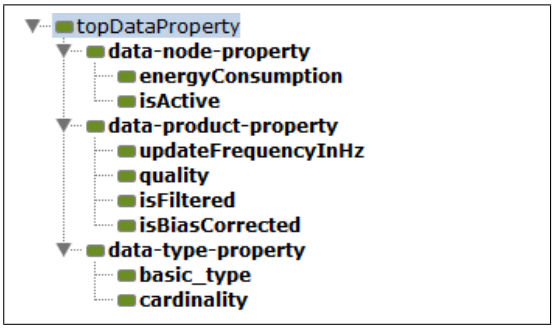


Figure 3.7: Data properties in the problem-space description ontology

3.3.2.2 Resource set description (information sources)

The resource set is conformed by all the information sources that can be used to solve a sensor fusion problem. In the proposed architecture, resources are organized in independent subsets that can be incorporated to or removed from the fusion process online. These subsets have to be described using a common language that can be understood by the Fusion Adaptation module. The description has to define the sources of information (virtual sensors) provided by the subset, features that are relevant for determining the applicability/suitability of a sensor in a particular problem, and the constraints on the use of the sensor. The last part can be based in context related concepts.

Depending on the problem domain, the description of a resource can cover the following aspects:

- Data produced by the resources: hierarchy of types, relevant morphological aspects, and others.
- Features of the data produced by the resources. They can be classified in static and dynamic features:
  - Static features are inherent to the resource and do not change over time, as the range of temperatures produced by a thermometer or the guaranteed bias stability of a gyroscope over time. This type of features can be described with datatype properties and predefined for each type of sensor according to the typical values in technical datasheets.

  - Dynamic features, on the other hand, describe observable qualities of the data that change over time. This includes quality metrics produced by sensors with self-assessment capabilities –confidence in the measure, data compression factor, value of automatically tuned parameters–, and a description of characteristic errors associated with a data product that are unknown but can be useful to take into account for the problem –bias of an accelerometer, relation between sensitivity and operation temperature.

    Dynamic features can involve a more complex modeling, defining new classes and creating object properties that indicate how the different elements relate.

- Description of configurable parameters, e.g. valid range of values, how changing the parameter affects data features. Depending on the sensor it will be possible to define an inverse schema, where the interface exposes the dynamic features that can be changed, and an intelligent controller tunes the configurable parameters to achieve them. This is a typical features of cameras that adapt aperture, sensitivity and exposure time (real control parameters) automatically to get images with the desired brightness, focus and stillness –a kind of virtual parameters.

- Operating conditions and other constraints indicating when the resource cannot be used or the data provided will not be useful. Operating conditions can refer to external circumstances, which may be related with context information either self-inferred or provided. In the case studied in section 5, the accelerometer of a smartphone is used to track the motion of a vehicle, but that information is only useful if the smartphone is resting in the vehicle. Such condition can be inferred using the readings of the accelerometer, but can also be reported by the smartphone if the screen is operative.

  An example of constraints not related with the context is a resource set that defines several virtual sensors based in different functioning modes of the same real device. In that case, it might be interesting to express that the virtual sensors are mutually exclusive: only one of them can work at the same time. This is the case of the windshield camera in an intelligent vehicle: a far focus is useful for detecting pedestrians and getting road images, while a near focus transforms the camera in a rain sensor that counts water droplets in the windshield.

For the example shown here, the description of resource sets is developed over the previous problem-space description. Sensors are defined as a subclass of DataNode, restricted to not take any data input –the ontology allows to express such constraint defining a Sensor as a subclass of the built-in type Restriction, that refers to datatype property "consumes" and type DataProduct defining an exact cardinality of zero–. The class Sensor uses the same other classes and properties defined in the problem space description, which can be used to describe fully-attributed individuals in a format understandable by Fusion Adaptation module.

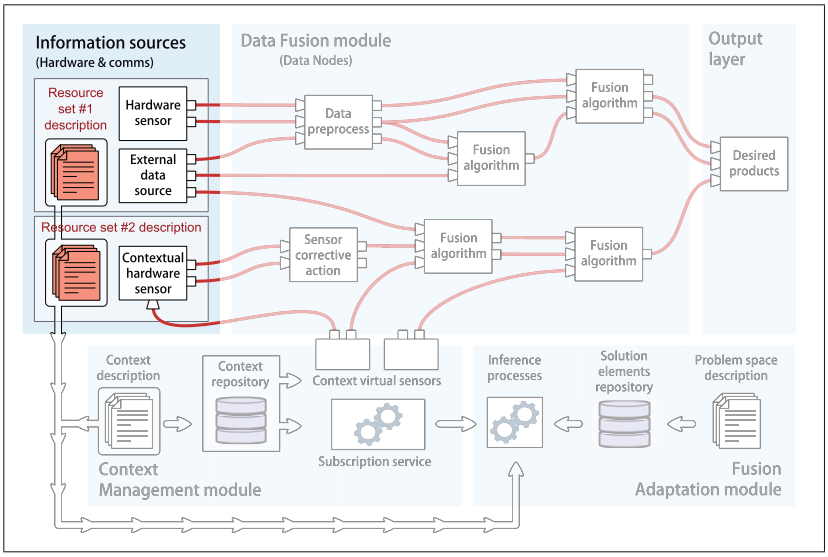


Figure 3.8: The resource description ontology, highlighted in the figure, has been integrated in our example with the description of the problem space. This way, concepts and properties can be reused to provide a homogeneous treatment of every Data Node in the system.

3.3.2.3 Context knowledge description

There are two different scenarios regarding context. In the first one, context description responds to the necessities and opportunities identified during the problem analysis/design phase. In those cases where the solution is created from scratch, we recommend the creation of a context knowledge representation specifically tailored for considered the domain, ample enough to allow extensions of the problem. If, on the other hand, context information is provided by an external system, we may have to reuse an existing contextual schema. In both cases, however, determining relevant context information is a very domain-specific task. It is difficult to provide a tentative taxonomy in this section, so we refer the reader to experimental sections.

The true power of context-aware sensor fusion systems comes from the ability to incorporate previously unknown contextual information and use it to improve the solution. In a system of limited reach as a sensor fusion solution, it can be unpractical to develop automatic processes that can discover the relation between unknown context elements and the variables of the problem. A good design choice is to design the description of the problem in such way that pieces of contextual knowledge can include in their description the effect they have on the problem or in the solution. For example, the description of a context-aware navigation system can include constraints on the speed of the vehicle, and the algorithms be designed to take into account arbitrary constraints. New context variables like traffic information can be automatically incorporated in the solution just by being annotated with those constraints.

3.3.3 Architectural components. Modules

Once the elements of the solution and the modeling languages have been explained, it is time to describe the modules of the proposed architecture. This section details their purpose, the specific functions they can be in charge of, how they are related with the rest of the modules, and specific questions regarding their implementation.

3.3.3.1 Information sources module

This module accepts two different implementations. The first option is to define a central control logic that manages the set of sensors and coordinates the addition, removal and/or configuration of components. Using this alternative, the central control is the only element in this module that can communicate with the Fusion Adaptation module. Among its advantages, we can cite a better decoupling of functionalities.

Another option consist on having independent resource modules that communicate freely with Fusion Adaptation module. This is the situation described in figure 3.1. This second option overloads Fusion adaptation module with extra management logic that could be considered out of its scope, but in small systems can simplify the decision of how to adapt the fusion solution, and making that decision effective. Both scenarios support the same functionality. Since the architecture is designed with centralized systems in mind, it is reduced to preferring one code organization schema above the other.

The following list summarizes the functionalities this module is responsible of. We are assuming it is implemented according to the first explained option, a central control logic that manages the set of sensors. In other case, some of this functionalities will be transferred to the Fusion Adaptation module:

- Discover resource sets. This process is not necessary in those sensor fusion applications that rely on a predefined set of on-board/integrated sensors. In other cases, it can be designed as an active process –search and query for resources– or as a reactive process that responds to offerings and announcements from collaborative resource sets. Resource discovery has sense in distributed scenarios, where the appearing/dissapearing resource sets are located in different platforms.

- Read resource set descriptions. Integrate description and components in the system. This process involves the Fusion Adaptation module regardless how this module is implemented, since it will need the description of the resources for managing the fusion solution.

- Check resource availability and status. Detecting failures and outages is likely to require an active control. An option is to implement a simple data quality control process.

- Notify relevant changes to the Fusion Adaptation module. This includes addition and lost of sensors and changes in their status.

Sensor configuration management is considered to be out of the scope of this module. That task has a direct relation with the fusion process, and is assigned to the Fusion Adaptation module. We can see that the central control logic, in the case it is implemented in the Information sources module, does not isolate the actual resources from the Fusion Adaptation module as a layered or a true modular architecture would do. We permit this leaky abstraction to reduce the complexity of potential implementations.

3.3.3.2 Context management module

Context management module is in charge of keeping an updated record of the context relevant to the fusion system. It is also in charge of satisfying the needs of contextual information of the different components of the fusion system, in a suitable format and through adequate channels. The actual functionality of the context management module will depend on the design of the solution. The following list includes some possible functions:

- Manage context information acquisition process. This includes keeping track of information sources and their state.
- Offer access to context variables through suitable mechanisms.
- Evolve/adapt context model and its relation with problem-space description, as new sources of context with a suitable description are integrated in the system.
- Keep context information updated.
- Keep context information repository in a consistent state. Inconsistencies can have its origin in different sensors contributing to the same contextual variable, and redundancies or overlapped meanings in context representation.
- Control granularity, when applicable. This includes temporal granularity (e.g. the update frequency of some variables) and spatial granularity (resolution of mapped data) among other factors.

We have defined two channels for accessing the context information maintained by this module: virtual sensors and subscription service.

Virtual sensors are linked to the context variable of interest. They can be integrated into the fusion solution by the Fusion Adaptation Module as any other data node. They can be read on demand or emit the corresponding value at a prefixed rate.

Subscription service notifies updates in context variables to the interested entities. In our case, the Fusion Adaptation Module can integrate these updates into its inference processes to improve the selected fusion solution. Both options are well known programming patterns with a simple implementation.

In this proposal, determining context relevance is responsibility of the Fusion Adaptation module, which relies in the description of the problem space. This factor limits the flexibility of the solution for incorporating new, unforeseen knowledge, although it can be partially alleviated if incoming context information is annotated with constraints or those variables of the problem it affects, as we explained in section 3.3.2.3. When the domain of the problem is more complex, the context database is large or poorly structured, and context mixes a wide range of abstraction levels –a normal scenario in full-scale information fusion systems– this solution is not applicable.

In those cases, it might be useful to develop an automatic strategy for determining context relevance. As an example, (Llinas, 2010) propose a query-based middleware that can infer and return the context information that is related with a situation, region or entity of interest.

3.3.3.3 Fusion adaptation module

The Fusion Adaptation module is the cornerstone of the architecture. This central component combines all available information (problem space description, information sources and context knowledge) to orchestrate the fusion process.

The proposed architecture accepts a large number of implementations, but the typical work cycle of a Fusion Adaptation module prepared to exploit the full potential of the proposed architecture comprises the following steps:

- Initialization:
  - Load problem space specification.
  - Initial load of context knowledge and resource set specification.
  - Subscribe to relevant context sources.
  - Determine initial solution and configure it using repository of elements, virtual sensors from resource set and virtual sensors provided by context management module.
- Reactive adaptation logic: triggered on context/resource/goal set events.
  - Context event: the adaptation logic is notified about a change in the value of a relevant context variable, or new contextual knowledge is added.

  - Resource event: added/removed sensor information is updated in the problem space description. If a sensor currently in use is lost or if a new sensor is added, adaptation manager will consider switching to a different configuration. For other events as –data features have changed, data quality has detected an anomaly– the actions range from changing sensor parameterization to changing the algorithms or the used sensors.
  - Goal event: the goals of the system (the desired data products) have changed. Goals can change requesting different products, or preferring quality in some variables above others.

- Proactive adaptation logic: involves monitoring the results of the fusion system to determine if there is a problem with any of the components of the fusion. Another approach is to routinely evaluate several alternative solutions in parallel and switch to the best one.

3.3.4 Designing a sensor fusion solution within this proposal. Workflow

The structure of the architecture, together with the assumptions made and the selected approach to a context-aware sensor fusion solution, enforce a particular order in the creation of such a system. In particular, the design will be strongly guided by the explicit modeling of the problem space, context and resources using ontologies.

Previous section 3.3.2 explained how our proposal is to build a unique description language that is composed by several parts (context, sensors, problem space) that can be plugged together, although each part responds to a different aspect of the problem. If we assume starting the design from zero (in many cases, it will be possible to reuse domain-specific languages), this is a tentative order for the procedure:

- Identify the components of the problem (including sensors) and its solution, and draft suitable Problem Space and Resource Set description languages. When possible, contemplate alternative solutions and algorithms that can work with different sets of sensors.

- Identify relevant context and how it affects the fusion solution. Draft the Context Information description language.

- Define how can the different aspects/parts of that context be used on each part of the fusion solution, both for adaptation and to improve the fusion processes. This analysis can be used to refine the representational aspect of context information, which in turn can affect the design of Problem Space description language.

- Define the capabilities and responsibilities of information sources module according to the domain of the problem.

- Design the adaptation logic. Purpose, desired capabilities, indicators/metrics guiding the process, conditions that affect or trigger the adaptation process, algorithms used to calculate solutions.

The procedure can be used as presented here to include posterior modifications into the problem and its solution.

3.4 Conclusions

The proposed framework is well suited for centralized, autonomous sensor fusion applications with high adaptive features that are not part of critical systems. Apart from depicting a general architecture for creating such systems, we have drafted some tools and mechanisms that can help in the creation of the involved components, and in gluing all the parts together.

One important contribution regards the definition and management of context information. We have proposed a standardized representation along with minimal metadata necessary to determine its relevance and to ensure that other components will be able to manage it.

3.4.1 Review of requirements

The following list reviews the requirement list proposed in section 3.2:

- This framework is, in principle, capable to host any kind of centralized sensor fusion solution. The proposed architecture and tools do not impose any restriction related with information types or processes. Next chapters 4 and 5 provide limited evidential support for this statement, although further scenarios should be proposed to ensure it.

- We think that both the architecture of the system and the design procedure drafted along this chapter do actually simplify the design and implementation of flexible/adaptable sensor fusion solutions. Find next a review of the desired capabilities that are supported:

  - Robustness against sensor outage/loss: the Fusion Adaptation Module can include the logics in charge of reacting against these problems. Thanks to the homogeneous access to sensors, sensor functional redundancy can be used to achieve seamless functioning under outage or loss circumstances.

  - Can incorporate new sensors: formal description of sensor sets through a predefined ontology allow their integration into the fusion process as long as the Fusion Adaptation Module is prepared to work with data they provide.

  - Supports sensors with features subject to change, as refresh rate or quality: as long as those features can be expressed in the corresponding Resource Set ontology and interpreted by the Fusion Adaptation Module (if relevant), they are transparent to the rest of the architecture. Fusion algorithms must support input data with varying features, but this depends on their implementation (falls out of the reach of this proposal).
  - Fusion Adaptation Module can adapt the produced solution to satisfy different requirements, and also adapt the processing scheme/parameterization to maximize the fitness of the solution.

- The Context Management module allows the collection and storage of context information with no restrictions on its type or abstraction level, thanks to the use of ontologies. This information can be disseminated using the proposed mechanisms: virtual sensors and subscription service.

- Context consumers receive context information as a finished product. The specific mechanisms for acquisition and management are implemented by the original data sources and the Context Management module, out of the scope of fusion algorithms and adaptation logics.

- Context information can be used for:

  - Relevance-based selection: context can be used by the Fusion Adaptation module to decide how appropriate are algorithms and data sources on each case.
  - Automatic reconfiguration: previous argument applies here.
  - Contextual fusion algorithms: algorithms can be defined as having an additional data inputs for specific context variables. This context is provided through a virtual sensor made available by Context Management module.
  - Contextual sensors: previous explanation applies here substituting fusion algorithms for sensors, because both use the Widget abstraction that makes it possible.

3.4.2 Domain of application

This work is focused, as stated before, into low level data fusion but incorporating context information. Among its features, we can cite:

- Autonomous functioning: this framework is oriented to the design of fusion applications that can be used as a black box, without requiring user actions other than specifying the desired output.

- Flexibility: supports online addition/removal of components, including sensors and processing algorithms.
- Graceful degradation: when properly used, the system can react to unexpected problems with information sources and change to alternative solutions that minimize the impact in functionality and quality.
- As a design principle, the fusion logic is kept totally isolated from context and configuration logics. This has two benefits: improves the reusability of fusion algorithms in other fusion systems, and facilitates migrating existing fusion solutions into the proposed framework.

3.4.3 Extensibility

Current work can be extended in two different directions: abstraction level and processing topologies. Our proposal is restricted to JDL levels 0-1. This imposes restrictions in the abstraction level of the involved data and, in consequence, simplifies the formalization work described in this chapter. Extending the proposal to higher levels (2-3) has the following implications:

- Problem-space description is relatively easy at current levels. It would be necessary to define a formal representation for the inferences and data types managed at fusion levels 2-3. The differences at this level are more related with semantics than with pure format. Ontologies represent an appropriate tool for this task, but such descriptions will be less generalizable (more problem-specific) than at lower levels.
- The fusion adaptation logics will grow in complexity. This will most surely require further decomposition/detail of the module, and probably the redefinition of some data/command flows.
- Representing context information and determining its relevance will be harder. The same considerations written for the problem-space description apply here. Also, since the same piece of context information can have different representation depending on the abstraction level or the interest of a potential consumer, the framework will need reasoning processes able to translate to equivalent statements at different abstraction levels.

This proposal, at current stage, can be used for building a network of cooperative fusion systems. Our work (Martí et al., 2011a) suggest a multi-agent approach, where individual fusion systems are built into autonomous software agents that interact with other agents in the outer world. Software agents can offer and ask for services, namely raw and fused data. The work developed in this dissertation provides the following facilities:

- A fusion system that can be asked for a particular fusion product, and automatically configures itself for that purpose.
- A fusion system that can determine the data and features required to produce a particular fusion product.
- A fusion system that can incorporate additional sources of information to improve the quality of its output products. Virtual sensor abstraction simplifies the process of integrating data from external sources.
- The basis for a common language that allows interaction between collaborating systems.

4

Application to maritime surveillance scenario: Design

This chapter explains the application of our proposal to a large scale maritime surveillance scenario covering open sea with highly populated areas such as harbors and anchorage zones. The system is composed by Automatic Identification System (AIS) stations and a set of coastal radars with partially overlapped coverage areas, that can detect vessels up to several tens of kilometers far from them. Radar contacts must be fused with the self-reported AIS messages to build an accurate description of the vessels in the area. This task can be challenging in complex scenarios as the densely populated harbors.

The operation is monitored from a central control room. Each sensor is assigned to a human operator that in principle is responsible of adapting its functioning parameters manually to optimize the output. The challenge in this scenario is to create a system that makes the best possible use of available information, while being robust against unexpected problems and changes in the configuration of the environment. Adaptability is a key factor for accomplishing both goals: exploiting information optimally can be achieved by adapting the employed algorithms and parameters, and robustness can be improved by minimizing the impact of incidences related with the sources of information. Our proposal fits the needs of this project. Let us define the requirements of the system to be created.

4.1 Scenario requirements

The maritime surveillance system must satisfy the following requirements:

- It must not be limited in size of covered area.
- It must be capable to manage several tens of radars and AIS stations.

- It must be capable to process in real time at least 1000 vessels, detected by whichever combination of sensors.
- Sensors can be added or remove from the process online. Sensor configuration can be changed externally, and the system must react accordingly to keep the quality of the fusion products.
- Fusion solution has to be configurable online (while the system operates). Configuration includes:
  - Change fusion algorithms and their parameterization.
  - Changes can be applied over limited spatial regions, to improve the results in conflictive areas without affecting its performance in the rest of the coverage. This means that the system must be able to operate applying a heterogeneous set of algorithms and parameters simultaneously to the observed zone.
- The fusion system has to generate indicators of the quality of the fusion, including at least:
  - Number of tracks, coasted tracks (not updated in the last cycle).
  - Number of created and deleted tracks (per cycle), including unexpected deletions, i.e. losing a track within sensor coverage.
  - Number of false radar detections (cannot be associated with an existing track)
  - Residual of the filters used to track vessels.

4.2 Initial fusion solution design

Based on the specification, we proposed the processing scheme shown in figure 4.1, where each sensor is assigned to a tracker that generates a local output. Local trackers send their interpretation of the situation to a central tracker (*Global Tracker* in advance) that combines them into the set of tracks that best describes the situation.

The decision of generating local solutions for each sensor that are combined later is suboptimal. However, these outputs can be used by the operators to monitor the process and decide how to adapt the parameterization of their sensors. Also, the fusion logic is better modularized using this approach: local trackers take care of the usual tracking process, that involves data preprocessing, solving the association problem and obtaining position/course/speed estimates for the tracks. The global tracker decides which local tracks represent the same vessel, and combines them to obtain a better estimation.

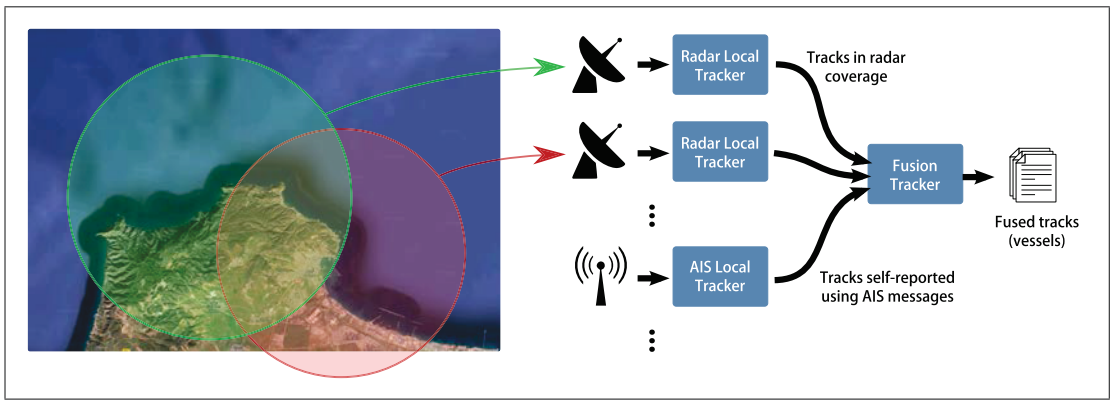


Figure 4.1: Selected processing scheme for the solution. Each sensor is processed individually, and local results are fused together by a global tracker.

4.2.1 Architectural solution

Following the recommendation of this thesis' proposal, every component of the fusion system has to be abstracted as a Widget. We went one step forward and designed a message-based fusion system: pieces of information are considered to be messages that circulate between components. Every component of the solution is defined as a message processor capable of receiving, interpreting and sending messages.

The proposal places the adaptation logics of the system in a central control component. In this scenario, adaptation comes from an external source –human operators–, but we will prepare our solution to switch to a fully automated schema when necessary. The Fusion Adaptation module, thus, is in charge of connecting the components that will work autonomously afterwards and will be configured according to messages passing through the system. The adaptability of the solution includes adding or removing local trackers (if the availability of associated sensors change). However, each tracker follows a fixed processing scheme that cannot be changed, as illustrated in figure 4.2. Blue boxes aligned over the arrow represent the fixed steps of the fusion cycle, and red boxes represent configurable components of those steps. As an example, we have implemented three association strategies that can be used indistinctly: Munkres (or Hungarian algorithm), Probabilistic Data Association (PDA) and Joint Probabilistic Data Association (JPDA) algorithms.

4.2.2 Message processors (Data Nodes)

The message passing architecture allows simplifies the interface between components, and give an homogeneous treatment to the different types of messages –quality metrics, tracks, sensors measures, configuration commands. The interface of a message processor is reduced to two functions: (a) connect its output to other message processor, and (b) accept an incoming

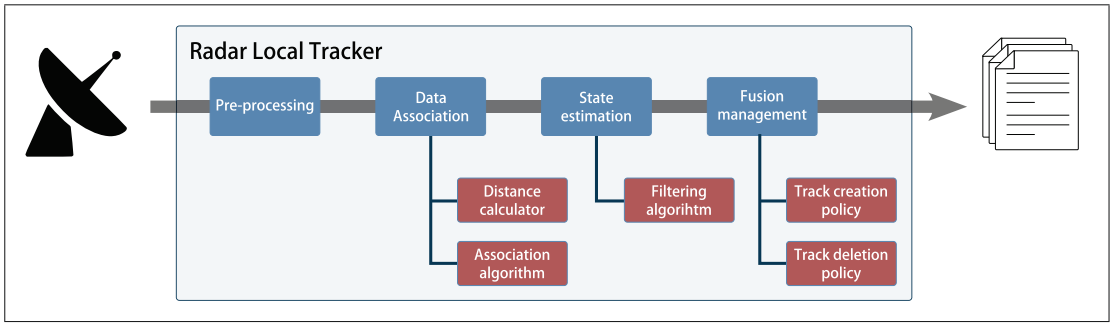


Figure 4.2: Selected processing scheme for the solution. Each sensor is processed individually, and local results are fused together by a global tracker.

message.

There are four types of message processor, depending on how they process and generate data:

- Synchronous: a message is processed as soon as it is received. The same thread does the acquisition and the processing tasks. This mode is not adequate for heavy tasks, because the message processor remains blocked until the job is finished and will not respond to other incoming messages.

- Asynchronous: incoming messages are queued on an internal buffer. A worker thread processes the messages as soon as possible following a predefined policy as FIFO or priority based.

- Event-triggered: messages are queued as in the asynchronous mode, but not processed until a certain activation message is received. In this moment, all the messages will be processed. This is useful for tasks that require a batch processing scheme, e.g. the fusion cycle of a radar local tracker, which needs to consider all radar contacts at the same time to solve the association problem, and is triggered when radar antenna has completed a rotation.

- Time based (or cyclic): buffered messages are processed when a timer expires. Timer is reset right after expiring.

These categories apply also for the output of the message processor. Generated messages can be sent synchronously, or being buffered to be sent as soon as possible, on a certain event or in a cyclic fashion.

One additional advantage of this modular and asynchronous design is that it exploits the power of multiprocessor architectures.

4.2.3 Algorithms to implement

For a first version, the following alternative algorithms have been selected for implementation:

- Distance calculation for the association process: one algorithm per type of tracker.
- Data association: Munkres, PDA, JPDA.
- Filtering algorithm: Kalman Filter, Interactive Multiple Model Filter (IMM).
- Track creation and deletion policies: N-out-of-M criterion, AIS time-based deletion criterion.
- Global track recombination policy: custom algorithm based on client specification.
- Global track component extraction policy: custom algorithm based on client specification.
- Global track state estimation: Gaussian mix of components, priority-based.

4.2.4 Zonal splitter for spatial configuration

Spatial configuration involves applying different algorithms to the elements of the fusion process depending on their geographical location. Assuming a message-passing architecture, zonal configuration can be seen as a flow control problem where input data is split according to its location. Each branch sends the elements of a particular zone to the corresponding processing node, that applies a particular fusion algorithm with the selected parameterization. Figure 4.3 describes this process. Left part shows the problem from the point of view of the architectural components: zonal splitter is configured to assign a processing node (and its complementary configuration profile) to different geographic zones. Right side of the figure shows the data flow view of the problem, where input data is split before being forwarded to actual processing nodes, and results are combined into an output that is consistent with the structure of the original input.

The proposed mechanism makes zonal configuration transparent to precedent and subsequent processes, that do not need to take into account if the node they are feeding or are fed by is a zonal splitter or a regular algorithm. The same can be stated about the underlying algorithms used by the zonal splitter, that do not need to prepare their input or output in a way that is different from the regular use.

Zonal splitter logic, however, has to know how to combine outputs in the case the internal process requires it. For example, a zonal splitter for association algorithm has to assemble the generated association submatrices into a single matrix that will be sent as a message. However, tracks are represented as individual messages, so that a zonal configuration that affects the

filtering algorithm just have to forward the updated output tracks without having to combine partial outputs.

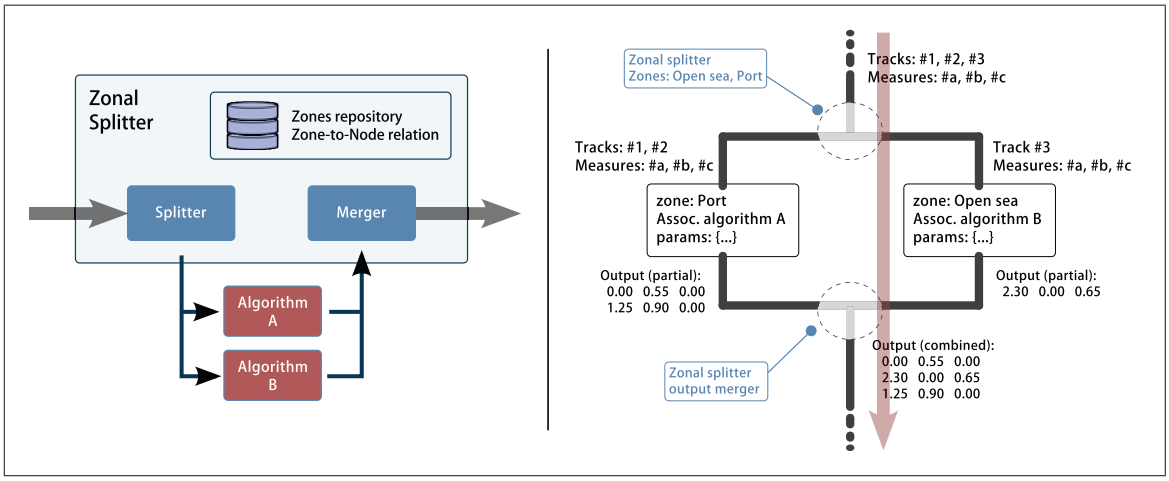


Figure 4.3: Zonal splitter processor divides input data in chunks (according to geographical location) that are processed using a different algorithm/configuration. Partial results are merged back to compose the output.

4.2.5 Generating fusion quality metrics

Fusion quality metrics are calculated from individual events of interest (as creating or deleting a track) or as statistics over some data variable (as the average filtering residual, that requires the residual of each track in the system). The message-passing architecture allow to generate events or individual data elements at the place of the fusion system where they are easier to calculate, and send them wrapped as messages through the network. A special message processor gathers them and calculates the related metrics.

4.3 Describing elements of the problem

We are following here the procedure described in section 3.3.4, that models first the problem space and the elements of the solution.

4.3.1 Problem space and sensors

We have used Protege to create a simple ontology that describes the domain of the problem and the elements of the solution. It is composed of a total of 50 classes, organized in 4 conceptual groups –that are also classes:

- Domain: elements belonging to the real world, physically or conceptually. Contains the physical sensor stations (radar and AIS), a single observable entity Vessel and the geometry-related notions Area and Point. These classes are enough to define the problem in its current version, and can be easily expanded later if needed.

- Message Processor: the same message processors identified during the design phase of the initial solution. This category comprises actual fusion elements as trackers and virtual sensors, as well as other auxiliary tools e.g. for visualizing the generated tracks, or for saving data to files.

- Message: entities or pieces of information that circulate between processors. They are equivalent to the DataType and DataProduct classes used in the example at section 3.3.2.1. Based on the initial analysis of the problem, we have identified as messages the inputs of the problem (RadarPlot, AISPlot and ConfigurationMessage), the outputs (GlobalTrack as the final product of the fusion process), intermediate fusion products as the result of the local tracking (RadarTrack and AISTrack) and the information related with quality metrics, as identified in section 4.2.5 (QualityEvent and QualityMetric).

- Algorithm: configurable parts of the tracking system. Trackers, as shown in figure 4.2, define a structure with "slots" where fusion algorithms are plugged in. We have created a class for each slot that is populated with the algorithms that fulfill that function. For example, the class AssociationAlgorithm is parent of Munkres, PDA and JPDA classes.

Algorithm can be implemented as Message Processors, but we considered more appropriate to specify them as separate categories in the ontology because it responds better to the proposed solution. Message processors are used by the central component to automatically adapt the structure of the sensor fusion solution (e.g. changes in the sensor set), while algorithms are related with the quality of the fusion products. Furthermore, according to the initial specification, algorithm configuration is determined by human operators.

These classes are complemented with some object properties used to express how elements relate and placing constraints on those relationships (figure 4.5). We explain next the meaning of these properties and how they are used. They are grouped according to the domain class of the property.

- Message processor properties:

  - generatesMessageType: used to indicate that a message processor is a direct source of messages of a certain type –that is, the processor creates the information associated to those messages. For example, a RadarLocalTracker generates RadarLocalTrack messages.

  - processMessageType: used to indicate that a message processor is interested in a type of message and/or specialize in processing it. This means that it will not simply forward those messages, but rather generate a derived product from them. Taking the previous example of a RadarLocalTracker, it processes RadarPlot messages to generate the declared RadarLocalTracks instead. These properties give a hint of how processors can be connected when the requirements of an element are related with the output of another one, as suggested in section 3.3.2.1 with the properties "produces" and "consumes".
  - receivesMessagesFrom and sendMessagesTo: the initial solution describes a rigid processing pipeline for the fusion-related components (sensors and trackers). These complementary properties are used describe the mandatory relationship between processors, e.g. a RadarLocalTracker receives messages from a RadarSensor and sends its output to a GlobalTracker.

- Tracker properties: it contains a single property called usesAlgorithm. It is used to define the internal processing structure of trackers, i.e. the slots where algorithms are plugged, as shown in 4.2. An example can be shown in figure 4.4, where this property is used to define 4 slots for configurable algorithms.

- Track properties: contains inverse properties hasComponentOf and isComponentOf. They are used to express that global tracks are composed of radar and AIS local tracks, and restrictions as the limit of a single AIS track per global track. Such restriction is displayed in Protege as in the previous example of "usesAlgorithm" (in the "SubClass Of" part of the class description), and the associated XML/RDFS code is the following:

```
<Class rdf:about="GlobalTrack">
  <rdfs:subClassOf rdf:resource="Track"/>
  <rdfs:subClassOf>
    <Restriction>
      <onProperty rdf:resource="hasComponent"/>
      <onClass rdf:resource="AISTrack"/>
      <maxQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">
        1
      </maxQualifiedCardinality>
    </Restriction>
  </rdfs:subClassOf>
</Class>
```

- Domain properties: at this point, it is populated by a single property coversArea, that can be used to link sensors with their coverage area in case it is useful for doing spatial reasoning.

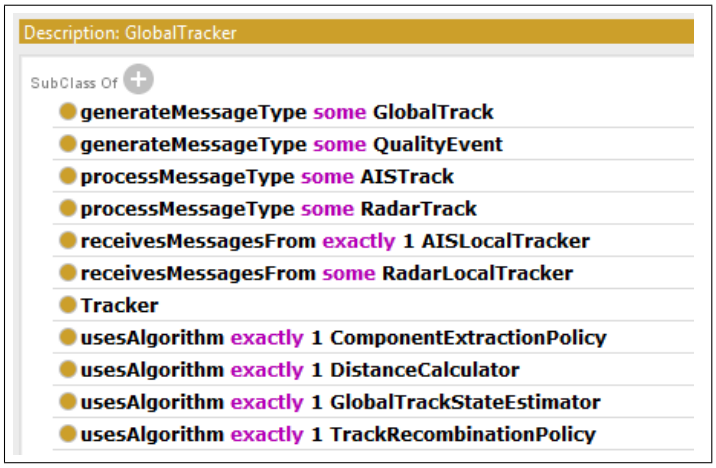


Figure 4.4: Definition of the GlobalTracker class in problem-space ontology. OWL restrictions are used to express the features and constraints identified during solution design: it accepts a single AIS tracker and an unlimited number of radar trackers, has 4 slots for configurable algorithms that fulfill an specific function, and generates global tracks and events used to calculate quality metrics.

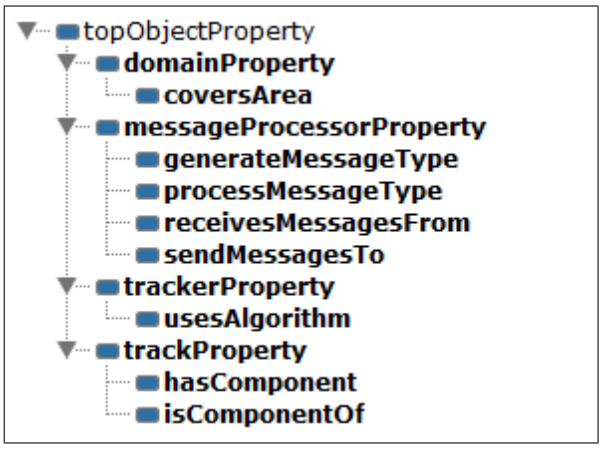


Figure 4.5: Object properties used to define the problem space. They are organized according to its domain class.

The proposed classes and properties are useful as a static description of the domain, but have a limited applicability to further automatic inference processes. Next section presents a representation for the relevant context information, and after that we will review how the combined language can be used to enable automatic algorithm selection, and which extensions are required for more advanced reasoning processes.

4.3.2 Context

According to the original specification, it is not necessary to extract or use context in this problem: human operators are responsible of sensor parameterization and algorithm selection. However, in preparation for possible extensions, we used technical meetings to gather information about the criteria used to determine sensor parameterization and algorithm selection.

Radars have a large number of parameters that can be configured separately, but analysts have decided that it requires a trained expert to know how they must be combined in order to get the desired result. This problem by defining profiles (combinations of parameters) that are specific for environmental situations or to improve the obtained information in some aspect. More concretely, we have identified the following profiles:

- Standard profile. Works well in the general case, achieving a good balance between update rate, detection ratio, discerning close vessels and other quality measures.
- Rainfall.
- Reduce noise. Related with the preprocessing software of the radar, uses a more aggressive filtering profile that eliminates false positives at the risk of discarding a true target.
- Follow fast targets. Mostly affected by antenna rotation speed.
- Determine boat size. Requires azimuthal –angular– accuracy and avoid dispersing energy to flood adjacent cells.
- Discern close targets. Requires detecting small angular gaps between boats.

Weather is an important contextual factor. Apart from affecting the optimal parameters for vessel detection, rainfall affects radar coverage by reducing the reach of the signal in a way that cannot be compensated by parameterization. Closely related with the weather, the state of the sea is known to affect radar performance. Waves can occlude small boats, and also generate false positives that have to be filtered out. The state of the sea can be characterized using the Douglas scale. It combines two numerical codes that describe waves: one for the wind waves (generated by the wind blowing directly over the area where waves are considered) and the

swell (not caused by wind but by weather systems, generates regular interval waves that have a long wavelength and travel long distances).

Our context representation contains three environmental features represented as subclasses of Environment: Rainfall, WindWaves and Swell. A potential Context Management module is in charge of deciding the scale of values for these features. Since the system covers a large extension, it may be necessary to specify environmental features by zones. For this purpose, we will use the Area concept defined in the problem space ontology. We achieved this by creating a new Area class in context ontology and declaring it equivalent to Area in problem space ontology (we are showing here full IRIs that have been skipped in previous examples):

```
<Class rdf:about="http://www.semanticweb.org/maritime-surveillance/
                  ontologies/2015/4/ms-context#Area">
  <equivalentClass
    rdf:resource="http://www.semanticweb.org/maritime-surveillance/
                  ontologies/2015/4/problem-space#Area"/>
</Class>
```

An alternative solution involves defining a separate ontology for common domain terms, that is imported by problem space and context ontologies.

Several aspects of the design, including location of the radars, are affected by how vessels are geographically distributed, and their expected behavior on each place. We are going to discern four basic zones: harbor, anchorage, coast and open sea. Let us explain the relevant facts about each one:

- harbor/port: contains anchored, stopped vessels. Anchored boats are densely packed near piers and loading bays. Vessel density is higher that in other areas, and they are expected to move slowly. Occlusions are frequent in this zone, causing radars to miss detections. Terrain and other fixed elements can occlude targets too.

- anchorage: a zone outside the port where boats stop waiting before entering it, or for other reasons. Density is also high, but less than in the port.

- coast: the band of water close to the coastline. Only small boats sail in these shallow waters. Radars may not detect targets in the coast, making it a suitable route by smugglers or pirates that want to avoid authorities control.

- open sea: any place not in the other zones. Boat in open sea are likely to be sailing at a normal speed or doing other tasks (e.g. fishing).

We have created a Zone class and four subclasses, one per zone. Each class will be populated with instances of the concept it represents in the scenario. Zones have an associated geographical

location, that will be defined using the Area class. This fact is expressed by the object property hasArea.

Some variables can be argued to be part of the problem space description rather than a contextual piece of information. This applies, for example, to the functional zones (harbor, anchorage, etc), that would most probably qualify as part of the problem domain if this system aimed to cover Levels 2-3 of the JDL model. As reasoned in 3.3.2, these ontologies describe parts of a bigger, common language. The optimal arrangement is more related with the conceptual question of how we, as designers, prefer to approach the problem, than with the functional possibilities of alternative choices.

In this case, we look for a design that solves current version of the problem –manual adaptation of the fusion solution–, but takes into account future extensions for making the system self-adaptive. With this purpose in mind, we decided to classify variables as part of the context when they are not required by the original tracking/fusion solution, or when it is information currently taken into account by human operators during manual adjustment of the system. So, we are using the distinction between problem space and context as a mechanism to separate the first implementation of the fusion system from information that might be used in the future.

4.4 Evolving the design towards a self-adaptive solution

At this point, we have the base for implementing an adaptive sensor fusion solution that satisfies the initial set of requirements. This section drafts a plan for evolving current solution to a self-adaptive system that can change fusion algorithms and parameters according to different criteria.

Some time after closing the design phase, we identified some worst case scenarios were the performance requirements could not be met. They were related with the high asymptotic complexity of some of the applied algorithms, as JPDA, that makes them unfeasible for large problem instances and can block the system. We considered necessary to design a "graceful degradation" strategy capable of keeping the fusion system working even if the produced results have a lower quality. This fact was discussed during technical meetings, concluding that operators are responsible of controlling that the selected processing scheme does not exceed system capabilities. They keep the fusion system within the required performance bounds applying some general rules as masking some sensors or avoiding costly algorithms in densely populated areas, as well as using the spatial configuration capabilities of the system for restricting high-cost algorithms to small areas where they are actually necessary –usually responding to dynamic situations. Thus, the already designed zonal configuration features of the system are a suitable tool for solving this problem.

We are going to solve this problem following the same approach used for generating quality metrics (since performance indicators are a type of quality metric). Message processors generate quality events that describe the time required to complete their tasks. These messages, apart from being used to generate additional quality metrics, will be gathered by a processor specialized on analyzing execution times that detects anomalies and sends alarms/events to central control. The first step is to modify ontologies to add the concepts and relations related with the computational performance problem. The modification involves the following actions:

- Create a subclass of QualityEvent for performance events: PerformanceQualityEvent.
- Create a subclass of Message for notifying performance related anomalies: PerformanceAlarm.
- Make Algorithm a subclass of MessageProcessor so that we can define that Algorithm generateMessageType PerformanceQualityEvent. In the first version of the problem space ontology, Algorithm was defined as a separate class because it was conceptually clearer to treat them as
- Create datatype property "generatedByProcessorWithId" with domain over individuals of QualityEvent class, and range over an integer number (can be changed to match the real data type of IDs). Add restriction to class PerformanceQualityEvent so that its individuals include exactly one instance of this property.
- Create class PerformanceMonitor as a subclass of MessageProcessor. Add restriction to express that it processes messages of type PerformanceQualityEvent (see 4.3.1 for an example on creating OWL restrictions), generates PerformanceAlarm messages and that it sends them to CentralControl processor.
- Modify CentralControl class to include the reciprocal restrictions.

The following step consists on implementing the automatic adaptation processes. Next, we present two alternative solutions for the performance monitoring part. The general adaptation process that optimizes the quality of the fusion will not be addressed here, as it is far more complex and do not provide additional exemplary aspects about the use of the framework.

As explained before, the adaptive actions for keeping computation within required performance bounds fall in two different categories: a background of static rules that constraint the application of certain algorithms in crowded zones, and a dynamic strategy that checks the execution time of individual processors to detect problems in real time. The static rules are related with the Zones defined as part of the context ontology. One solution consists on defining rules and constraints in the ontology, and reasoning directly over them. The work (Gómez-Romero et al., 2015) describes the implementation of a reasoning process over an OWL

ontology using Semantic Web Rule Language (SWRL) rules, for detecting abnormal vessel behavior inside a port. It describes a problem with some similarities with ours, as a reasoning process that involves geometric/geographic criteria and contextual information.

Central control module can, alternatively, load restrictions/rules from the ontology and integrate them as part of the algorithm selection logic. The second part of the solution, the dynamic strategy, is related with the implemented changes in the problem space ontology. Algorithms generate quality events describing the time required to complete their tasks, or can alternatively send those events at the beginning and end of a task. The performance monitor receives those messages, detecting processes that consume too much CPU or that started running a task too long ago but have not notified the end. Central control is alerted about such situations, and can decide to stop a process or substitute it with a less costly algorithm.

4.5 Conclusions

This chapter describes how to apply this thesis' proposal to build a sensor fusion system for maritime surveillance. We have reviewed the major steps of the process: design a fusion solution from a set of requirements; design the mechanisms that will be used to make the system adaptable; describe the domain of the problem, the elements of the solution and context information using ontologies; and finally modifying the system to include new requirements and processes.

We have shown how ontologies can be used to provide a compact, unambiguous description of some aspects of the problem and its solution. In our experience, using ontologies during the design phase of the solution helped in the identification of a few inconsistencies, and resulted in a cleaner result. The first version of the system (manual adaptation process) makes use of the proposed software abstractions with good results: the system is more flexible thanks to them, their impact in the computational performance of the system is negligible, and the software makes a better use of multiprocessor computers due to the low coupling between algorithms.

5

Application to vehicle navigation in urban environment: Design and Experiments

Navigation in urban environments pose a challenge for GPS-based navigation systems. Buildings and other obstacles affect the incoming satellite signals either blocking it and creating delayed echoes. Traditional GPS navigation systems fuse location information with road maps to improve the accuracy. This approach is still insufficient in urban scenarios with short, narrow streets: trying to position series of low quality GPS fix over a dense grid of streets creates an ambiguous situation.

In this chapter we explore the problem of mixed urban/road vehicle navigation without maps. It has a great importance for forthcoming applications such as cooperative driving, automatic maneuvers for pedestrian safety, autonomous urban vehicles, and collision avoidance, among other Intelligent Transportation System applications.

This chapter illustrates how the proposed architecture can be used to build a system that integrates context information to adapt its structure, to correct undesired effects in uncalibrated sensors, and also as an additional source of information that improves the accuracy of the algorithms.

The experimental part is based on a redundant set of sensors with heterogenous features: a GPS receiver with differential capability and a mid-cost Micro Electro-Mechanical Systems (MEMS) Inertial Measure Unit (IMU) that are mounted in the test vehicle, and a smartphone that provides low quality versions of the same sensors and is placed inside the vehicle. Tests take place in open road conditions with real traffic.

5.1 Introduction

5.1.1 Fusion of GNSS/IMU for navigation

Localization by Ground Navigation Satellite System (GNSS) has become a ubiquitous facility in outdoor conditions. There are different enhancements for GNSS, usually classified in Ground-Based Augmentation System (GBAS) and Satellite-Based Augmentation System (SBAS). European Geostationary Navigation Overlay Service (EGNOS) is the European reference for SBAS system, with 33 ranging and integrity monitoring stations, while Wide-Area Augmentation System (WAAS) is the reference in USA, by the Federal Aviation Administration. GBA systems consist of ground antennas which transmits differential corrections by VHF data broadcasts to the receiver. An example is the US Local Area Augmentation System (LAAS), used in the proximity of airports to guarantee maximum integrity in GPS position, but this idea is being available in many other environments (Chae et al., 2010; Morales, 2008).

Besides the ubiquity of GNSS receivers, the recent advances in low-cost inertial sensors based on MEMS technology has let them emerge as the other big reference technology for navigation. The inertial navigators contain a set of accelerometers in orthogonal axes and aligned gyroscopes which sense vehicle turn rates and accelerations in the body frame. The processor obtains the attitude of vehicle by integrating angular rate measurements in time, and then the position is computed and continuously updated with respect to an initial solution with the projected accelerations measured on body frame.

So, GPS and INS sensor systems are complementary key technologies, and a carefully designed sensor fusion process can be used to provide a navigation solution. This type of systems can be explained in simple words as enhancing GNSS with dead-reckoning capability, so that accurate navigation remains available for a certain amount of time when GNSS signal data becomes unavailable or seriously degraded. However, experience indicates that this solution can be very limited, and the time to support outages or degradation of GPS position is not much longer than some tens of seconds due to very quick drifts in time. GPS/INS fusion is vulnerable to residual errors so a continuous monitoring of the process is necessary to guarantee that the quality of navigation is acceptable, minimizing the effect of these factors during GPS availability drops.

5.1.2 Challenges

An important aspect to take into account is the need of using available low-cost sensors, in order to develop scalable solutions, which can be implemented at large scale and facilitate new driving coordination paradigms (Shladover, 2009). So, the basic sensing technologies must

be improved by powerful data processing techniques to handle high-performance expectations, resilient to main causes of faults and lacks of availability/integrity in sensors.

This system presents significant variations of quality and reliability depending on the conditions and available enhancements. The accuracy is typically around $20m$ (1 sigma) in urban outdoor conditions, depending on the number of available satellites and geometrical configuration (dilution of precision, DOP), signal propagation through the atmosphere and especially on the presence of a multipath,when a signal is reflected by elements in the environment and the receiver receives the original one but also several echoes. The worst case of multipath is referred as the "urban canyon" problem (Morrison et al., 2012), when there are almost no satellites with direct visibility and receivers have to use signals bounced in walls of close buildings, with the corresponding degradation or even loss of any solution. Figure 5.1 shows the different problems arising in urban canyons. The vehicle (dot in the ground) has not direct line of vision of the first satellite (left to right), but receives a reflected signal that introduces an error in the computation of a position fix. Signal from second satellite reaches the vehicle both directly but also after bouncing in the walls, forcing the receiver to identify and discard the duplicate. Third satellite is completely blocked by the buildings.

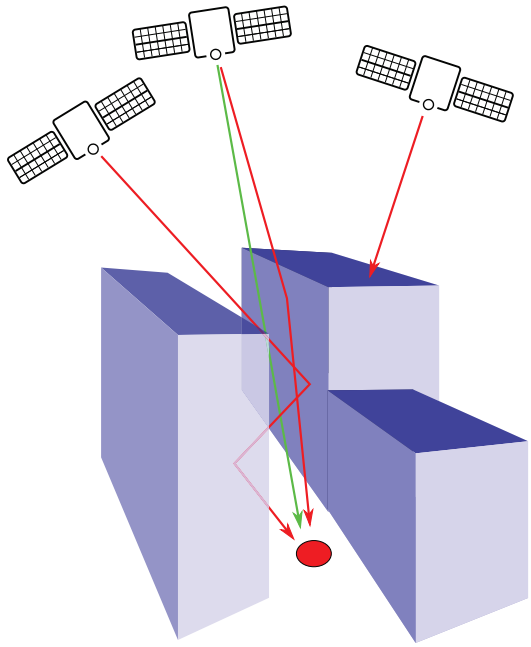


Figure 5.1: *Urban canyon* creates a degraded GPS environment where signals are blocked and reflected.

The proposed system has two features that make this problem more complicated to solve, and are thus key to understand the contribution of the work. The first one is that the sensing system is a simple ensemble of low cost commercial off-the-shelf (COTS) sensors that can be mounted anywhere in the target vehicle. This circumstance increases the difficulty of the

problem, since the sensors can be subject to additional dynamic effects that have not been explicitly modeled -–such as rolling in corners due to the lateral acceleration.

The second feature is that the sensors have not been calibrated before executing the experiments. Instead, the system automatically corrects those effects online. Continuous self-monitoring and adaptation is necessary to work with the selected low cost sensors, in order to compensate the varying biases and errors over time. The experimental results, with approximately 50 km in different and representative conditions show the competitive performance of the proposal.

5.2 Proposal

This section defines a fusion solution based on adaptive filters (KF, UKF), which are continuously monitored by a contextual reasoning process, to provide improved performance. The system features a cascaded architecture, separating attitude and kinematic filters, to create a loosely coupled closed-loop scheme that continuously estimates the INS biases to correct them and exploit whenever the GPS data is degraded or unavailable.

The main contribution is the proposal of a robust and adaptable solution, exploiting the good trade-off between non-linear estimation and efficiency of UKF, and including explicit domain knowledge to drive the algorithms.

The system includes explicit knowledge reasoning about vehicle dynamics, to adapt the model to the real conditions. Conditions such as stops, straight motion, lane changes, turns, roundabouts, can be integrated in the model or its associated reasoning processes. Besides, there is a GPS monitoring system with rules depending on conditions based on extra information (availability and age of differential corrections, number of satellites, Dilution Of Precision (DOP) value, standard deviation, etc.). This is applied to weight the fusion parameters or switch the bias estimation processes accordingly to the conditions. Additional external information, such as the presence of blocking buildings and trees creating multipath problems could be considered in a future step, together with the integration of static databases about road and terrain elevation.

Other additional context information can be used to alter the structure of the solution, as deciding the set of sensors to be used or selecting the set of algorithms returning the best fusion solution.

5.2.1 Sensors

5.2.1.1 Mid cost equipment for on-board platform

The vehicle is equipped with two sensors, a Novatel GPS receiver and a MicroStrain IMU. Both sensors were mounted in a magnetized sheet of metal that can be attached to and removed from the roof of any vehicle, as shown in Figure 5.2. This arrangement guarantees sensors will not move during tests, and allows to correct position and alignment in case it is required.

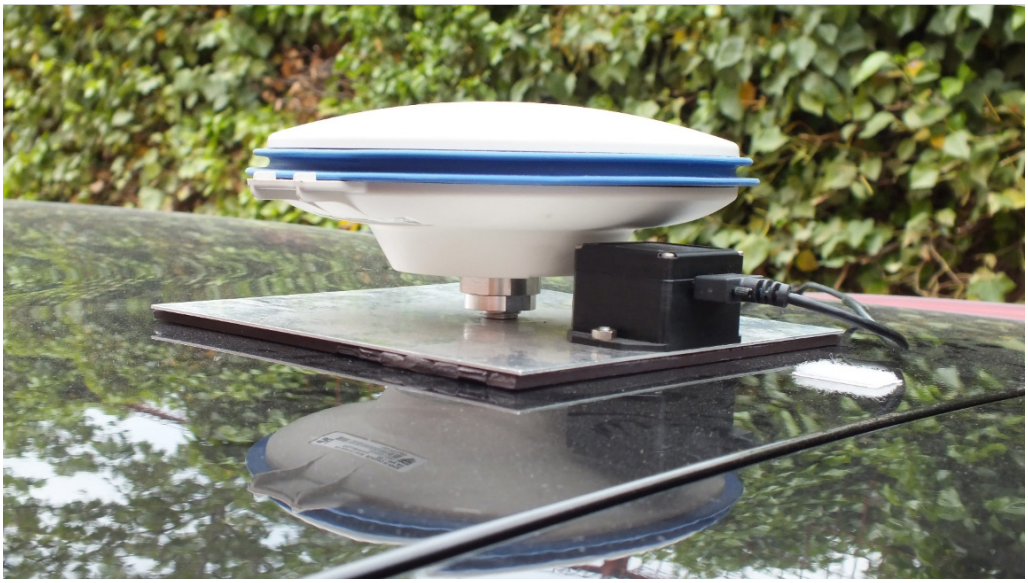


Figure 5.2: Mid-cost sensors as mounted in the roof of the test vehicle.

GNSS Novatel OEM-1G The mid-cost GNSS solution is a Novatel OEMV-1G board. It offers GPS+GLONASS L1 tracking, providing reliable positioning even in obstructed sky conditions. The receiver is embedded on a Novatel compact enclosure (FlexPak-G2-V1G) for outdoor applications as base station and vehicle position in urban environment. This device can work in differential (DGPS) and single point position modes (SINGLE mode). The experiments of this paper combine both. Maximum accuracy in single point mode requires "optimal conditions", which is translated into observing six or more healthy satellites and relatively low multi-path (to assure enough quality of the received data).

IMU MicroStrain 3DM-GX2 We have selected a MicroStrain 3DM-GX2 IMU, which integrates triaxial accelerometer, gyroscope and magnetometer. It is a high-quality MEMS-based device in the price range of the few thousand dollars, so it is considered a mid-cost device by some studies (Chao et al., 2010). It includes an internal Complementary Filter (Higgins, 1975) that fuses raw data into a stabilized attitude estimation. The accuracy of the calculated

orientation is around $0.5 - 2.0$ degrees according to manufacturer specifications. The proposed experiments focus on raw gyro and acceleration data instead, since they are more suitable for calculating the required pieces of contextual knowledge.

5.2.1.2 Low cost smartphone platform

Experiments are based on the data captured by an LG Nexus 5 smartphone. This high-end terminal features a number of sensors that replicate the on-board platform. However, while the on-board platform is designed to work exclusively for this system, the smartphone is an auxiliary tool that offers a very limited availability. External factors as battery life, or being used for other purposes (making calls, taking photos, playing games) will cause an interruption of its service as data provider for the sensor fusion system. These causes are explored later in section 5.2.4

GNSS Qualcomm WTR1605L GPS is integrated in the RF processor, Qualcomm WTR1605L (2G+3G+4G-LTE+GPS). Apparently, these chipsets integrate some limited WAAS/EGNOS functionality, improving horizontal location accuracy up to 3m (1 sigma) in optimal conditions. This statement is not confirmed by the documentation available on manufacturer website, although we have checked that the device reports horizontal errors within this bound of 3 meters.

Small GNSS devices, such as these type of chipsets in smartphones, have antennas with a low gain and a limited computational power. This makes very difficult (or even impossible) to get an initial estimation of the location. This problem is solved using Assisted Global Positioning System (A-GPS) technologies, that use additional information to reduce dramatically the time until the first position is calculated –something called Time To First Fix (TTFF). A-GPS receivers communicate with external systems as cell network antennas that can provide a first approximation of their geographical location, forward useful data about satellite constellation or atmospheric perturbations, and even do some computations for the device.

Modern smartphones use advanced assistance technologies, that combine rough locations from the cell network and nearby Wi-Fi networks with other data as barometric altitude. Nexus 5 includes a Bosch Sensortec BMP280 barometer, that can provide altitude with a relative accuracy equivalent to +-1 m, and a temperature offset equivalent to 0.12m/K.

IMU InvenSense MPU-6500 Inertial measures are provided by an InvenSense® MPU-6500™ chipset. MPUs have been proposed by mobile sensor manufacturers as an evolution of traditional IMUs that include powerful on-chipset processing/fusion algorithms to compensate the lower quality of the sensors and reduce energy consumption. They generate the traditional raw measures as well as a number of processed outputs such as attitude or step count

Table 5.1: Sensor refresh rates

	Refresh rate	
IMU	***Maximum***	***Configured***
On-board	300 Hz	50 Hz
Smartphone	100 Hz (estimated)	50 Hz
Magnetometer	***Maximum***	***Configured***
On-board	300 Hz	50 Hz
Smartphone	100 Hz (estimated)	50 Hz
GPS	***Maximum***	***Configured***
On-board	10 Hz	1 Hz
Smartphone	1 Hz (variable)	As fast as possible
Barometer	***Maximum***	***Configured***
Smartphone	1 Hz (estimated)	As fast as possible

(podometer). The inertial measures can be complemented with readings of the AsahiKasei AK8963 magnetometer.

Data is captured using a custom Java (Android) application. Sensors are accessed through Android API, abstracting the real hardware and its details. The application sets sensors to provide the highest possible update rate. Table 5.1 describes the compared refresh rates of the employed devices.

5.2.2 Notation and conventions

Let us define, beforehand, some conventions and rules about notation followed over the rest of the document. The coordinate reference systems used by the sensor fusion solution are Cartesian, right-handed. World coordinates follow ENU convention. The origin is located near Colmenarejo or Leganés Campus of the University Carlos III de Madrid, depending on the experiment. All tests take place within a distance of 20 km from the origin of coordinates. The three axes of world coordinate system will be referenced as X, Y, Z through the rest of the document.

Both the IMU, GPS and vehicle are considered to share a common reference system, that will be named as "Local". Its axes are named X^L, Y^L, Z^L, with X^L following the motion direction of the car (positive when the car moves forward, negative when reverse), Y^L growing towards the left side of the vehicle, and Z^L pointing upwards. For simplicity, the origin matches that one of the IMU. Raw GPS, accelerometer and gyroscope measures are transformed to this reference system before being used by the system. The case of the smartphone is more complex, since it is not completely attached to the structure of the vehicle. Device readings follow Android local axes, which are transformed to our unified local coordinate system and have to be aligned with the global axes afterwards. We have not included this difficulty in

the formulation of the problem: since the experiments are run offline, smartphone data is preprocessed so that the framework does not need to take care of axis alignment details.

The word "attitude" is referred to the vehicle/IMU. It express the orientation difference –rotation— between the local axes X^L, Y^L, Z^L and world reference system X, Y, Z. This works represent rotations either as matrices or as global Euler terns $Eul_G[x, y, z]$. The latter express the attitude as a sequence of three rotations around global axes X, Y, Z. These individual rotations are respectively named tilt, elevation and azimuth.

Gyroscope readings express angular rates around local axes X^L, Y^L, Z^L. They will be also referred as roll, pitch and yaw respectively.

5.2.3 Design of initial solution

Figure 5.3 shows the initial fusion solution, before adapting it to the context-aware adaptive framework. This proposal does not take into account sensor redundancy, and follows a fixed processing scheme. In general, information flows from left (sensors) to right (output of estimation processes). Given the importance of context information for calibrating sensors, we have included a module that provides context information to every other component in the system. The attitude estimation is feed back into the sensor refinement process.

The figure shows how data moves around the system. Solid lines represent raw data as captured by the sensors, blue dotted lines refers to refined sensor information –this that is, original data that has been processed to compensate known errors, or to produce features that summarize a number of raw values—. Finally, dashed lines are reserved for new data that has been obtained by advanced processing techniques such as filters or reasoning algorithms. The meaning of each data component will be detailed in the corresponding section, where it is produced.

5.2.3.1 Estimation of 3D vehicle pose

Many navigation-related works describe the motion of ground vehicles using a 2D model, as in (Rezaei and Sengupta, 2007), where GPS is combined with a wheel speed detector and a gyroscope. This simplification works well when acceleration measures are not used or gravity effect does not need to be corrected, and the error of using gyroscope readings in their own frame of reference does not introduce significant errors due to the features of vehicle dynamics.

However, this simplification cannot be applied in many car driving conditions. The best example is probably that one related with accelerometer readings and the effect gravity has on them: a vehicle driving uphill at constant speed in a β degrees slope (elevation angle) registers a residual acceleration along its local X^L axis with magnitude $acc_{Rx} = g \cdot sin(\beta)$, that

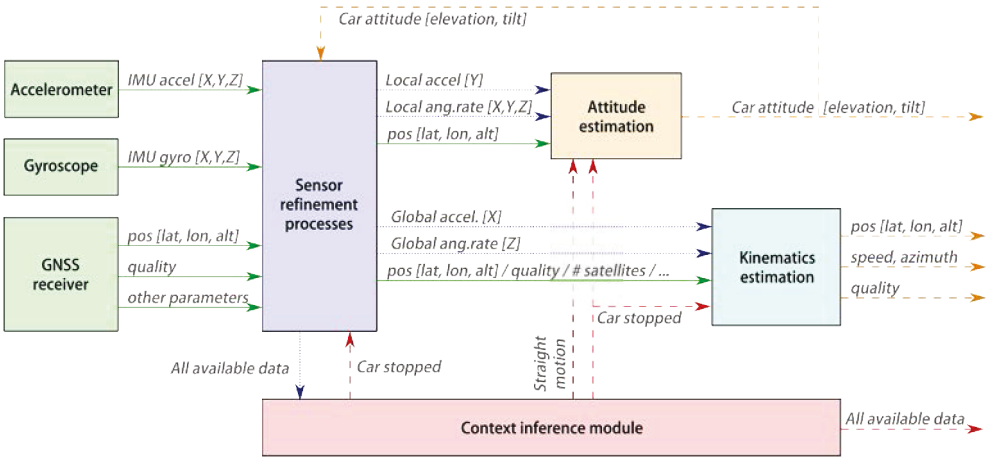


Figure 5.3: System architecture for the initial non-adaptive solution.

is approximately $0.17m/s^2 \cdot deg$ for small angles $\beta < 10$ deg. This residual acceleration is large enough to introduce important errors when the system has to predict vehicle state using only inertial measures, even for short periods. The same can be applied to accelerations along Y^L axis depending on the tilt of the vehicle, and to how gyroscope registers turns depending on road features as the cant.

These arguments would be enough to justify a full state estimation, which can account for the full spectrum of movements of a ground vehicle in the 3D space. The most straightforward approach consists in applying a single filter that works with a constrained 6 Degrees of Freedom (DoF) system. This solution can be found in some of the already cited works, as (Li and Leung, 2003; Zhang et al., 2005).

A two-stage solution has been preferred instead: the first block estimates the attitude of the vehicle for correcting the inertial inputs, and the second one predicts its kinematics using a simpler 2D model that take into account the motion constraints of a ground vehicle. It has been shown that uncoupled solutions offer a poorer performance when compared with loosely- and tightly-coupled formulations (Hafner et al., 2011). The reason that impelled us to select the uncoupled solution is that it makes easier to apply the implemented sensor refinement and context-based techniques.

Next sections describe the prediction and measurement models of the two state estimators, those estimators, which will be used in two uncoupled UKFs. After that, the UKF algorithm has been particularized to the problems of attitude and car kinematic trajectory, using the cascaded architecture introduced before. In the following subsections, the particular state vectors and non-linear dynamics of each subproblem are presented. For an introduction to the UKF, see Appendix A.

Attitude estimation model Let us describe the attitude of the vehicle as a global Euler tern $Att = [\psi, \varphi, \theta]$, where ψ is tilt, φ is elevation and θ is azimuth. This tern represents a sequence of ordered rotations –first X, then Y, last Z— around the axes of world system of reference.

This part of the system, shown in Figure 5.4, estimates the tilt and elevation of the vehicle. These two components contain all the necessary information for:

- Subtracting the effect of gravity from accelerometer readings,
- Translate local gyroscope readings to world system of reference –great importance during turns where the car is tilted.

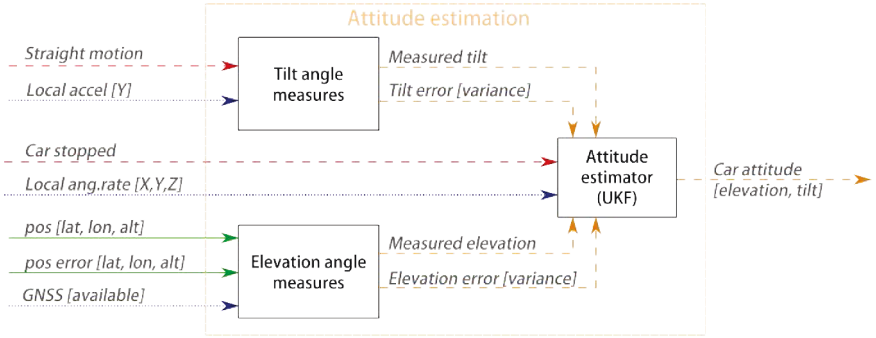


Figure 5.4: Sensor refinement module.

So, the state vector is $x_{att} = [\psi, \varphi]$. Given the gyroscope readings $u_{gyr} = [g_x, g_y, g_z]$, which represent the angular rates (in radians) around the local $\{X^L, Y^L, Z^L\}$ axes, the prediction model for estimating the new attitude of the car after a time span t follows the procedure described below. First we detail the prediction model, which takes as control inputs the gyroscope readings and carries out a numerical approximation, and then the observation model, which needs external information to infer observations of these magnitudes. The estimates cast by both models are then integrated with the UKF estimation process.

Since gyroscope readings represent a simultaneous rotation around the three local axes at the marked angular rates. That means that the local reference system changes continuously over time. For infinitesimal time increments, the simultaneous rotation is similar to applying three sequential infinitesimal rotations around each one of the axes, with independence of the

order. Using matrix form, this can be expressed as:

$$M_\delta = M(X, \delta g_x) \cdot M(Y, \delta g_y) \cdot M(Z, \delta g_z) = \\ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta g_x) & -\sin(\delta g_x) \\ 0 & \sin(\delta g_x) & \cos(\delta g_x) \end{bmatrix} \cdot \begin{bmatrix} \cos(\delta g_y) & 0 & \sin(\delta g_y) \\ 0 & 1 & 0 \\ -\sin(\delta g_y) & 0 & \cos(\delta g_y) \end{bmatrix} \cdot \begin{bmatrix} \cos(\delta g_z) & -\sin(\delta g_z) & 0 \\ \sin(\delta g_z) & \cos(\delta g_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

Where δg_k represents the infinitesimal angle rotated after sustaining the g_k angular rate around axis K during the infinitesimal time δt. By integrating the differential rotation M_δ over time, the total pose change can be obtained.

Our model performs a numerical approximation of this approach. First, the prediction time span t is divided in n steps of duration d=t/n seconds. The pose change in any of the steps is calculated as a sequential rotation of $u_{gyr} \cdot d = [g_x \cdot d, g_y \cdot d, g_z \cdot d]$ radians around the three axes, which results in the differential rotation matrix M_d. The total pose change after prediction time span is the n-th power of the differential rotation matrix, $M_{rot} = M_d^n$.

As a side note, our choice for n is such that the duration of the step is smaller than $d = 10^{-4}$ seconds. The obtained results were compared with those yielded by the widely accepted quaternion kinematics equation, resulting in errors around one part per billion.

The new vehicle attitude can be calculated as:

$$M(x)_{+t} = M(x) \cdot M_{rot} \tag{5.2}$$

Where $M(x)$ depends on the reduced vehicle attitude $x_{att} = [\psi, \varphi]$ expressed as a rotation matrix. Note that pose change matrix post-multiplies the attitude because the rotations are expressed around global axes. Transforming the resulting matrix $M(x)_{+t}$ to Euler notation –again, around global axes— and discarding the azimuth values gives the new vehicle attitude $x_{att(+t)} = [\psi', \varphi']$.

Finally, UKF equations are applied to combine the prediction with asynchronous measures and provide the estimated tilt and elevation angles.

Generation of tilt angle measures The tilt angle of the car can be calculated based on the gravity transmitted to accelerometer Y^L axis. As previously stated, the reading on this Y^L axis at a given time is:

$$acc_y = a_y^L + b_y^L + g_y^L + n_y(16) \tag{5.3}$$

Where:

- a_y^L is the real lateral acceleration associated to vehicle motion,
- b_y^L is the bias of the accelerometer on its Y axis,
- g_y^L is the effect of gravity in the local Y axis of the accelerometer,
- n_y is a random sample distributed as white noise with variance σ_{acc_y}.

During fragments where the vehicle is moving in a straight piece of road, the car will not be subject to lateral accelerations and it is valid to assume that $a_y^L = 0$. Regarding the random noise, it can be cancelled by averaging several measures representing the same effective acceleration.

This also happens during straight motion. If bias has been corrected, we found that in these fragments of trajectory $acc_y = g_y^L$. The effect of gravity can be calculated by transforming it to local axes. Assuming that the attitude of the car is $x_att = [\psi, \varphi]$, the rotation matrix that performs such transformation is:

$$
\begin{aligned}
M(X,\psi) \cdot M(Y,\theta) = \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} = \\
&= \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ \sin(\psi)\sin(\theta) & \cos(\psi) & -\sin(\psi)\cos(\theta) \\ -\cos(\psi)\sin(\theta) & \sin(\psi) & \cos(\psi)\cos(\theta) \end{bmatrix}
\end{aligned}
\tag{5.4}
$$

This matrix has to multiply the global-referenced gravity vector $[0, 0, g]^T$. The Y component of the transformed gravity vector will be $-\sin(\psi) \cdot \cos(\theta) \cdot g$. The tilt and elevation angles of a vehicle in normal road conditions are usually in the range $[-5; 5]$ degrees, and hardly ever exceed 10 degrees. This makes possible to apply the approximation $\cos(\theta) \sim 1$.

Back to the reading under straight movement conditions, we have that:

$$
acc_y = g_y^L = -g \cdot \sin(\psi) \tag{5.5}
$$

The tilt angle ψ can be calculated as $\arcsin(-acc_y/g)$. Thus, true tilt angle at time step t can be estimated using the average accelerometer reading over a window of k samples taken during straight motion as:

$$
\psi = \arcsin\left(-\frac{\sum_{i=0..k-1} acc_y(t-i)}{(g \cdot k)}\right) \tag{5.6}
$$

Generation of elevation angle measures Raw elevation angle can be estimated using GPS information of consecutive measures, as illustrated in Figure 5.5. Assuming that GPS measures have been transformed to a Cartesian system of reference, we can calculate:

$$D_G = \sqrt{(lat_2 - lat_1)^2 + (lon_2 - lon_1)^2} \tag{5.7}$$

$$A = (alt_2 - alt_1) \tag{5.8}$$

$$\delta\theta = \arctan\left(\frac{A}{D_G}\right) \tag{5.9}$$

This estimation of pitch angle is quite sensitive to the measurement conditions. In one hand, it is important to use two GPS measures close enough in time so that the path of the vehicle between them can be well approximated by a straight line, and also that the elevation angle has remained near constant. On the other hand, the 3D points must be as separated as possible so that the error of GPS does not have a large impact on the calculated elevation. Our GPS device provides measures at a fixed rate of 1Hz. The distance between consecutive positions will depend on the speed of the vehicle.

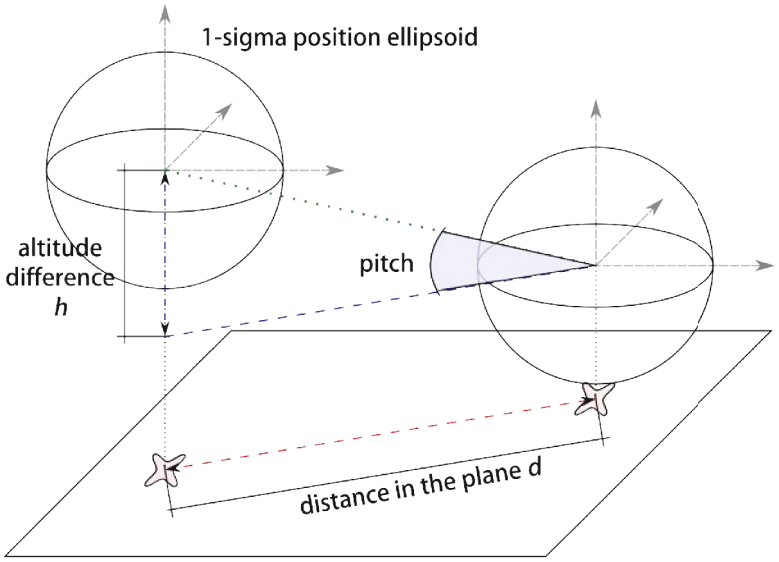


Figure 5.5: Calculation of elevation angle from two GPS measures.

5.2.3.2 Estimation of 2D vehicle kinematics

For locating the vehicle on surface, a 2-dimensional model is proposed. We assume no wheel slippage. In this section, we define three different kinematic models. The first one is a near-constant velocity linear motion model to be integrated in a Kalman Filter, that can be used if IMU information is not available (only GPS measures). The other two models are non-linear and, thus, will be used by the Unscented filter. They use a state vector $x = [p(x), p(y), v, \varphi]^T$,

where $p(x), p(y)$ describe the position of the vehicle in the X-Y plane of world coordinates, v is the linear speed of the vehicle and φ is the azimuth angle which marks its course.

The azimuth angle complements the output of the attitude estimation model, to form the complete attitude vector of the vehicle. The prediction function takes the state of the system, and a control input $u = [a_x, \omega_z]^T$ formed by the corrected (world coordinates, non-biased) longitudinal acceleration of the car and yaw angular rate.

Near-constant velocity model The near constant velocity model assumes a straight motion pattern where the attitude of the vehicle is aligned with its speed. It defines a state vector $x = [p(x), p(y), v(x), v(y)]^T$, where $p(x), p(y)$ describe the position of the vehicle in the X-Y plane of world coordinates, and $v(x), v(y)$ represents the speed of the vehicle in the same plane.

The prediction model is defined by the following matrix:

$$x_{t+\Delta t} = f\left(x_t, \Delta t\right) = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

Non-linear model for low angular rates The non linear model for low angular rates is similar to the near-constant velocity model, but integrates as control input the inertial readings of accelerometer and gyroscope. The kinematics of the vehicle are predicted using the following simple model:

$$x_{t+\Delta t} = f\left(x_t, \Delta t, u_t\right) = \begin{bmatrix} p(x)_t + v_t \cdot \cos(\theta_t) \cdot \Delta t \\ p(y)_t + v_t \cdot \sin(\theta_t) \cdot \Delta t \\ v_t + a_x \cdot \Delta t \\ \theta_t + \omega_z \cdot \Delta t) \end{bmatrix} \tag{5.11}$$

Non-linear model for turns During turns, the prediction function switches to an adaptation of Ackermann steering model (Dwiggins, 1968) for four wheeled vehicles where the two frontal can turn. According to this model, a vehicle with its wheels turned a fixed angle will describe a circle.

The radius of this circle can be calculated as the quotient between the linear speed of the vehicle and its angular rate $R = v/\omega_z$. This radius is the criterion for selecting between models. The model for turns described here is selected when $R < 100m$, because the expected errors of the simple model will be exceeded by those of the sensors and other estimation processes.

As both position and kinematic data are referred to the location of GPS/IMU sensors, we consider the radius of the turning as the distance between the center of the rotation and the

IMU. The origin for the rotation can be calculated as:

$$P_{rot} = \begin{bmatrix} p(x) & p(y) \end{bmatrix}^T - \vec{x} \cdot L + \vec{y} \cdot R \tag{5.12}$$

Where L is the distance between the IMU and the center of the rear axis ($L \approx 1.3m$ in test vehicle), R is the radius of the described circle, and $\vec{x}, \vec{y}$ are unit vectors following the direction of IMU local axes. These vectors can be expressed in world-reference coordinates using vehicle azimuth angle, $\vec{x} = [\cos(\theta), \sin(\theta)]$, $\vec{y} = [-\sin(\theta), \cos(\theta)]$.

The new location of the vehicle is the result of rotating the old one $\omega \cdot \Delta t$ radians around P_{rot}, this is:

$$P_{loc} = \begin{bmatrix} (p(x)_t \\ p(y)_t \end{bmatrix} - P_{rot} \tag{5.13}$$

$$\begin{bmatrix} (p(x)_{t+\Delta t} \\ p(y)_{t+\Delta t} \end{bmatrix} = \left(\begin{bmatrix} \cos(\omega_z \cdot \Delta t) & -\sin(\omega_z \cdot \Delta t) \\ \sin(\omega_z \cdot \Delta t) & \cos(\omega_z \cdot \Delta t) \end{bmatrix} \cdot P_{loc} \right) + P_{rot} \tag{5.14}$$

Where the old position $[p(x)_t, p(y)_t]^T$ is then first translated so that the origin of the rotation P_{rot} is located at $[0, 0]^T$, then a rotation matrix is applied, and the result is translated back to world-referenced coordinates.

Speed and attitude are calculated as in the first formulation,

$$v_{t+\Delta t} = v_t + a_x \cdot \Delta t \tag{5.15}$$

$$\theta_{t+\Delta t} = \theta_t + \omega_z \cdot \Delta t \tag{5.16}$$

5.2.4 Context variables. Description and acquisition

This application uses four context variables, summarized in table 5.2.

Table 5.2: Summary of context variables used in the ground vehicle navigation solution.

Variable	Type	Values
Vehicle motion condition	Inferred	Stopped, Motion straight, Motion turning
Vehicle environment	Inferred	Open road, Urban, Underground
Smartphone energy policy	Provided	Critical, Low, Normal, Plugged
Smartphone placement	Inferred	Resting, On-hand, Unknown

The motion condition variable is directly related with the state to be estimated, but we decided to classify it as context for two reasons: first, because it has a higher abstraction level compared with the rest of the data (symbolic vs. numeric) and their categorization can

require setting an arbitrary (subjective) threshold. Second, it is *optional*: fusion algorithms do not actually need to know if the vehicle is stopped or turning, this information is used to improve some aspects of the estimation process, as in the dynamic correction of sensor drifts and estimation of vehicle pose.

This section details how context is generated and acquired. The uses of these context variables are detailed in next section 5.2.5.3. Motion condition is split into two separate aspects: if the car is stopped, and a trajectory analysis for deciding if motion follows a straight trajectory or a curve.

Detect car stops The selected stop detection algorithm is solely based in IMU readings, to improve the availability of the service –GPS is subject to outages—. Although using the car acceleration values for this task could seem the most suitable approach, it is important to notice that this sensor is subject to several effects that can cause some algorithms to fail. First, accelerometer measures can be biased, and those biases are usually unknown to us –our proposal tries to correct those effects online—. In second place, if the vehicle is not perfectly stabilized then part of the gravity acceleration is usually transmitted to X^L and/or Y^L axes of the accelerometer.

Both problems together makes very difficult (if not impossible) to estimate if the acceleration of the car is zero at some point. This can be seen in Figure 5.6, especially in axis X^L readings, where the acceleration registered in the stops differs in $0.9m/s^2$ because in the second stop the car is on a steep road (approx. 5 degrees).

However, a significant amount of vibrations are transmitted to the IMU while the vehicle is in motion, even over the smoothest terrains. The three axes show a near constant reading during the time spans where the car is not moving.

Thus, this sub-module uses the amplitude of accelerometer measures in a window of time –maximum measured value minus minimum— as an indicator of a "flat" reading. The advantages of the proposed method include:

- Simple and fast computation (maximum/minimum of a reduced amount of numbers)
- Independence of biases, attitude and other conditions

However, it requires setting a suitable threshold for determining when the car is stopped ($0.5m/s^2$ in our case). Although this can be easily set taking into account the expected random error according to the manufacturer, more robust alternatives can use GPS position estimates to enhance the stop detection and learn this threshold online.

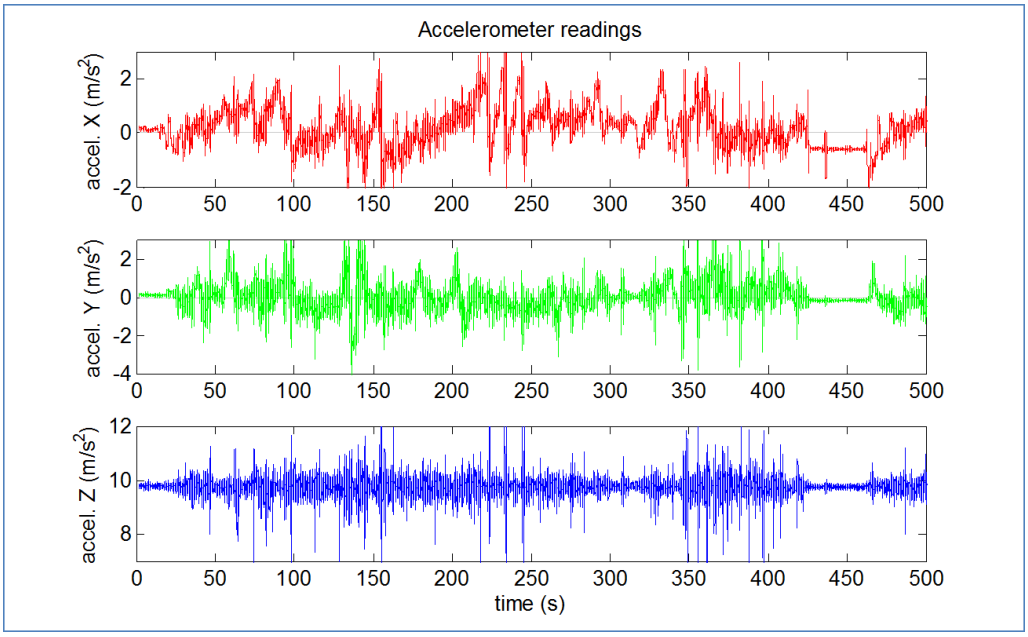


Figure 5.6: Sample accelerometer readings featuring two stops around t=[0;15] and t=[430;460] seconds. Bias combined with gravity effect makes the raw signal not adequate for detecting stops.

Trajectory analysis Our system features a robust algorithm for detecting certain trajectory features that have a special meaning. In this work, the implementation has been limited to detecting where the car is driving over a lane in a straight fragment of the road, with no lateral or longitudinal maneuvers. As well as it happened with car stops, this information can be useful for additional fusion or sensor refinement processes, although it can be applied to more advanced reasoning processes.

Turns are clearly indicated by the gyroscope yaw reading (around Z^L axis), tightly related with changes in the azimuth of the vehicle. Detection of straight movement takes the average gyro reading in a window of time (typical values range from 0.5 to 2.0 seconds), and determines a straight movement when the absolute value is under a threshold.

This detection requires, however, a non-biased gyroscope reading which is provided by the sensor refinement module. The details are provided in next section.

Environment detection This context variable can be extracted directly from GPS quality measures: number of satellites and accuracy. Its evolution over time can be used to provide an stabilized estimate that can be used to adapt the navigation solution. We have defined three values:

- Road: more than 6 satellites and accuracy within 2 times the smallest expected error (around 1.5 meters for the Novatel receiver, 4 meters for the smartphone).

- Urban: Between 2 and 5 satellites, or error larger than double the smallest expected error.
- Underground: Less than 2 satellites, or no valid fix available.

Switching between modes requires several GPS updates with a different category. Current value is set to 5 seconds. This value and the criterion are subject to change, although this refinement is out of the scope of our experiments.

Smartphone energy policy Even when the smartphone is available, its battery level can discourage its use as a set of sensors. We have defined the following levels of energy:

- Plugged: independently of the actual battery level, it is connected to a charger. Can be used by the navigation system.
- Normal: allows a normal use of the smartphone for the next hours. Can be used by the navigation system.
- Low: discourages its use as set of sensors, except in special cases when the adaptive fusion system cannot find suitable alternative configurations.
- Critical: cannot be used under any circumstance. Battery reserved for other uses of the phone.

While the three previous context variables had to be inferred from sensor readings, Energy policy is extracted directly from the (numeric) battery level of the device and its charge status. Both elements are provided by Android OS API. The above categories are calculated as in table 5.3, but they can be redefined as desired.

Table 5.3: Rules for determining energy policy.

Energy policy	Battery level	Charging
Plugged	any	Yes
Normal	$level >= 50\%$	No
Low	$20\% <= level < 50\%$	No
Critical	$level < 20\%$	No

Smartphone placement Smartphone inertial sensors readings are referred to the device local reference system. We want to use it for determining vehicle kinematics, so they have to be transformed to vehicle local reference system. This is only possible if the relative position and attitude of both elements is fixed. Context variable "placement" gives useful information about this fact.

This variable is a mix of inferred and provided context. Table 5.4 describes its calculation. As a rule of thumb, inertial measures will be used only when device is resting. Any other value disregards its integration into the solution.

Table 5.4: Rules for determining smartphone placement

Accel./Gyro. indicator	Screen	Placement
Still	Off	Resting
Still	On	Unknown
Motion	Off	On-hand
Motion	On	On-hand
Undetermined	Off	Unknown
Undetermined	On	Unknown

The accelerometer/gyroscope placement indicator is based on discerning characteristic patterns of vehicle motion from the rough, less stable readings related with a user having the phone in their hands. An example of activity recognition from inertial readings can be found in (Blázquez Gil et al., 2012a), but literature is plenty of examples as (Antos et al., 2014).

5.2.5 Using context to improve sensor fusion

This section details the expected usage of context in the sensor fusion system. We have tried to cover all the applications of context information supported by the framework, in order to test its aptitude.

5.2.5.1 Online sensor calibration

Sensors are subject to external and internal conditions which affect their performance in different ways. Sometimes, the sensor itself will be capable of providing a self-assessment of its observations quality, as in the case of GPS measures that include the number of available satellites or the precision of the calculated solution.

For some others, however, it is necessary to apply external checks for detecting degraded performance or faulty sensor conditions. An example can be found in inertial measures: accelerometer and gyroscope readings are subject to a systematic deviation known as bias. In micro-electronic based devices, such as the one used for these experiments, this deviation is is stable at short/medium term, but suffers a slow drift related with factors such as the internal temperature of the device.

This means that an initial calibration is not enough for keeping a system running over long periods of time. The best solution involves monitoring the quality of sensor readings, and calculating the parameters that correct them when possible.

In the original proposed system, sensor refinement is understood as a layer between the sensors and any other process accessing their data (Figure 5.7). Some of the processes depend on information that is only available in later components, in particular:

- Transforming local IMU readings to world frame of reference requires an estimation of car attitude
- The algorithm for correcting gyroscope bias works is active when the car is not moving

This can be solved by introducing a feedback from latter layers. It must be checked that there are no cyclic dependencies or that even under them the system will converge to a stable solution.

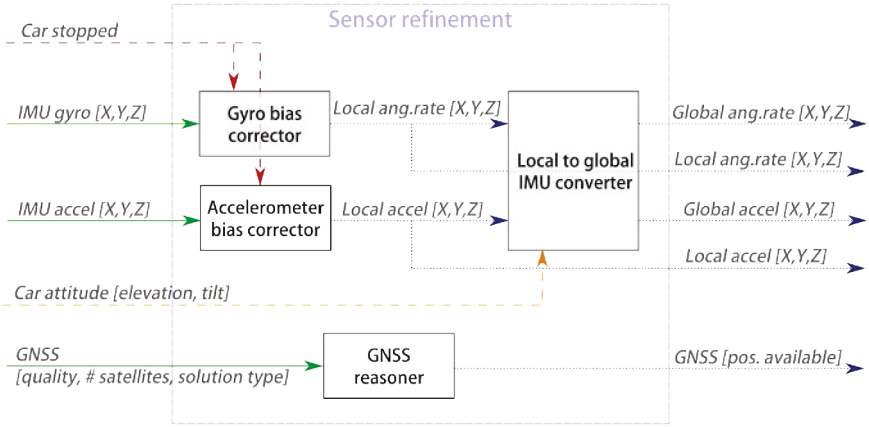


Figure 5.7: Sensor refinement module.

Gyroscope bias correction Gyroscope bias can be corrected when the car is stopped: the reading on each axis is its bias, plus a random perturbation which can be modeled as Gaussian noise. Therefore, we can get bias is estimated as the average of the readings on each axis over the period where the car is stopped (the reduction in noise variance will be inversely proportional to the number of samples). So, the key to update biases in gyroscopes will be the good inference of contextual information, triggering the process each time the car is stopped.

Accelerometer bias correction The case of accelerometer bias is much more complicated than the gyroscope, due to the effect of gravity. With the vehicle stopped, the reading on the

accelerometer axes is:

$$acc_{\{x,y,z\}^L} = b_{\{x,y,z\}^L} + R^T \cdot (g \cdot \vec{z}) \tag{5.17}$$

Where $b_{\{x,y,z\}^L}$ are the biases of the three local accelerometer axes, g is the magnitude of the acceleration due to gravity, $\vec{z}$ is the unit vector which follows the direction of Z axis in world coordinates and R is a rotation matrix that express the attitude of the IMU (R^T transforms from world-referenced to local-referenced coordinates).

According to the equation above, bias and rotation have to be determined simultaneously. Solving this problem with no prior calibration represents a challenge, since degree-level errors while determining the attitude can be compensated by drifting the estimated accelerometer bias in a reasonable quantity.

Bias estimation is restricted in our model to the X^L axis of the accelerometer, since this reading is the one used in the kinematic model of the vehicle. It is possible to see that, if car attitude R is expressed as global Euler tern (tuple of 3 elements), then azimuth angle can be safely ignored. Let us parametrize the attitude of the IMU using only tilt and elevation angles, $R \rightarrow R_{\psi,\varphi}$.

Applying the same reasoning process followed in (Rezaei and Sengupta, 2007), instant accelerometer reading for X axis when the vehicle is stopped can be written as $acc_X = b_a(X) + g \cdot sin(\varphi)$, which leads to.

$$b_a(X) = acc_X - g \cdot sin(\varphi) \tag{5.18}$$

Bias is estimated from accelerometer readings during stops, subtracting the expected effect of gravity according to the estimated pose of the IMU.

5.2.5.2 Context-aware fusion algorithms

We have defined two algorithms that use context information to improve the result of the fusion: the UKF for attitude estimation, and the UKF for kinematic estimation

We have described in 5.2.3.1 Attitude estimation uses the motion condition of the vehicle (detected stops) as a marker to take absolute estimations of its tilt angle, based on the direction gravity vector. Thus, the formal specification of this algorithm will require as input the context variable that indicates if vehicle is stopped.

The Unscented Kalman Filter for kinematic estimation uses the straight motion indicator to select the best prediction model. Stops are also taken into account to interpret inertial measures differently and avoid a noisy output.

5.2.5.3 Adaptive fusion solution: sensor and algorithm selection

For this work, we implemented a simple Fusion Adaptation module that loads the specification of information sources and data processing nodes, and generates a valid fusion solution that provides the required outputs.

The last element in the figure is the Inference process, in charge of combining the available solution elements to create the sensor fusion system expected to deliver the best performance in the present conditions. It selects which sensors and algorithms are used, and how they are connected. The optimal implementation depends on the domain of application and the considered factors, e.g. contextual information and solution quality indicators. For this work, we have chosen an event-triggered search process, where the terms of the search are affected by contextual factors. The process is as following:

Event processing Fusion adaptation module receives an event. We have defined the following events: (a) Some sensor is no longer available (b) A new sensor is available (c) Some context variable has changed (d) The list of desired fusion products has changed.

Determine reach of the event The inference process determines how the received event affects the elements of the solution. The effect can be direct, e.g. if a sensor is down, the equivalent Data Node has to be marked as not available so that it is not used in a solution, or indirect, e.g. in the selected case of use, some fusion algorithms cannot be used if the vehicle is moving underground.

This information is ideally described as constraints or rules in the different ontologies used to describe the problem. An inference process can be used for generic constraint reasoning. Previously in this paper, we referenced the work (Gómez-Romero et al., 2015), which defines rules using SWRL to reason directly over the domain ontology. Ontology-based reasoning is, however, a computationally expensive choice. In this scenario, we overcame this problem using the Drools rule-based system (de Ley and Jacobs, 2011): this library, written in Java, provides a fast inference engine (implements the RETE algorithm). Rules can be defined in text files that can be loaded dynamically, and the inference engine can use Java objects of the target application as facts for the knowledge base.

The rules used in this case, shown in 5.5, have direct translation between SWRL and Drools languages.

Compose sensor fusion solution Once the elements of the solution have been modified according to the events, a new sensor fusion solution is composed. For this work, we chose to restrict valid solutions to a tree: a directed graph with no loops. The list of desired fusion

Table 5.5: Contextual constraints on sensor set and algorithms

Sensor	**Conditional**	**Context**
smartphone (any)	INCOMPATIBLE	EnergyPolicy == LOW
" (any)	INCOMPATIBLE	EnergyPolicy == CRITICAL
" accel.	REQUIRES	Placement == RESTING
" gyro	REQUIRES	Placement == RESTING
Algorithm	**Conditional**	**Context**
KFKinematic	INCOMPATIBLE	Environment == UNDERGROUND
UKFAttitude	INCOMPATIBLE	Environment == UNDERGROUND

products is fed into the system as a data node that consumes data without producing any output. It is the root of the tree.

Solutions are composed through back-chaining, guided by a search algorithm that follows a depth-first strategy. The leaves of the tree will be sensors (pure information sources), that produce data without requiring any input. In many cases, there are several valid solutions for a given set of conditions. Our implementation determines which is the best solution using a set of rules which determine the suitability of each solution under different contexts. It comprises basic checks such as avoiding self-connections and loops, as well as more advanced criteria as maximization/minimization of numeric indicators –energy consumption, accuracy score.

5.2.5.4 Summary of adaptive solution

Once we have defined the elements of our solution, identified relevant contextual facts and the desired adaptation mechanisms, we have a clearer picture of the whole system. Figure 5.8 shows the adaptive navigation solution, instantiated in the proposed architecture.

The figure specifically includes an external input specifying the goals of the system. These goals are expressed as a list of wanted data products, this is, data elements with features expressed using the terms of the ontology (quality, update frequency, etc.).

Context inference mechanisms have been represented as a set of Data Nodes in the Data Fusion module that send results to context repository. The question of where this inference mechanisms should be placed is an interesting matter of discussion, and with this purpose we show the *Energy Policy* node duplicated in the smartphone sensor set. Its actual location depends on the semantics of that context variable: if the values of energy policy can be calculated according to the same criteria for several different applications and problem domains, then it is interesting to keep this node in the resource set as a virtual sensor. However, if the categorization is exclusive for this navigation problem, then it is right to place the node in the Data Fusion module and calculate its value using *Battery* and *Charging status* values.

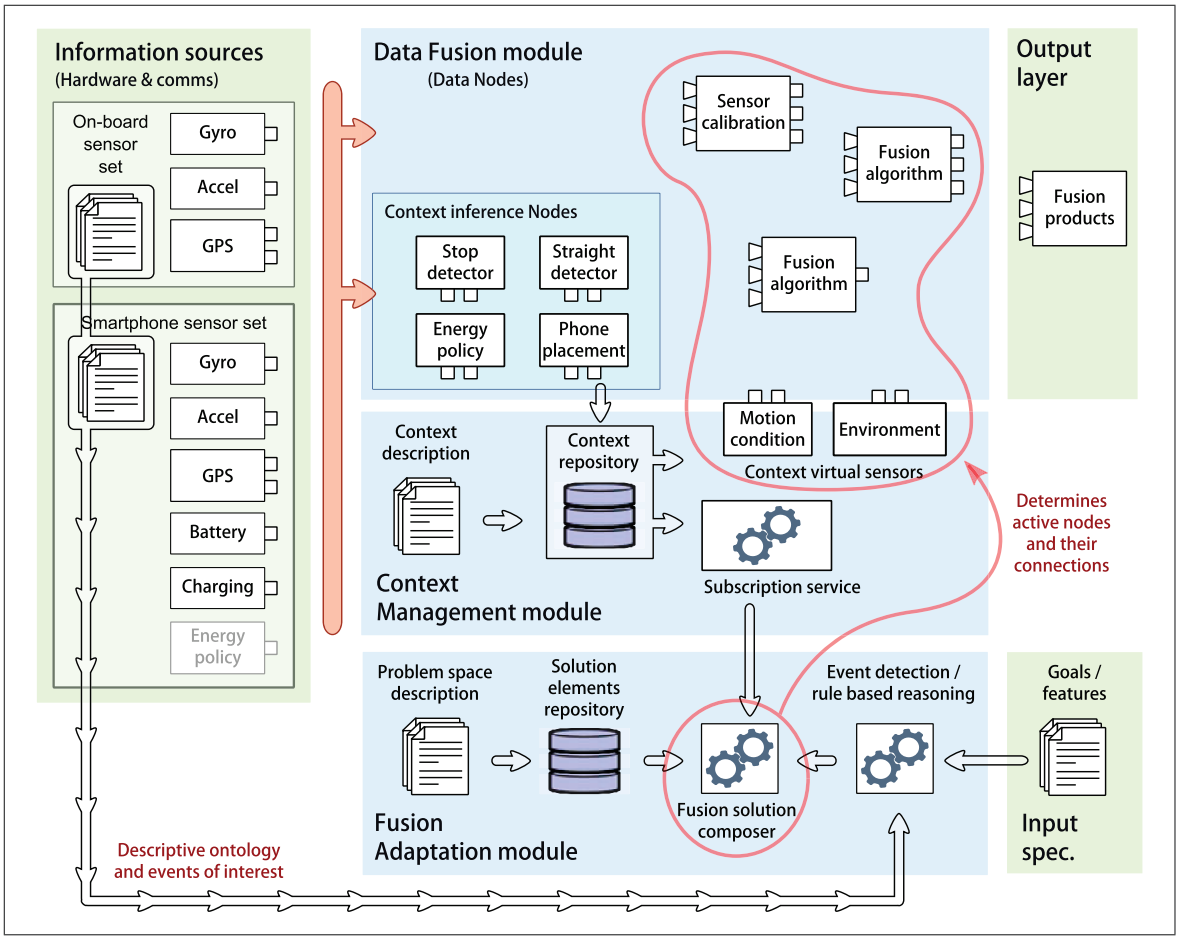


Figure 5.8: Instantiation of the designed adaptive navigation solution into the proposed framework

As an additional consideration, the real implementation connects context inference nodes directly to consumers, skipping the repository part. We decided to do that because the nature of the contextual information used in this problem is instantaneous and keeping past values is superfluous. However, this is just an implementation detail that can be changed as necessary.

5.2.6 Formal model of the system

The system relies in a formal model of Problem Space description, the set of sensors (resource sets) and context. This model serves as a precise and shareable description of the system features, but is also used by the defined adaptation processes.

We are going to build the formal model of the system over the sample ontology suggested in the proposal, section 3.3.2.1, adequately adapted to the particularities of this problem.

5.2.6.1 Problem Space description

The problem space description, in this application, regards the elements that will populate the Data Fusion module: Data Nodes (repository of algorithms), the data elements managed by those Data Nodes, and the relevant features of these entities. This description is the base to define the logics that will guide the automatic construction of fusion solutions by the Fusion Adaptation module.

Classes The following description is built over an extension of the basic ontology described in the proposal (section 3.3.2.1). We defined three basic classes: *DataNode*, for every piece of logic that produces and/or consumes information (i.e. algorithms and sensors); *DataType*, for the basic types of data managed by the system; and *DataProduct*, an auxiliary class for attributing data types with features are properties relevant to the problem.

Since the Fusion Adaptation module give sensors (defined in the resource set) and algorithms an homogeneous treatment, both have been defined as data nodes. However, to allow a better categorization, this class has two subclasses, *Operation* and *Sensor*. The repository of algorithms is composed by individuals of the subclasses of *Operation*:

- 1 Data Conversion algorithm: transforms GPS fixes (latitude-longitude using the WGS84 ellipsoid model) to local Cartesian coordinates referred to a given origin.
- 3 Fusion Algorithms:
  - Kalman Filter that uses position measures to estimate position and speed of the vehicle.

  - Unscented Kalman Filter that calculates the attitude (orientation) from the position and the angular rates of the vehicle.
  - Unscented Kalman Filter using the position, angular rate and acceleration of the vehicle to estimate its full kinematics. It combines the two non-linear models described in section 5.2.3.2.

- 1 Sensor Corrective Action: estimates and compensates the bias of the gyroscope. It requires knowing when the vehicle is stopped, and the raw (biased) angular rates.
- 2 Virtual Sensors for context data: a stop detector and a turn detector. The availability of these sensors is determined automatically from the Context description ontology (see next section). There is a potential virtual sensor for each context variable, but they have not been integrated in the solution search process because are not useful as inputs for fusion algorithms.

These algorithms have been inserted as individuals in the hierarchy of classes reproduced in figure 5.9.

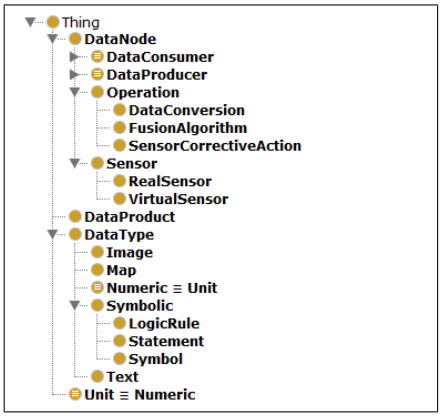


Figure 5.9: Hierarchy of classes in the problem-space description ontology

The data types can be extracted just considering the input and output data of every sensor and algorithm in the system. We found 9 numeric types: *acceleration3d*, *altitude*, *angularRate3D*, *attitude3d*, *batteryLevel*, *latLon*, *linearSpeed*, *position3d* and *speed3d*. The system does also define 3 logical statements: *isPhoneChargingStatement*, *isStoppedStatement* and *isTurningStatement*, related with the equivalent contextual variables.

The use of DataProduct will be shown later, after detailing the object and datatype properties for the problem space ontology.

Object Properties This problem uses the same object properties defined in section 3.3.2.1 (reproduced here in figure 5.10) which serve as a description of how components can be connected.

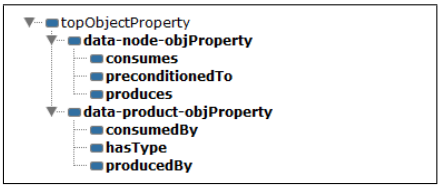


Figure 5.10: Object properties in the problem-space description ontology

Datatype Properties Reproduced in figure 5.11, allow to add literal values to individuals. Following the example of Object Properties, We have defined three categories depending on the domain class.

The first category is represented by the super-property *data-node-dataProperty*. Fusion Adaptation process works over active data nodes, so we have a property to define such status: *isActive*, with range over boolean data type. Due to smartphone battery restrictions, solution configuration logic is also aware of the power consumed by sensors and, optionally, algorithms. Thus, we have defined *energyConsumption* property, with four possible values: high, medium, low and none.

The super-property *data-type-dataProperty* contains two data properties: *cardinality*, describing the size (number of elements) of a data type individual, and *basic_type*, that can be used to indicate the type of its underlying data elements. These properties describe inherent features of a type, but cannot be used to give information about the concrete features of the data produced by a certain node.

This last use is responsibility of DataProduct elements, which are associated with properties defined under the general *data-product-dataProperty*. We have identified 4 useful properties: *isBiasCorrected* applies to inertial measures, *isFiltered* is a general statement to identify data elements produced by a filtering algorithm, *quality* is a rough categorization in three levels (high, medium, low) for the expected accuracy of sensor raw data, and finally *updateFrequencyInHz* can be used to express the capacity of sensors and also the requirements of algorithms regarding data rate (this information can be useful for parameterization purposes).

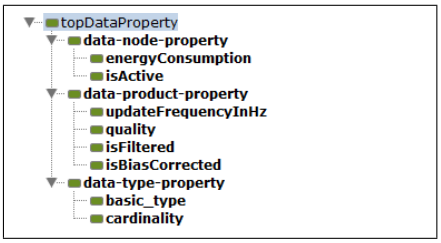


Figure 5.11: Data properties in the problem-space description ontology

5.2.6.2 Resource sets

This problem makes use of two difference resource sets, one for the sensors of the on-board platform, the other for the smartphone. We plan to implement an automatic fusion solution configuration process that gives an homogeneous treatment to fusion algorithms and sensors. With this purpose, resource set description takes as base the Problem-space ontology –inheriting useful data types, and object- and datatype properties–. Actual sensors re defined as individuals of the ontology belonging to the Sensor class. The information introduced in the ontology is summarized in table 5.6.

Table 5.6: Available sensors

On-board	**Product**	**Quality**	**Freq.**	**Bias corrected**
Gyroscope	angular rate	medium	100 Hz	No
Accelerometer	acceleration	medium	100 Hz	No
GPS	Lat-Lon	medium	5 Hz	
	Altitude	medium	5 Hz	
Diff-GPS	Lat-Lon	high	5 Hz	
	Altitude	high	5 Hz	
Smartphone	**Product**	**Quality**	**Freq.**	**Bias corrected**
Gyroscope	angular rate	low		
Accelerometer	acceleration	low		No
GPS	Lat-Lon	low	1 Hz	
	Altitude	low	1 Hz	
Battery sensor	battery level			

As stated above, sensors are represented of individuals of the *Sensor* class, Their data is expresses as DataProduct individuals annotated with the correct datatype properties.

5.2.6.3 Context model

Context variables have been defined in the ontology following the specification described in 5.2.4. With that purpose, we have defined four classes grouped by the entity they are referred to, as shown in figure 5.12.

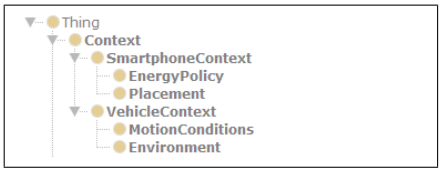


Figure 5.12: Context description ontology for the ground vehicle navigation experiment

Since context variables are symbolic, each class is populated with individuals representing those values. Figure 5.13 shows an example for the smartphone energy policy context variable.

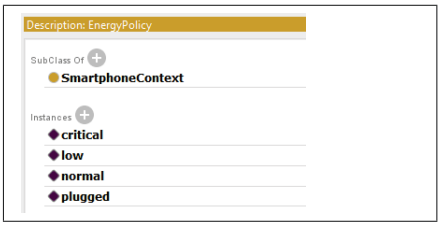


Figure 5.13: Values of the EnergyPolicy class are defined as individuals

Finally, figures 5.14 and 5.15 exemplify how ontology elements are combined to describe the problem space. The first figure shows the algorithm and sensor repository after it has been populated with all the elements defined for this solution. The other figure illustrates how DataProduct is used to annotate sensors and algorithms with their requirements and products as attributed data.

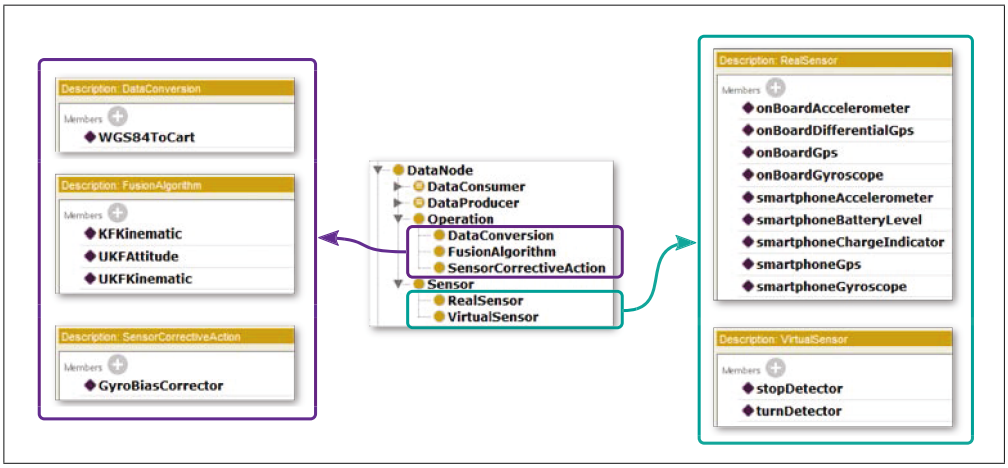


Figure 5.14: Classes are populated with individuals representing objects the Fusion Adaptation module can manage

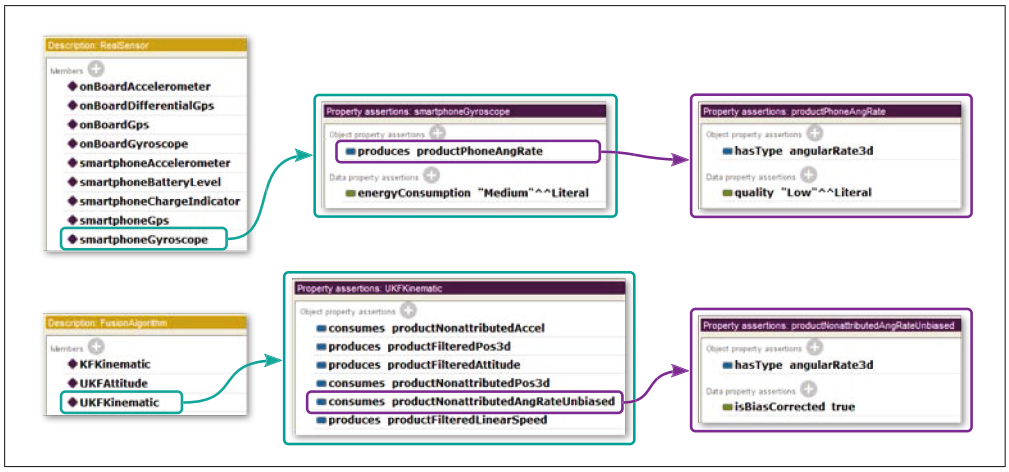


Figure 5.15: Annotating individuals with object and datatype properties allow to describe the required problem space information

5.3 Experiments

5.3.1 Context-aware fusion algorithms using on-board sensors

The experimental validation has been carried out in a set of representative scenarios to show the reliability of the proposed system. In the first place, we present some results about contextual analysis and sensor correction subsystems. The other results display the performance of the filters when GNSS signals are unavailable or severely degraded in complex urban environments.

5.3.1.1 Evaluation of context reasoning and sensor correction subsystems

The stop detection algorithm is based in measuring the "roughness" of accelerometer output over time. For this purpose, a window of 0.5 seconds proved to offer good results without introducing a significant delay. Moreover, stops are useful when extend over a few seconds, so that the delay is not usually important. Figure 5.16 shows the performance of the selected algorithm over the second stop of previous trajectory, demonstrating its validity even with biased inputs.

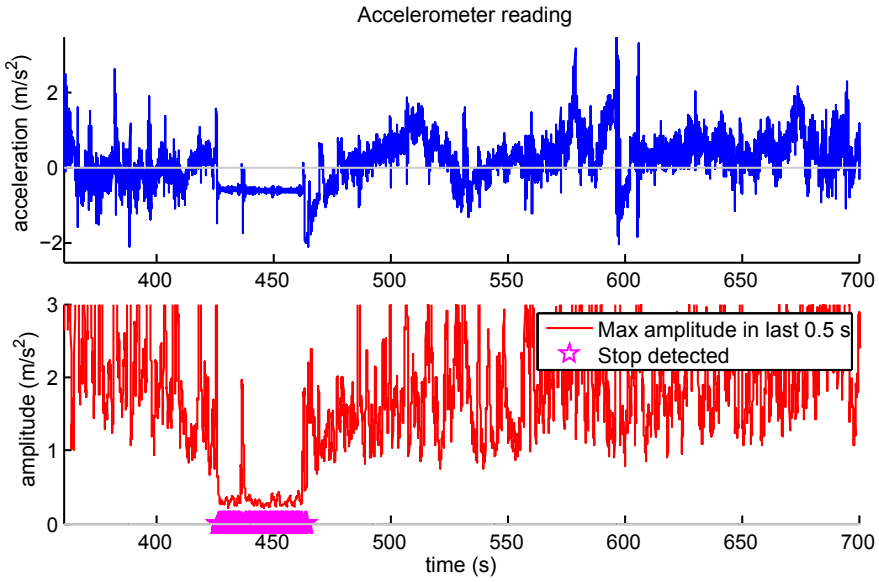


Figure 5.16: Sample accelerometer readings, processed signal, and output of the car stop detection module. This figure shows the validity of the applied strategy.

For the trajectory analysis part, the non-biased gyroscope reading around Z global axis is used. This data element is provided by the sensor refinement module. In Figure 5.17 is possible to see the raw output of the thresholding criterion that determines when the car is following a straight motion pattern. The selected limit of ± 0.5 degrees per second does not

completely guarantee a straight movement, but it rather indicates that the readings of the other sensors will not be affected by some of the effects of curves, e.g. car inclination and lateral accelerations. A further refinement has been implemented by interval analysis to discard fragments shorter than a few seconds, to avoid those detections between linked turns.

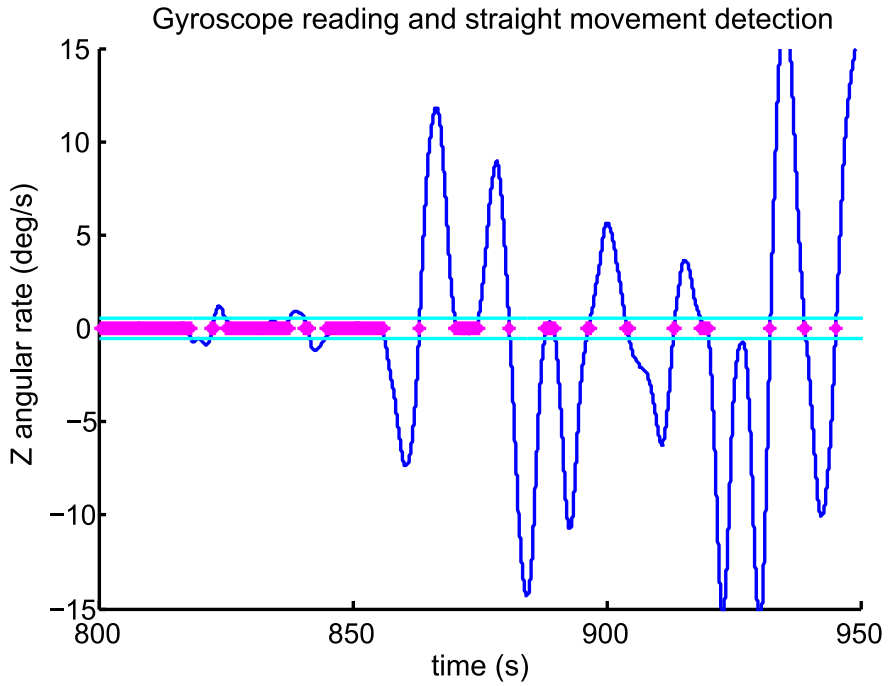


Figure 5.17: Output of the trajectory analysis module: straight movement detection using accelerometer readings.

It is interesting to see the two small interruptions of the straight movement around $t = [820; 825]$ and $t = [840; 845]$: they represent two consecutive changes of lane, the first one to the right and the second back to the left. Readings from $t = 860$ in advance are part of a curvy mountain road with brief straight segments, revealing a satisfactory performance even with strong slopes.

With respect to bias drift, in this type of sensors is known to be caused by temperature changes, and thus is a slow process. The 15 stops detected in the experiments returned a quite stable estimation of $b_g = [0.29, -0.31, 1.05] deg/s$, with a variance $var(b_g) < [0.01, 0.01, 0.02] deg/s$. This can be explained because all the records were taken in the same day, starting half an hour after the device was mounted and exposed to direct sun light (temperature variations are known to affect the stability of the bias).

The use of dynamic adaption would have kept the gyroscope calibrated under any other conditions. Taking into account these considerations, two indicators of algorithm performance were examined:

Table 5.7: Algorithm for estimating the error of calculated elevation angle from two GPS fixes.

Input	2 GPS measures
Output	Estimated error of elevation angle

- Accuracy of dead-reckoning navigation.
- Bias estimation process should return similar values for car stops that are close in time, but for which the elevation angle is different.

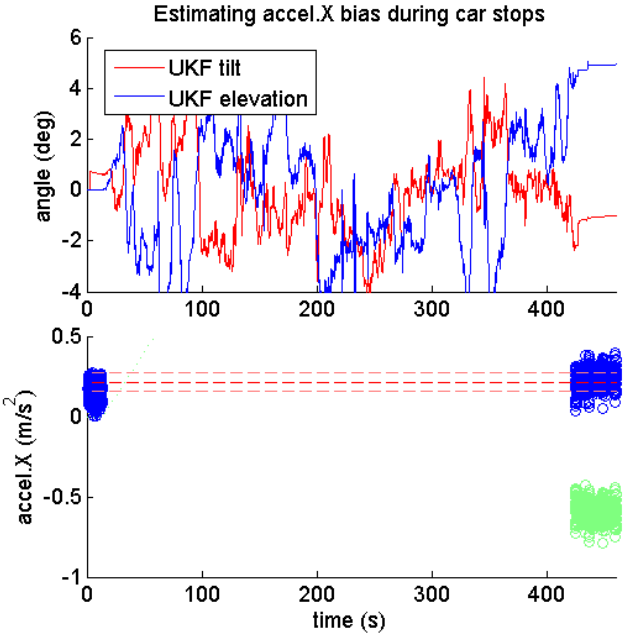


Figure 5.18: Accelerometer bias can be corrected during stops if elevation angle has been already determined.

The second point is illustrated in Figure 5.18, where the bias estimation process returns an average value of $0.21 m/s^2$ (blue circles in lower part) after correcting the effect of gravity according to the estimated car elevation, which is close to zero in the first stop and close to $5 deg$ in the second one. The green circles during the second stop represent the raw accelerometer reading, before correcting gravity effect.

Regarding the estimation of car elevation angle from GPS positions, the expected error –depends on the random position error of the two consecutive GPS fixes– is difficult to describe analytically, and is clearly not distributed as a Gaussian. The selected solution involved a Monte Carlo simulation that describes the probability distribution function of the error. Its second order statistic (variance) was calculated, for getting an approximate Gaussian description of such error. The detailed procedure is described next:

Figure 5.19 shows the expected standard deviation of the calculated elevation angle

depending on the speed of the car and the accuracy of the GPS device. For a goal of under-degree level error, the conditions have to be near optimal, with a good GPS accuracy (standard deviation in the three axes around 1 m) and the car traveling at a high speed (over $10-15m/s$).

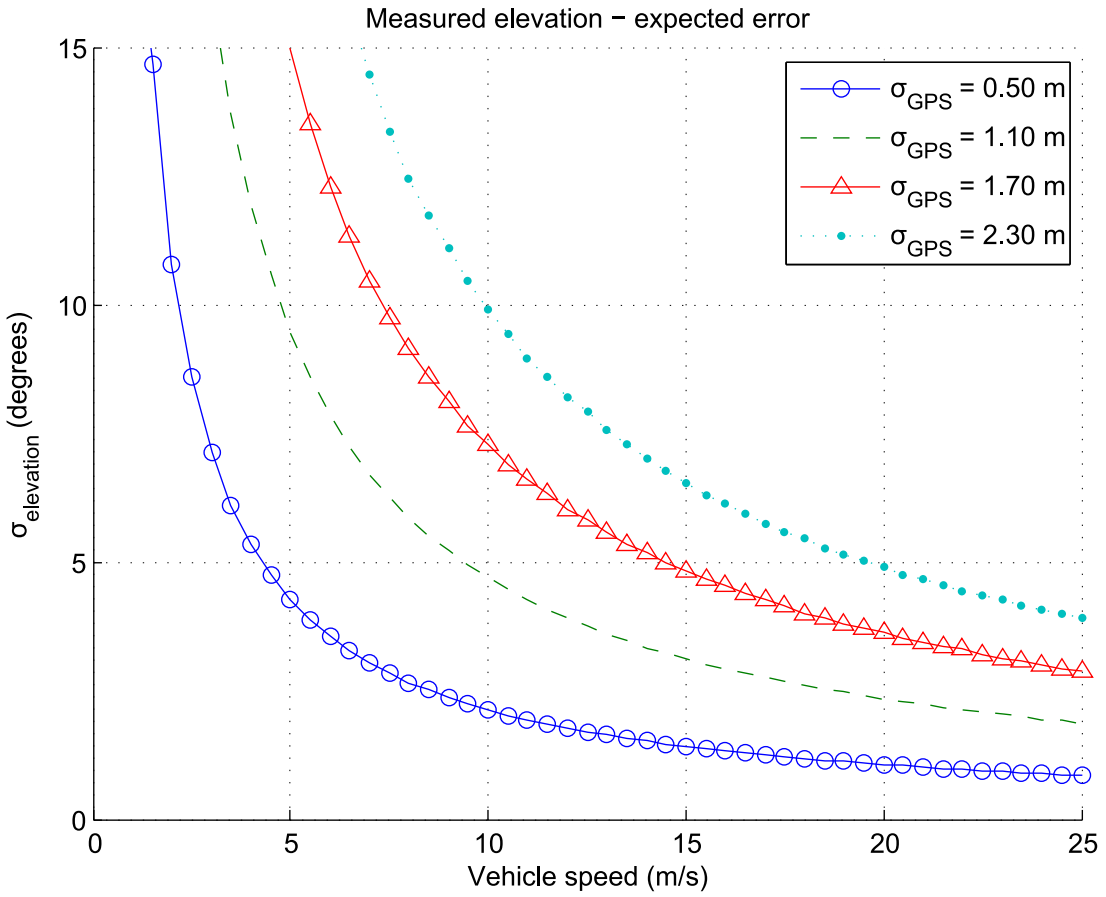


Figure 5.19: Expected standard deviation of GPS-obtained elevation angle, depending on vehicle speed and fix horizontal accuracy (simulation, 10 million iterations per point).

With respect to the whole navigation system, several complex scenarios have been selected to assess the overall performance. These experiments include typical cases as stops or turns in urban environments, enriched here with especially complex cases such as roundabouts with different exits, turns in the banked road at mountain pass, underground parking areas, long tunnels, driving under elevated pedestrian bridges, or short tunnels under motorways to change direction.

The first complex scenario includes a total GPS blackout in a non-underground parking. The calculated position by GPS appears as a constant value whereas the vehicle is passing through the parking area, which is not underground but has a roof that occludes the satellites (Figure 5.20). The standard deviations show a high value in the middle of the graph corresponding to

this situation of non-available solution at the rover receiver. The right graph displays this effect of inactive DGPS mode maintaining a constant value of differential age and zero satellites used in the solution, when the vehicle is passing through the non-underground parking. It can be observed at the parking exit that the receiver changes to SINGLE mode when the satellites are visible. The left graph displays a gap of the trajectory caused by non-calculation of the coordinates by the receiver that maintains the last calculation. The GPS blackout has a total length of 56 seconds.

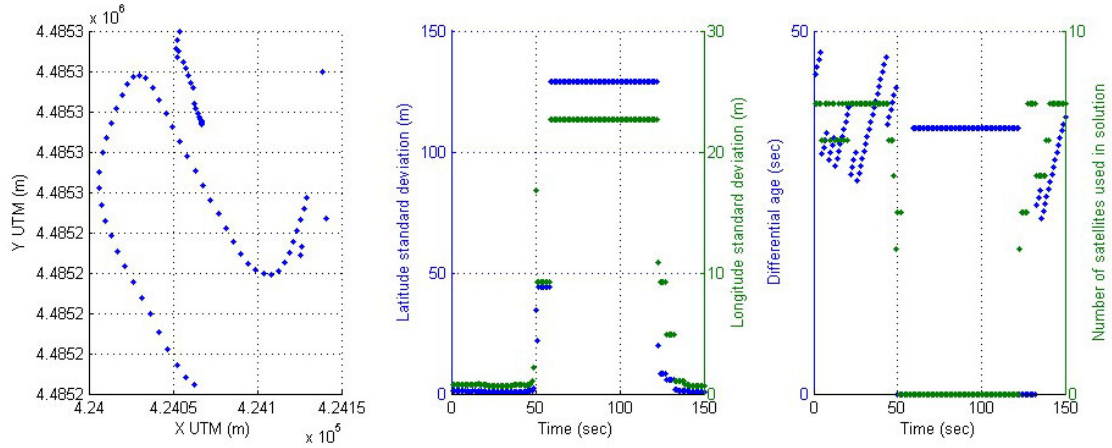


Figure 5.20: Non-underground parking area with zero satellite visibility and inactive DGPS mode using a constant value.

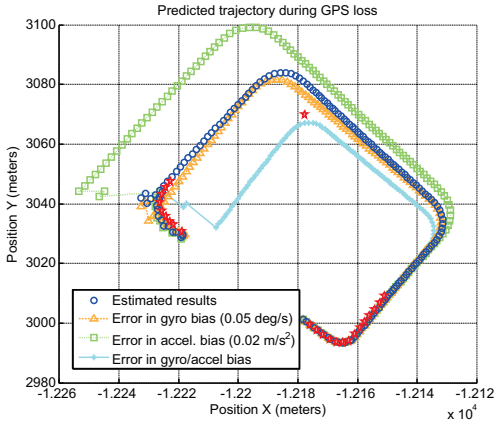
At this point, and after more than 10 minutes running, the system has accurately determined biases. Dead-reckoning conditions are not optimal, though, since at this point the last effective measure of the pitch was received more than two minutes ago, so the attitude has been maintained by the filter integrating IMU measures. Figure 5.21b shows that when GPS signal is recovered (red stars), the positioning error of the filter is around 15 meters (blue circles). For establishing a comparison, three other predictions are shown, corresponding to simpler solutions where the sensors are not dynamically adjusted. They use IMU bias estimation with an error around 0.05 degrees per second for the gyroscope and/or $0.02m/s2$ for the accelerometer.

The estimation for gyroscope bias that the proposed system achieves is stable within 0.02 degrees per second. The error in position caused by the drifting attitude estimation is not very important compared with that of the accelerometer. It is reasonable to conclude that gyro bias estimation is accurate enough in our system. It is different for the accelerometer bias, where an error of $0.2m/s^2$ has a much profound impact. It is worth remembering that residual accelerations of a similar magnitude can appear spontaneously if the vehicle elevation is estimated with a deviation of 1 degree.

As a conclusion, the results on this scenario show that the biases estimated by the proposed system have been set correctly, and that small changes inside the expected IMU bias stability

can be the source of large errors.

(a) Vehicle trajectory



(b) Predicted trajectories

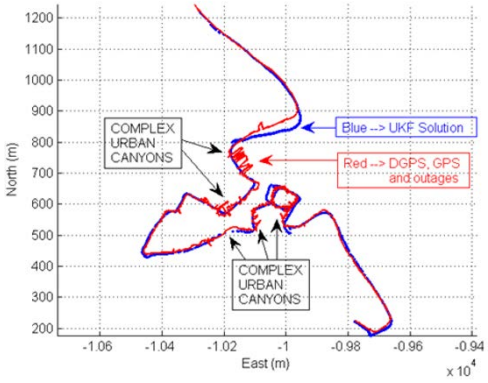
Figure 5.21: Underground parking lot experiment

The second scenario is related to a complex urban environment where the vehicle is passing through urban canyons with low visibility of satellites. The Figure 5.24 shows cases with active DGPS mode, cases with active DGPS and high values of differential ages, cases with inactive DGPS mode and active SINGLE mode solution, and cases with zero satellites used in solution. The trajectory can be observed in the left graph where the vehicle arrives to complex urban canyons, and the rover receiver is changing frequently their mode depending on environment conditions through complex urban environment. The accuracy is recovered at the exit of the urban canyon and this effect is detected at the end of the middle graph. The right graph displays the diversity of cases presented in this experiment, thus it is difficult to obtain optimal conditions in complex urban canyons that can be solved by sensor fusion.

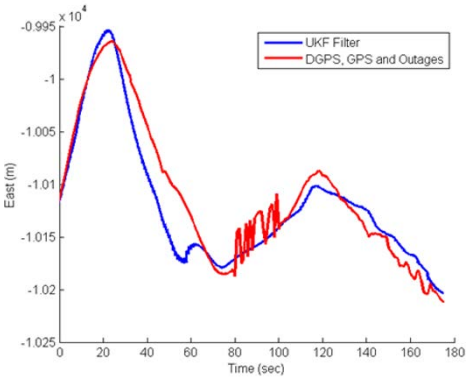
The sensor fusion solution is presented in Figure 5.22a, where red trajectory displays the difficult calculation of positioning by rover DGPS system. Inaccuracies are caused by rover navigation within complex urban area. The estimated solution using UKF filter is blue trajectory. The reliability of UKF solution can be observed in detail for 175 seconds of initial trajectory: figure 5.22b for the east coordinate and figure 5.22c for north component.

The effect of entering in an urban area is displayed in Figure 5.23a. Initially the DGPS trajectory is the same that the UKF filter trajectory, but DGPS inaccuracy appears when the rover is close to big trees and is entering in a soft urban environment. Figure 5.23b displays an increase of DGPS East standard deviation caused to use four satellites in the solution with high differential ages, and Figure 5.23c is the time-domain representation of the standard deviations where the filter shows low positioning errors.

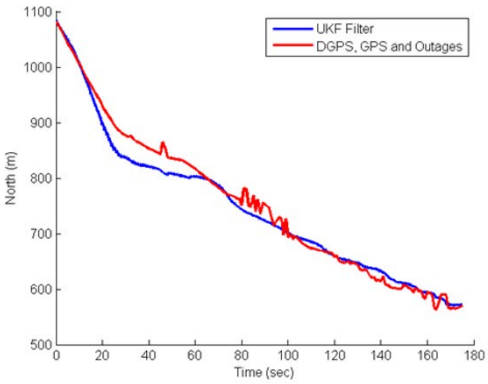
The movement of the vehicle in a complex canyon is displayed in Figure 5.25a, where close buildings cause GPS and DGPS inaccuracies, and outages. The UKF solution is presented in



(a) DGPS, GPS and Outages (red), and UKF solution (blue)



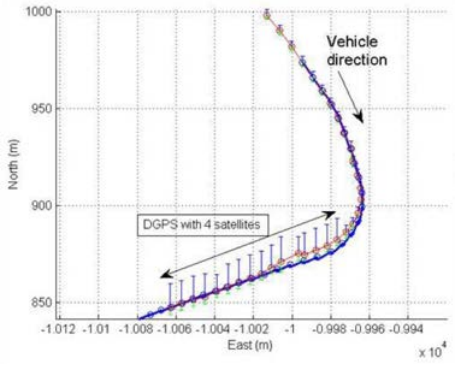
(b) Time-domain detail of East UKF solution



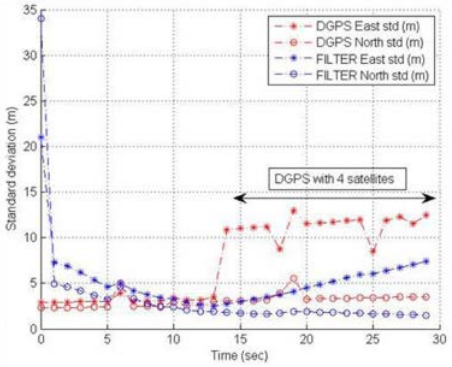
(c) Time-domain detail of North UKF solution

Figure 5.22: Effect of complex urban canyons in UKF performance

(a) Rover trajectory entering in an urban area.



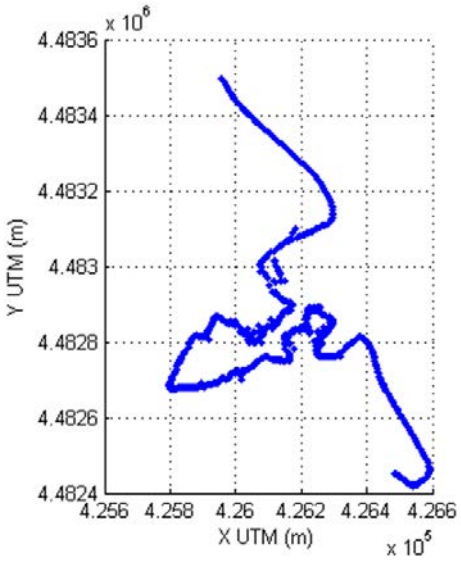
(b) Loss of accuracy in a soft urban environment (DGPS with 4 satellites) and UKF filter solution



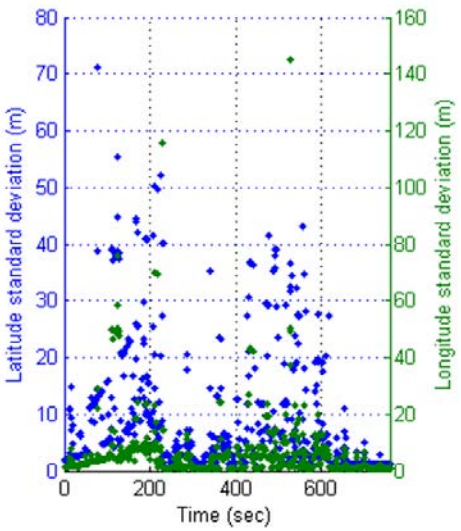
(c) Time-domain standard deviation evolutions (DGPS and filter)

Figure 5.23: Sample trajectory: urban area entrance.

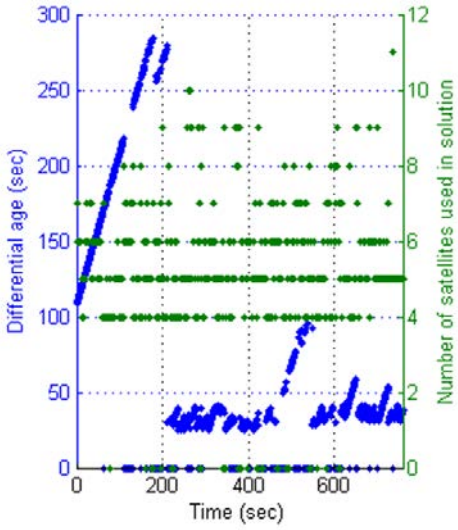
5.25b (blue trajectory), and shows the filter reliability with a smooth trajectory that corresponds to the real trajectory following by the vehicle, as can be observed in 5.25a. The GPS and DGPS standard deviations are presented in 5.25c to show the positioning errors that are solved by sensor fusion.



(a) GNSS fixes



(b) Error (1-σ) in lat/lon

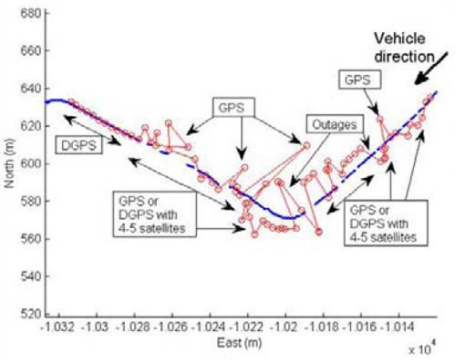


(c) Indirect quality indicators

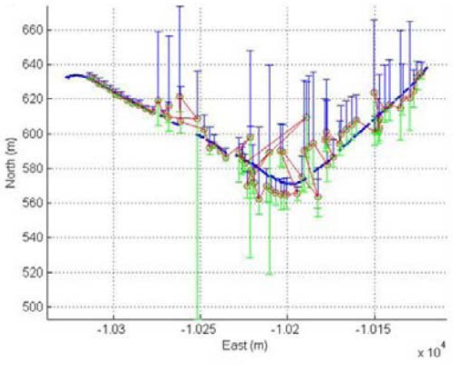
Figure 5.24: Effect of complex urban canyons in GNSS measures

A third situation is shown in Figure 5.26a. This urban trajectory presents several inaccuracies and the UKF filter solution displays again reliability to estimate the position. The difference

(a) Rover trajectory within complex urban canyon.



(b) DGPS and GPS solutions (red) and UKF filter solution (blue).

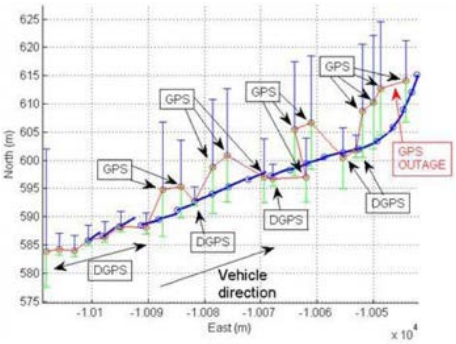


(c) Standard deviations of DGPS and GPS: East (upper blue bar) and North (lower green bar).

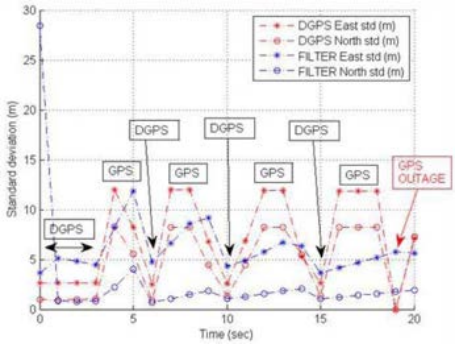
Figure 5.25: Sample trajectory: complex urban canyon.

with the former case is the use of the filter solution only for this trajectory, so the filter is started at the beginning of this trajectory. The estimated positions by the filter are shown in 5.26b. In this case, the filter solution has again better precision than the GNSS device, and time-domain standard deviations of 5.26c shows the performance of the filter.

(a) Rover trajectory within complex urban canyon.



(b) DGPS and GPS solutions (red) and UKF filter solution (blue).

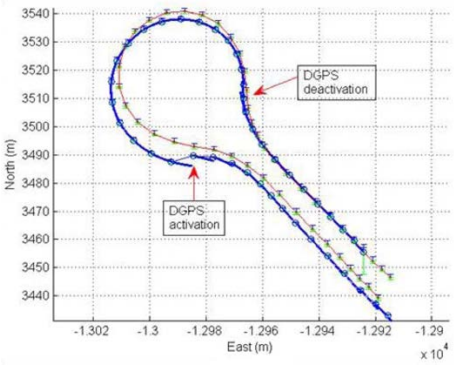


(c) Standard deviations of DGPS and GPS: East (upper blue bar) and North (lower green bar).

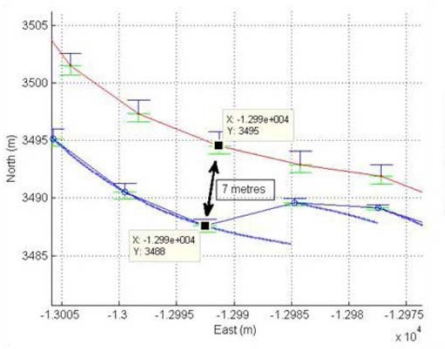
Figure 5.26: Sample trajectory: complex urban canyon.

The validation of the UKF filter solution is presented in Figure 5.27a. The validation is based on the comparison of the DGPS precise trajectory as groundtruth with calculated solution provided by the UKF filter. Figure 5.27b displays the validation experiment, where DGPS deactivation is performed at the beginning of a roundabout and the reactivation is at the end of the roundabout, so the UKF filter has a deactivated GNSS positioning for 15 seconds. In terms of correlation, both coordinates presents good results, the R2 values for East and North coordinates are 0.9959 and 0.9904. The deviation of the real trajectory at the end of roundabout is 7 meters as can be observed in 5.27c, where the East standard deviations (DGPS and filter) are indicated as upper blue bar, and North as lower green bar. The time-domain standard deviations of DGPS and filter are compared showing an increase of the UKF filter errors from second 12 to second 27 (5.27d), the effect corresponds to deactivation and reactivation of DGPS device.

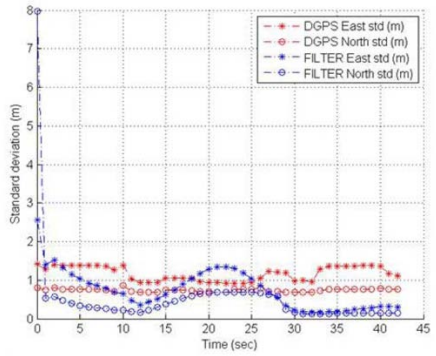
(a) Roundabout with DGPS groundtruth.



(b) DGPS trajectory (red) with deactivation and activation and UKF filter solution (blue)



(c) Detail of UKF filter solution without 15 seconds of DGPS solution



(d) Time-domain standard deviation evolutions

5.3.2 Context-aware fusion algorithms using smartphone

In this part we compare the quality and features of smartphone sensors with the on-board platform. We want to asses that they are usable as part of the navigation system, and check if the automatic context inference processes offer comparable results.

The presented results are based on a single open traffic experiment, which was recorded simultaneously by the mid-cost system and the smartphone. We drove the test vehicle in the surroundings of Leganés Campus (Universidad Carlos III de Madrid), describing the $4.5km$ long trajectory shown in figure 5.28. The experiment took for roughly 1100 seconds. According to public altimetry maps, slope is in the range $[-5\%; 6\%]$ with average value 1.7%. This trajectory features a typical urban driving scenario in Spain: buildings around five floors high, narrow streets, roundabouts, different types of pavement (including bumps), and regular speed/stop patterns.

Figure 5.28: Trajectory followed by the test car. Results presented here are based on this record.

5.3.2.1 GPS quality

Before reviewing the quality of GPS devices, let us remark that the trace recorded by Novatel GPS is quite atypical. The device was subject to blackouts between 5 and 50 seconds long in relatively open places. It also triggered several alarms revealing problems in the calculated

solution (integrity warnings and singularity in covariance matrix among others). This was probably caused by start recording data a short time after a cold start. However, we opted to keep it since it proposes very interesting situations for further experiments. As a rule of thumb, Novatel GPS fixes have a far superior quality compared with the smartphone. The principal factor is most probably the antenna: satellite signal reception is much clearer. The consequences can be observed in the number of satellites detected by each device (figure 5.29), and also the number of satellites used by Novatel device when determining the last fix. In spite that the constellation of the smartphone is reduced (never above 10 satellites), it is much more constant: between 400 and 750 seconds, it never goes down 7 satellites. On the other hand, Novatel device oscillates between 5 and 16 visible satellites.

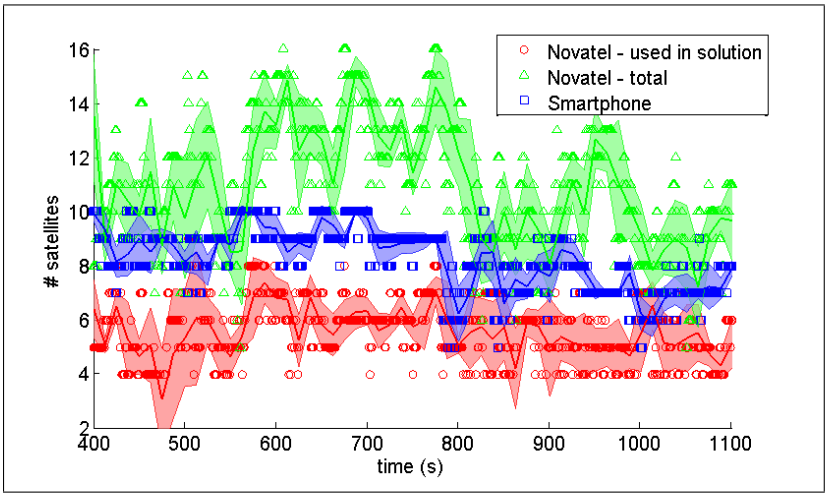


Figure 5.29: Number of satellites over time.

The superior quality of Novatel fixes is illustrated in figure 5.30. It shows a piece of the trajectory, as recorded by both sensors. It has been selected to be representative of remarkable effects found in the full sample. Observation conditions are good: open sky, no severe multipath problems. The vehicle enters the left part of the image, on the right (bottom) lane of the street, and then turns right on the junction to disappear on the bottom. Later on, the car reappears on the top-right side of the image, and drives back to the point where it first entered into scene. The green trajectory corresponds to the date recorded by Novatel GPS device. It is possible to discern on which lane the car is driving; maneuvers are clearly, accurately depicted. This is characteristic of a system with a very low relative error. The red line represents smartphone trace. Both relative and absolute errors are higher (unable to discern lanes, since the two lines do cross several times), and maneuvers show some "inertia", that is likely to be caused by internal processing on the device –on-chip assisted GPS algorithms, operating system corrections such as fusion/filtering/smoothing.

Establishing a comparison for both accuracies is a complex task, because we lack groundtruth

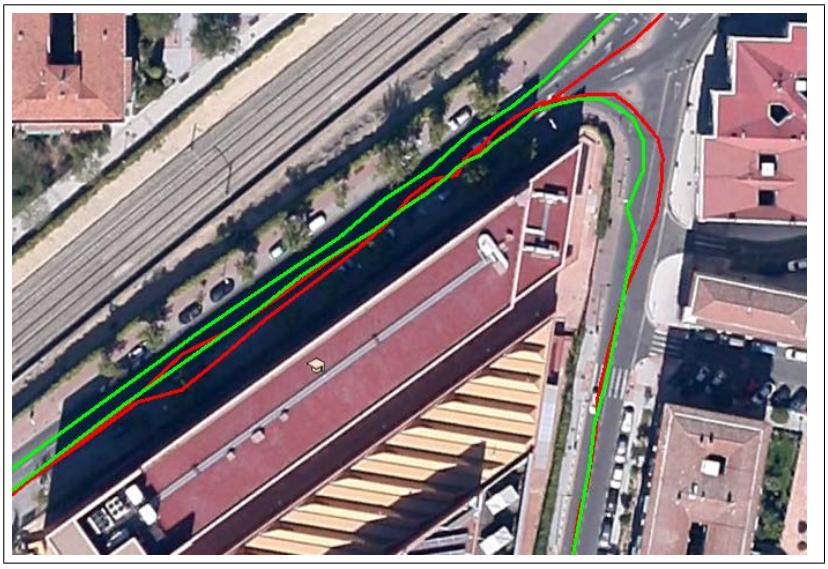


Figure 5.30: Comparison of Novatel GPS (green) and smartphone (red) raw fixes. Detail.

data. An alternative solution is to align both series of GPS fixes to a common timeline and compare their discrepancies with the accuracy information provided by the sensors. Data is aligned by applying linear interpolation between consecutive fixes. The result for part of the experiment is shown in figure 5.31 (the line for distance between fixes has been smoothed for a clear visualization).

The difference between both GPS traces follows a segmented pattern. When satellite visibility conditions are good, this difference is around a single standard deviation of the best sensor (before $t = 420s$ in the figure). The highest peaks are concentrated between $t = 420$ and $t = 550$, matching those Novatel fixes with the largest reported errors, that also happen to have a large bias. We find reasonable to conclude that this punctual errors in the mid-cost GPS system are the main cause of the error peaks in the plot.

5.3.2.2 Inertial measures quality

A comparative sample of accelerometer data along Y axis is shown in figure 5.32. It was taken with the vehicle stopped, and illustrates the accuracy of each sensor. The calculated standard deviation calculated for MicroStrain mid-cost device is four times lower than smartphone readings. Gyroscope readings around the same axis, for the same period of time, are shown in figure 5.33. Smartphone gyro shows a strong quantization effect, but its standard deviation is surprisingly lower than that of mid-cost gyro. The quantization step ($0.06 deg/s$) is consistent with the 16 bit AD converter working over a scale of $2000 deg/s$. Setting smartphone gyro on a lower scale (more suitable for vehicle navigation) would improve these results. However, Android sensor manager documentation indicates that gyro maximum range must be at least

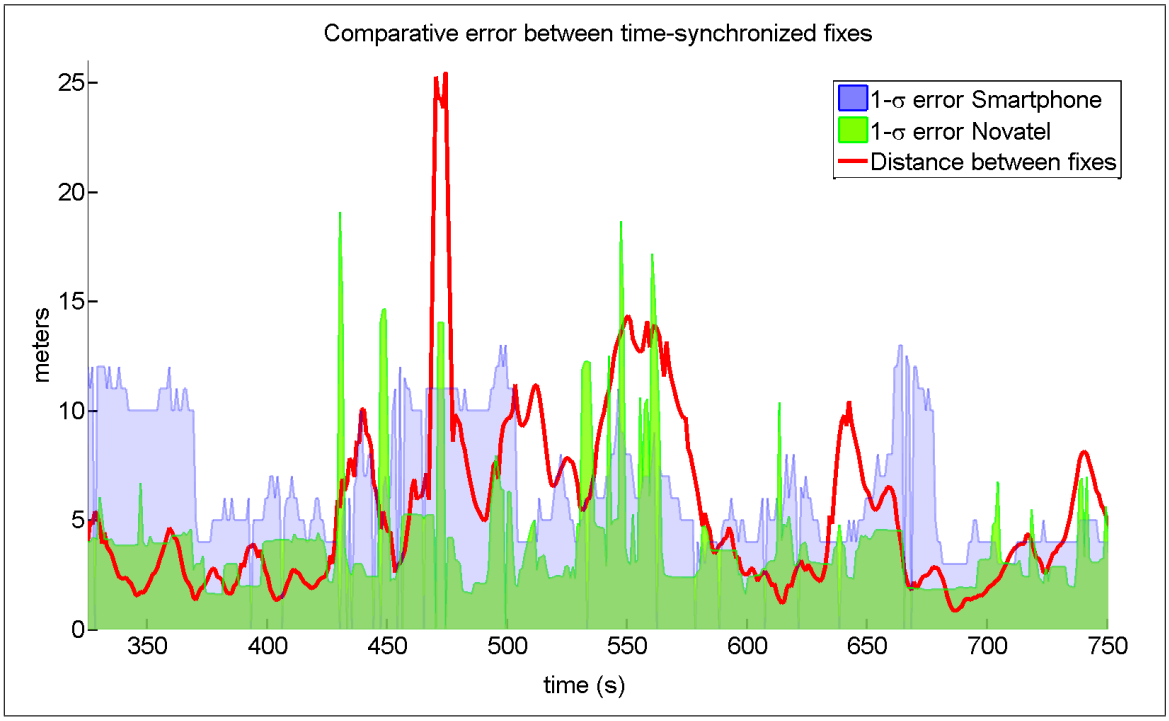


Figure 5.31: Distance between GPS fixes from Smartphone/Novatel devices, compared with the self-reported accuracy (one standard deviation).

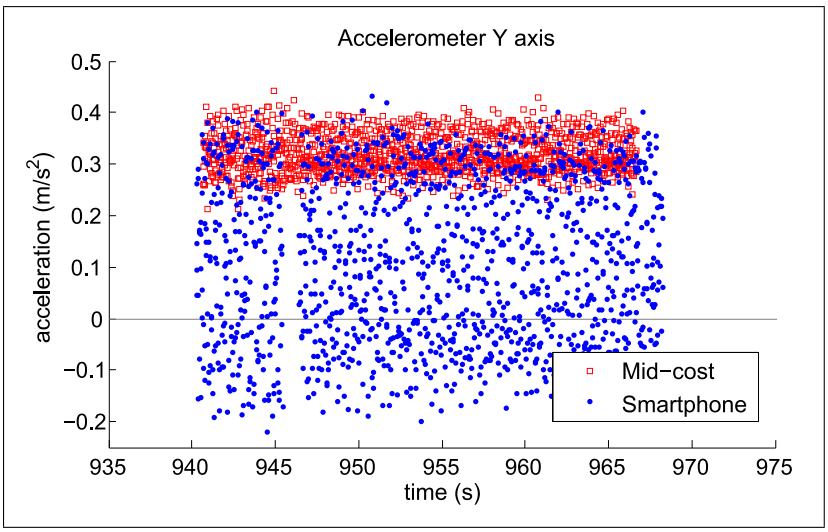


Figure 5.32: Accelerometer readings with the vehicle stopped. Note that bias is not corrected on these samples.

Table 5.8: Observed noise levels of IMU components.

Accelerometer	**Std. deviation**
On-board	$0.038m/s2$
Smartphone	$0.159m/s2$
Gyroscope	**Std. deviation**
On-board	$0.171deg/s$
Smartphone	$0.118deg/s$

$1000deg/s$, and it does not expose any method that can modify the setting. Changing the range is probably not compatible with normal smartphone functioning. Table 5.8 compare the average noise magnitude observed during stops for the different sensors.

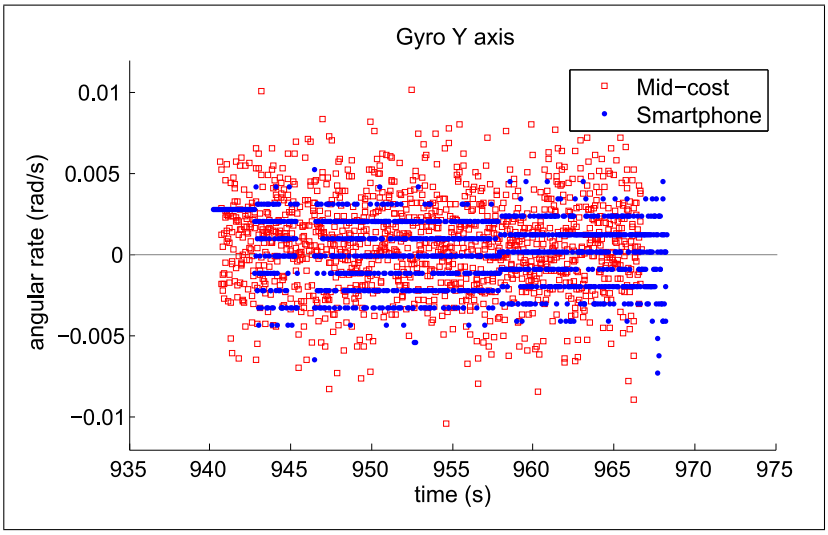


Figure 5.33: Gyroscope readings with the vehicle stopped.

Figure 5.34 tries to gather in a single plot all the strange effects found in smartphone inertial data. In first place, it is common that during maneuvers, as the one taking place between $t = 360$ and $t = 380$ seconds (a roundabout), smartphone reading is temporarily delayed about 0.3 seconds. The delay starts at $t = 370s$ and extends for 20 seconds. Later, at $t = 410$ seconds, smartphone gyro output becomes noisier for about a minute (standard deviation up to five times higher). This happens in the three axes, and returns later to normal levels. The best explanations for this behavior is that, while the mid-cost IMU is attached to the body of the car, the smartphone lies free on a surface. Driving dynamics can affect how firmly the smartphone rests on that surface, and this is translated into different vibration levels.

5.3.2.3 Automatic context extraction

Simple automated techniques can achieve comparable results for both sets of data. The best candidate is an algorithm that:

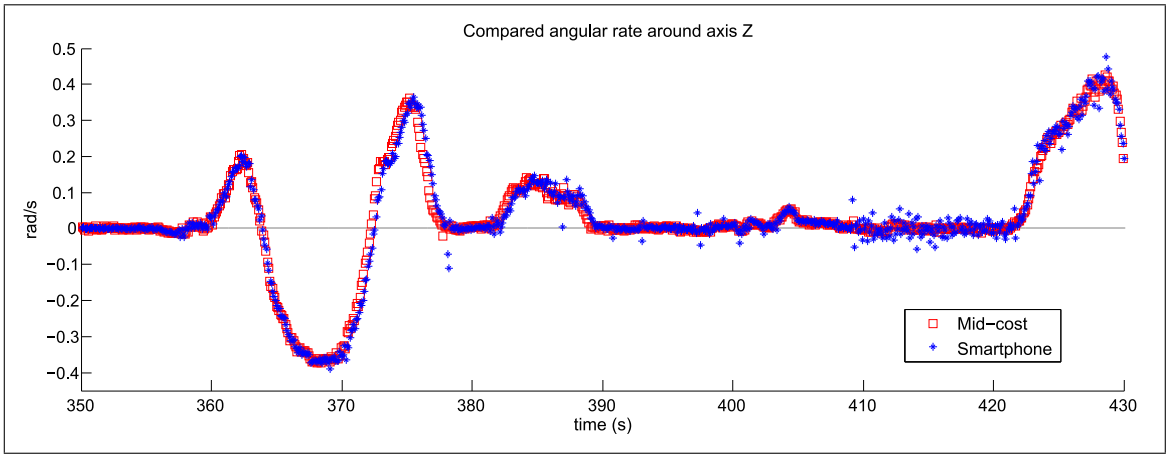


Figure 5.34: Compared angular rate on mid-cost and smartphone gyroscope (subsampled for the sake of clarity). The comparison exposes anomalies, as the smartphone signal becoming noisier around $t = 410s$, or a 0.3 seconds delay from $t = 370s$ to the next straight fragment, around $t = 390s$.

- Does not have a strong dependence on the quality of the sensor.
- Does not require prior calibration
- Is robust under sensor dynamic changes (e.g. bias drift, temperature offset)

An example is the algorithm for detecting vehicle stops. It works over accelerometer data, split in chunks. The amplitude of each chunk (difference between maximum and minimum value) is consistently low when the car is stopped, so that detection by means of thresholding is possible. This algorithm is independent of the orientation of the sensor (even in a smartphone that can change its position during the record), and works over biased data. The threshold can be estimated automatically using only inertial data using simple statistics. As a side note, chunk amplitude criterion is more consistent and clear than using the standard deviation of the signal, as used in (Han et al., 2014). Below, figures 5.35 5.36 show relevant pieces of the detected stops and straight fragments, after applying the same non-parameterized algorithms to both sets of data (mid-cost equipment and smartphone).

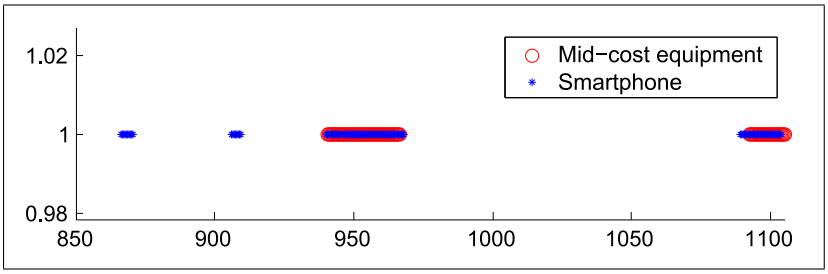


Figure 5.35: Sample of stop detection output, compared for both sensors.

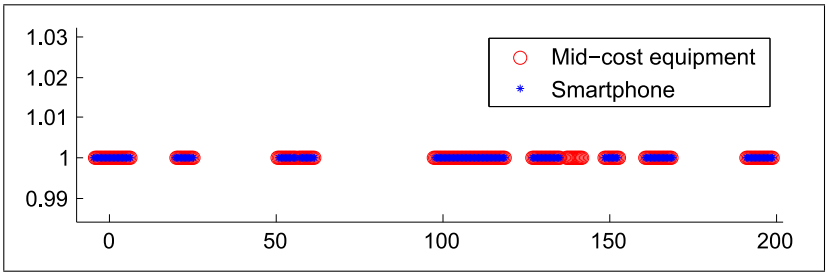


Figure 5.36: Sample of straight motion detection output, compared for both sensors.

Results are comparable. In spite that the smartphone output is a bit more sensitive in the case of stop detection, it allows to estimate the gyroscope bias with similar accuracy. The proof is that the output of straight motion detection algorithm, that requires unbiased gyro measures, is very similar for both sensors: the figure shows the only discrepancy found in the whole trajectory, around $t = 140s$.

5.3.3 Context-aware adaptability

We have tested the system for a mixed urban and open road trajectory. The Fusion Adaptation module is notified of relevant changes in the system:

- Addition and removal of sensors: smartphone is available only from $t = 100s$ to $t = 300s$.
- Changes in relevant context: the trajectory starts in urban environment, switches to open road around $t = 150s$ and goes back to urban close to the end. The battery status of the smartphone starts in "plugged", and switches to "critical" at $t = 200s$.
- Changes in the list of required fusion products. We start asking for position and linear speed, change to only position but with high accuracy during the open road fragment. At the end, we add turn detection to the list of desired products.

The adaptation module calculates a new solution right after detecting each change. We include two sample solutions reached by the system. Fig.5.37 is a solution around $t = 120s$ that returns the vehicle position and its linear speed, with smartphone sensors available and moving in a strict urban environment with poor GPS signal. The system chose to take the position from a Kalman Filter, and use the Unscented Kalman filter to extract the speed.

The second solution, shown in Fig.5.38, describes a solution in open road environment where the system is asked to produce a high-accuracy position and stop detection, with smartphone in critical battery status. In this case, the solution includes the simple kinematic Kalman Filter using the available differential GPS readings. The stop detection module, fed by the on-board sensors (smartphone is not available due to battery status), is also connected to the output.

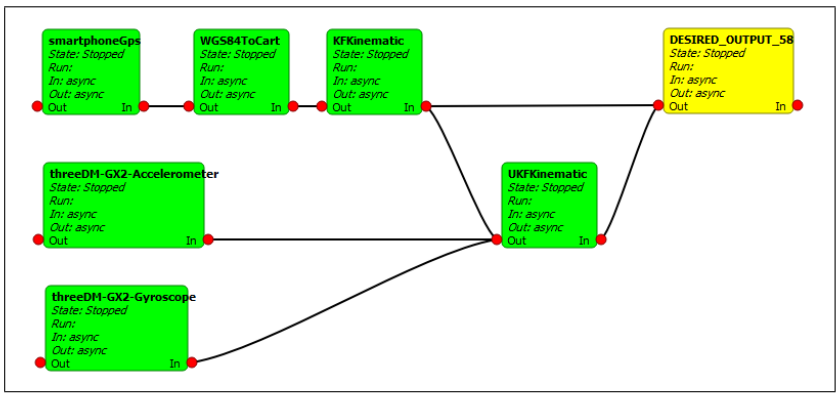


Figure 5.37: Sample fusion solution for getting vehicle position and linear speed

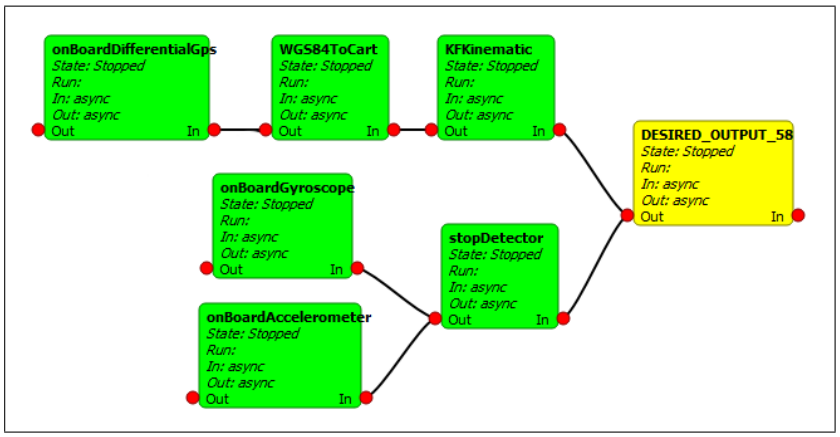


Figure 5.38: Sample fusion solution for getting high-accuracy location and stop detection

The response of the system is almost instantaneous as it detects a relevant change in context information. There are some considerations regarding this function, though: current implementation launches a full-scale search process that configures a sensor fusion solution from scratch, requiring around a second of computing in a modern CPU. Depending on the frequency of this process and the host platform, this time has to be lowered. A faster alternative could work over the last solution, modifying only those parts that are actually affected by the contextual change. Systems with severe power restrictions could rely on a set of fixed sensor fusion schemas with reduced adaptive capabilities. An example can be found in chapter 4, where the local trackers have a fixed structure with some configurable aspects.

Another aspect to take into account is the sensitivity of the adaptive strategies. Changing the processing structure or switching algorithms usually involve a short initialization time span where fusion products converge to their expected accuracy and stability. The adaptive logic should consider system stability as another factor that affects the adaptation process, in order to avoid continuous changes not leading to a significant improvement in the quality of the fusion.

5.4 Conclusions

Along this section we have shown how to design, using the framework proposed in this thesis, a sensor fusion system with self-adaptive features that makes use of context information to enhance all its processes. This example demonstrates the capabilities of the architecture designed in chapter 3, as well as the advantages of the tentative workflow as a systematic process for buildings sensor fusion applications within the scope of the framework. Regarding adaptive features, the application includes automatic selection of sensors and fusion algorithms, as well as online sensor calibration. It shows how context information can be used to decide how to adapt the system but also as an additional source of information that improves the output of certain algorithms.

We conducted an extensive set of experiments that measures the performance of the navigation system, both for each one of its parts and as a whole. Results show the superior performance of context-aware fusion algorithms, and the robustness of the system against malfunctions, degraded input information and changing conditions that would invalidate fixed-schema solutions.

6

Conclusions

This last section reviews the work done, checking if the proposed goals have been accomplished, and summarizes the contributions of this dissertation. These parts are complemented with considerations that constitute a brief discussion of the thesis.

6.1 Contributions

The contributions of this thesis are mostly theoretical, although they are supported with experiments over the implemented prototypes. We have identified the need for adaptable sensor fusion systems, and the great potential that lays under the use of context information for creating such systems. From this starting point we conducted an analysis on how context information can be used to improve fusion results, both as an additional source of information to be used by the fusion processes and as a set of factors that determine the optimal configuration of the fusion system. A key point of the analysis involved identifying the difficulties and conflictive aspects of modeling and exploiting context information. This analysis lead to a set of conclusions that have been used to define a domain-agnostic framework for creating adaptive context-aware sensor fusion applications. As far as we know after an exhaustive literature review, this is the first attempt to create a general purpose architecture that includes context information as an integral part of the sensor fusion process and proposes specific mechanisms to maximize the adaptability of the system.

The architecture has been used for the development of two prototypes in very different domains, demonstrating its applicability and showing that it is flexible enough for fitting a broad range of sensor fusion problems. The first prototype is a wide area maritime surveillance system, that tracks vessels fusing measures of radar sensors and AIS stations. This system was required to operate with different fusion configurations (algorithms and parameters) in parallel for different geographical zones. The zonal configuration should be modifiable online. We were able to provide a flexible and effective solution within the conceptual frame of the proposed

architecture, scalable in the number of zones and sensors, and showing good stability around transition zones. Additionally, the section shows how modeling domain, solution and context knowledge explicitly makes easier to create a system that will be easy to augment later with automatic adaptation capabilities.

The second prototype is a GPS/inertial navigation system for ground vehicles that can use context information to improve location accuracy in urban scenarios, where GPS positioning suffers from larger errors and can even be unavailable for short spans of time. Another contribution of this thesis is the implementation of a self-adaptive fusion system where context information is used to calibrate sensors as the vehicle moves (Martí et al., 2012). The system can also take a redundant and dynamic set of sensors, select the most appropriate subset and generate a fusion solution that combines them for getting the best solution according to the goals of the system and the contextual circumstances.

6.2 Differences between using the proposed framework and ad-hoc system design

This thesis proposes a framework as a set of conceptual tools that makes easier the design of sensor fusion solutions that are context-aware and adaptive. In order to estimate the success of this dissertation, we have to consider the advantages of using the recommended procedure and conceptual structures at design time compared with the potential hurdle of having to respect the imposed tools and structures (ontologies, widget style, virtual sensors) during the implementation phase.

Two are the advantages introduced by our proposal at design time: first, the architecture of the system is designed to avoid artificial dependencies between elements by splitting the different aspects of the problem (sensor fusion process, management of the sensor fusion system and context information) in separate modules and fostering encapsulation at the lowest possible level (sources of data and fusion algorithms) through the widget and virtual sensor abstractions. Using the proposed design, it is easier to use or integrate new data sources at any point in the architecture. This is translated into reduced risks and effort to integrate into the system potential changes in the specification of the problem, its domain or the context.

The second aid is methodical approach for designing systems that combine adaptability and use of context information within the proposed framework. This method was loosely specified in 3.3.4 as a series of tentative steps towards modeling the system using ontologies. Following a methodology, even if it does not enter in deep details, reduces the risk of skipping or making a wrong usage of important information.

One of the inconveniences of this framework is requiring explicit domain modeling. This can represent a significant effort, specially when we introduce context information in the equation.

Reaching a valid model is a complex task that involves a number considerations, as those noted in section 3.1.2.1, that can be rather vague. The required modeling effort is proportional to the complexity of the domain, but we also expect the benefits of having a formal model of the problem to be directly proportional to this complexity. The usual approach in the industry is having complex systems with a large number of implicit and hardcoded relations between its elements, described by a set of heterogeneous technical documents. Modifying those systems requires a deep knowledge of the documentation and system internals, making the process prone to errors. So, apart from the possibility of automatic processing, having the problem modeled with ontologies represents a valuable resource for developers and designer on the event of introducing changes in the sensor fusion system.

Also, forcing modularity can be difficult or even impossible to attain on certain problems. This principle is directly opposed with highly coupled solutions, that should be defined as a single large module. If the fusion system is composed of a number of algorithms and sensors entangled in a immutable, tight coupled scheme, then either the system can be considered as not adaptable or the adaptation logic has to be restricted to other aspects (parameters).

As an example of the problem of forcing modularity, the maritime surveillance scenario required us to combine Kalman and Interacting Multiple Model algorithms for filtering with several association algorithms that include JPDA. Our first design attempt defined independent modules for filtering algorithms and association strategies. However, we discovered that decoupling these two steps is a very hard task: JPDA requires specific work with the independent models and state vectors of the IMM filtering. A workaround for this problem, in our case, consisted on defining a module for each possible combination of filter and association algorithms.

6.3 Benefits of using context information in the fusion process

Context information has been proved to be helpful for enhancing fusion systems in numerous occasions, and confirming it was not among the goals of this work. Our experiments support the validity of the proposed framework as a tool for including context without negatively affecting its effectiveness.

The results presented in chapter 5 show the following benefits:

- Navigation accuracy: detecting stops and straight motion fragments allowed to apply more specific prediction models, which in turn provided limited benefits on the numerical accuracy of filtering algorithms.

- Online sensor calibration: without taking motion condition into account, it would be harder or even not possible to remove inertial sensor biases using the proposed non-invasive

procedure. This means that the inclusion of context can make possible some things that, otherwise, would simply not.

- Adaptive structure of the fusion solution: combining context information with a set of rules has been proved as an effective mechanisms for configuring fusion solutions that are appropriate regarding many different factors at the same time. Context can enrich some processes with additional semantics that open new possibilities.

- Robustness against unexpected problems: the adaptation module reacts to neutralize problems that would render useless any fusion solution with a fixed structure. It guarantees achieves full-time availability, provided that the set of working sensors is sufficient to support at least one of the proposed navigation solutions.

As long as we understand, the framework not only makes possible design and implementation of solution that integrate context information with these benefits, but it also simplifies the process.

6.4 Computational performance questions

The proposed requirements for the framework included two aspects related with the computational performance of the system. This framework, as any tool that imposes an structure and additional processes, introduces a computation overload. The first requirement stated that this overload must be negligible compared with the total amount of computation required by the fusion process or at least not suppose a big penalty. The second requirement regards scalability: if the fusion system grows in size (more sensors, more algorithms, larger context database), the computational requirements as memory and processor use must grow accordingly. Next subsections cover these questions in detail.

6.4.1 Performance penalty associated to framework usage

The use of ontologies is a controvert aspect of our proposal, especially when they have to be used in strongly resource constrained portable devices. This question has been approached before in (D'Aquin et al., 2010), that tests the time required to load and process small-medium ontologies in the smallest existing netbook at the time of writing the paper. The conclusions indicate that a small netbook can process small ontologies reasonably well (a few thousand triples using a triple-based representation instead of the XML coding employed on our experiments), although there seem to be tools better than Jena for lightweight ontology processing, as Sesame (Broekstra et al., 2002).

We can discern two different processes regarding ontologies: initial load, and query execution. The impact of the initial load is not as important as the cost of reasoning over the ontology, because for small-sized ontologies that fit in memory is a one time task during the initialization of the fusion system. However, it is important to take into account that the cited study (D'Aquin et al., 2010) reports times close to one minute for mid-size ontologies when the load process checks semantic assertions and executes additional inference processes.

Our recommendation is to avoid reasoning over the ontology except when that is not possible. Since they are used in this work as the basis for a common language, resource constrained systems should load data into an internal representation that supports the reasoning processes. This approach has been shown in the experimental part.

Regarding the rest of the processes of the system, we will use as reference the maritime surveillance application in section 4, because its performance oriented implementation using C++ language. The message-passing implementation based on the Widget abstraction has been tested under a wide set of conditions. We have observed that the computational load of processes that are not directly related with the fusion process –message passing, system configuration, management processes– represent around 1% of the total execution time in scenarios with a heavy load. We consider this result satisfactory regarding the stated performance requirements.

6.4.2 Scalability

The maritime surveillance scenario shows that the computational load associated to message sending is proportional to the number of messages and length of the traversed chain of message processors –linear complexity–. Our implementation, based on the Boost Signals library [1], was able to send several million messages per second on a modern machine. This shows that the Widget abstraction is a valid mechanism for low coupled architectures where the ratio number of messages / amount of computation for processing them is low. Application that are highly coupled or involve passing a big number of small messages that require barely no processing should consider an alternative communication mechanism as shared memory pools, that can still be compatible with the widget abstraction.

With respect to the number of sensors, scalability depends on the implemented management/adaptation processes. In the maritime scenario, complexity is nearly-linear for the addition of new sensors because the possibility of action is quite limited. In the ground navigation application, however, adding new sensors and algorithms has a big impact on the automatic selection of fusion solutions due to using a basic search strategy. The response time is kept around a few seconds in a regular computer for algorithm repositories containing a hundred

[1] Documentation available at `http://www.boost.org/doc/libs/1_58_0/doc/html/signals.html`

nodes. This should be improved for real time embedded systems, although there are a number of strategies available for that purpose.

6.5 Future work

It would be interesting to design further experiments that can characterize how the quality and properties of the fusion products change depending on the quality and availability of context knowledge, e.g. when some variables are not available or their value is uncertain. This work, actually, does not mention the problem of uncertainty in context information, but it is interesting to approach it starting with a theoretical analysis and following with some experiments.

The presented proposal is restricted to centralized fusion systems. Its encapsulation and abstraction mechanisms (virtual sensors, widgets) facilitate the development of limited distributed system capabilities, but the architecture itself does not take into account the specific aspects and problems related with distributed systems such as communication bottlenecks, delays, and others.

An interesting future work consists on extending the solution to include mechanisms for enabling distributed schemas. One possibility suggested in (Martí et al., 2011a) is to interpret independent fusion systems as software agents that can create dynamic coalitions to exchange information and services. The advantage of this solution is that it can be implemented by adding an external layer of software over a fusion system built using this thesis proposal. More coupled solutions involve, most probably, substantial changes at many of the defined levels and would also place restrictions in the –at this moment unconstrained– core of the fusion solution, the algorithms.

A

Filtering algorithms

This appendix covers the description of some of the filtering algorithms that have been used during the development of this thesis. It starts with the most basic of the employed recursive filters, the Kalman filter. After it we introduce the Unscented Kalman Filter, a non-linear extension that produces more accurate estimations that competing unimodal techniques, and the Particle Filter as a representative of sample based algorithms. Figure A.1 shows these and more filtering algorithms according to some of their features.

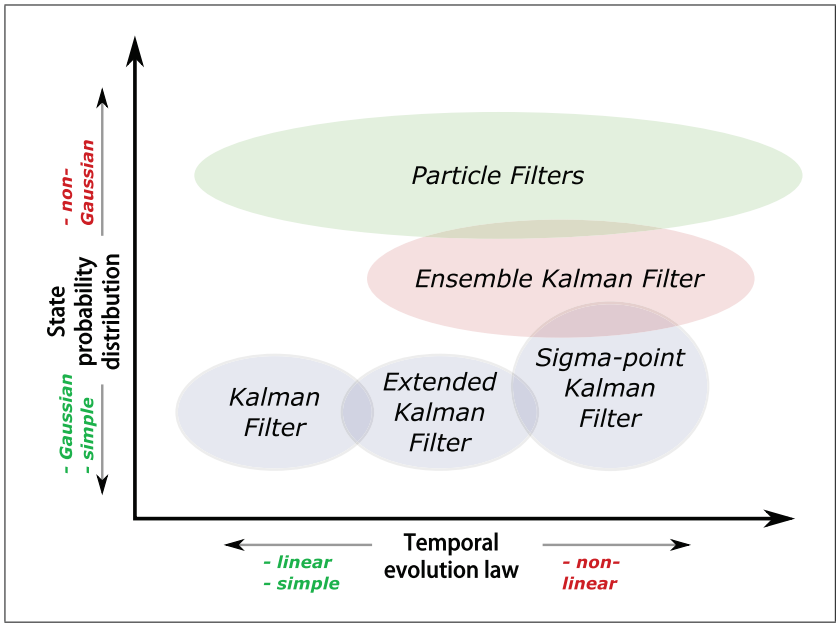


Figure A.1: Comparative chart of several filtering algorithms, considering two variables: how complex/powerful is the probability distribution used to described the estimated state of the system, and the expressiveness of the prediction model supported by the filter.

A.1 Kalman Filter

Takes its name from Rudolph Kalman, considered to be its creator and one of the first developers of its theoretical basis. A Kalman Filter calculates the optimal state estimation for systems whose temporal evolution is described by a linear process subject to a zero mean, gaussian distributed noise, and where the state is partially observable through measures that can be described as a linear transformation of the true state and are also affected by a random samples distributed as a multivariate gaussian noise.

Prediction Prediction uses the state estimation at current time $[x_t^+; P_t^+]$, and calculates the prior estimation of the state at next time step $[x_{t+1}^-; P_{t+1}^-]$. "Prior" means that this estimation does not incorporate and observation at that instant. Prediction requires applying the following equations:

$$x_{t+1}^- = F_{\Delta t} \cdot x_t^+ P_{t+1}^- = F_{\Delta t} \cdot P_t^+ \cdot F_{\Delta t}^T + Q, \quad w_t \sim \mathcal{N}(0, Q) \tag{A.1}$$

Matrix $F_{\Delta t}$ represents the linear prediction process. It is used to explore how the state vector x_t^+ changes, and also to know how state uncertainty P_t^- grows over time when no new observations arrive. Term w_t is the Gaussian-like plant/process noise, described by covariance matrix Q.

Update Kalman Filter update process incorporates the information contained in an observation y_t into state estimation in order to correct the prior estimation and to reduce its uncertainty:

$$x_t^+ = x_t^- + K_t \cdot e_t P_t^+ = (I - K_t \cdot H) \cdot P_t^- \tag{A.2}$$

Measure function has to be a linear transformation of the real state, as required for the prediction model. Thus, it can be represented by a matrix H that multiplies state vector (or state covariance matrix) to project its information into the observation space. The term e_t is known as "innovation". It is the difference between the actual observation y_t and the observation we could expect from prior state estimation:

$$e_t = (y_t - H \cdot x_t^-) \tag{A.3}$$

K_t is known as Kalman gain matrix, and is calculated as:

$$K_t = P_t^- H^T \cdot (H P_t^- H^T + R)^{-1}, \quad v_t \sim \mathcal{N}(0, R) \tag{A.4}$$

Where v_t is the observation noise as described by covariance matrix R.

Constrained to linear problems, the Kalman Filter cannot be applied to a large number of relevant problems. Many alternatives have been proposed to overcome this obstacle. In the

next sections we will explore two of them in detail: the Unscented Kalman Filter, also known as "Sigma-Point" Kalman Filter, and the Particle Filter.

A.2 Unscented Kalman Filter

The UKF is a member of the Kalman family. As the basic Kalman filter (Welch and Bishop, 1995), it is a recursive algorithm that estimates the state $\hat{x}_k$ of discrete-time dynamic system composed by a mix of partially observable and hidden variables. The estimation is described as multivariate Gaussian distribution with mean x_k and covariance P_k. These filters use a mathematical description of how the system evolves over time, the prediction model $f(\cdot)$:

$$\hat{x}_{k+1} = f(\hat{x}_k, u_k, v_k) \tag{A.5}$$

Where u_k is an input that complements the prediction model but does not provide any information about the state by itself, and $v_k \sim \mathcal{N}(0; R_v)$ represents a process noise distributed as a Gaussian with mean zero and covariance matrix R_v.

A series of measurements are received over time:

$$\hat{y}_k = h(\hat{x}_k, w_k) \tag{A.6}$$

Which are observations of the true state transformed by a known measurement model $h(\cdot)$ and perturbed by a random sample of the observation noise $w_k \sim \mathcal{N}(0; R_w)$ with the same restrictions applied to process noise. The information provided by such observations is integrated into the state estimation during the update step.

The UKF (Uhlmann and Julier, 1997; Van der Merwe et al., 2000) is an extension of the original algorithm that allows using nonlinear models. Given a L-dimensional state, this filter uses a set of $2L + 1$ weighted sample points χ –called sigma points– that are deterministically chosen according to the mean x_k and covariance P_k of the state estimation:

$$\chi_0 = x_k \tag{A.7}$$

$$\chi_i = x_k + \left(\sqrt{(L + \lambda) \cdot P_k}\right)_i, i = 1, \dots, L \tag{A.8}$$

$$\chi_i = x_k - \left(\sqrt{(L + \lambda) \cdot P_k}\right)_i, i = L + 1, \dots, 2L \tag{A.9}$$

Where $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter, and constants α, κ are the spreading of the sigma points around the mean and a secondary scaling parameter respectively. $\sqrt{((L + \lambda) \cdot P_k)}$ is a matrix square root, and $(\cdot)_i$ represents its i-th column.

These points are propagated using the prediction function $\chi_(k+1)^- = f(\chi_k, u_k)$. The new state probability distribution $x_(k+1)^-, P_(k+1)^-$ are calculated as the weighted mean and covariance of the sigma points:

$$x_{k+1}^- = \sum_{i=0..2L} W_i^m \cdot \chi_i^- \tag{A.10}$$

$$P_{k+1}^- = \sum_{i=0..2L} W_i^c \cdot (\chi_i^- - x_{k+1}^-)(\chi_i^- - x_{k+1}^-)^T \tag{A.11}$$

Where the weights for the mean $W_i^((m))$ and covariance $W_i^((c))$ are given by:

$$W_0^{(m)} = \lambda/(L+\lambda) \tag{A.12}$$

$$W_0^{(c)} = \lambda/(L+\lambda) + (1-\alpha^2+\beta) \tag{A.13}$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2\cdot(L+\lambda)}, i = 1, \dots, 2L \tag{A.14}$$

Being β a parameter that controls the shape of the distribution ($\beta = 2$ optimal for Gaussian distributions).

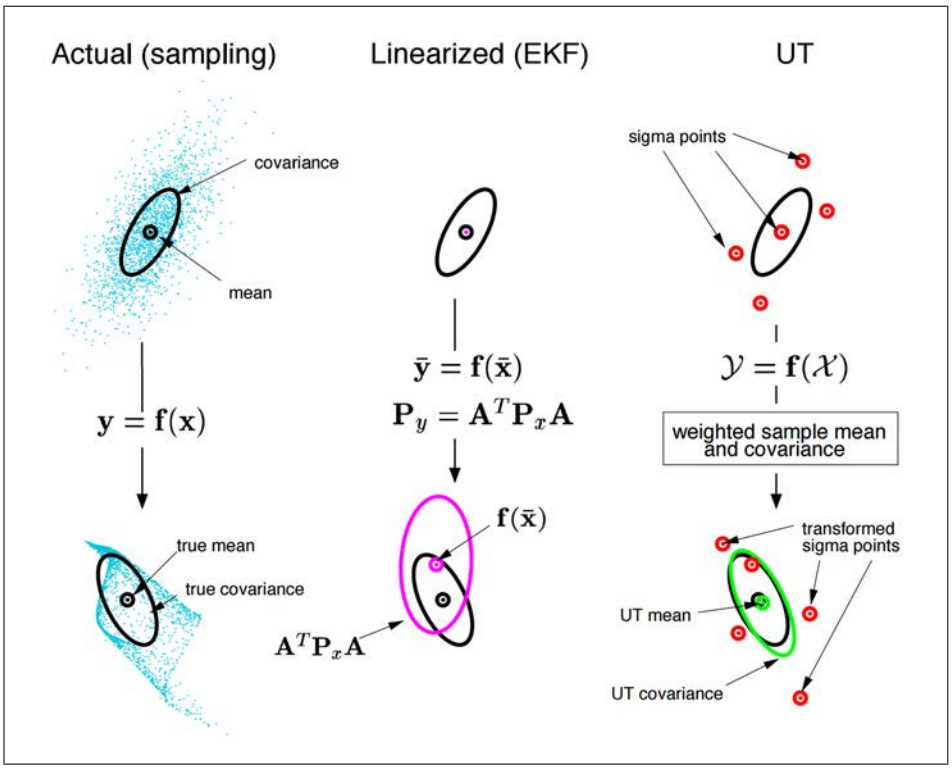


Figure A.2: Advantages of the Unscented Transform over other linearization methods. Taken from (Wan and Van Der Merwe, 2000).

The Unscented Kalman Filter models with great accuracy the result of a Gaussian distributed state that is transformed by a non-linear process –some studies have shown that they keep

3rd-order statistics of the real transformation–, using a fraction of the computational power required by traditional sampling methods. Figure A.2 provides a visual comparison of the Unscented transform process with the simpler (but extensively used) Extended Kalman Filter and a sample based technique like the Particle Filter. For extended details on the basics of Kalman-like filters in general and UKF in particular, the excellent (Van der Merwe et al., 2000; Welch and Bishop, 1995) are recommended.

A.3 Particle Filter

Particle Filters, as formulated in the original work (Gordon et al., 1993), represents the most general and powerful approach to the filtering problem. It is a good choice when the state of the system follows a highly complex distribution (multimodality and discontinuities), its temporal evolution or the observation model are highly non linear and/or stochastic, and the uncertainties are also complex.

Particle filters are sequential recursive estimators based on Monte Carlo theory: they approximate the real state probability distribution using a large number of random samples They start with an estimation of the system state at time t, represented by a set of n particles:

$$p(x_t)^n \sim x_t \cong x_t^1, x_t^2, x_t^3, \dots, x_t^{n-1}, x_t^n \tag{A.15}$$

Where x_t^i is the $i-$th particle used to approximate the probability distribution of the system state. The prediction model $f(\cdot)$ is applied to each individual particle, $x_{t+1}^i = f(x_t^i)$ Prior state estimation at next time step is represented by the new population of particles $\tilde{x}_{t+1}^-$, whose distribution should be similar to the density function of the real state $p(x_{t+1})$.

Each particle is assigned a weight proportional to the probability that particle has of describing the real state. Actual state density is a combination of this weight with the distribution of the particles in the state space (actual spatial density). The best estimation of the state is calculated as the weighted average of the particles.

Computational cost is the principal drawback of Particle Filters. They can be used to solve complex problems, but may require a huge number of particles to get accurate results. Compared with the UKF, that needed $2L + 1$ samples (where L is the dimensionality of state space), a Particle Filter can require thousand of particles for problems with 4-6 dimensions –navigation in 2D or 3D.

A.3.1 Formal description

This section explains how particle filters work, starting from the pure Bayesian theory of recursive estimation –for which the Kalman-type filters are particular approximations–. From that point,

we will discuss how Monte Carlo methods can be used to solve the estimation problem. Last, we will explain the sampling and resampling techniques, that are used to reduce the number of particles required to keep a good estimation of state probability distribution.

This section makes use of the following symbols (apart from those previously used):

- $Y_k = y_1, y_2, \ldots, y_{k-1}, y_k$ is the set of measures/observations up to instant k.
- $X_k = x_1, x_2, \ldots, x_{k-1}, x_k$ is the set of states up to instant k

A.3.1.1 Recursive Bayes Estimation

The filtering problem is defined as knowing the state of a system at current time x_k, given all the available observations from the starting instant Y_k. Since these observations are not perfect (and maybe not even complete), the problem is redefined as calculating the probability distribution of that state conditioned to the observed measures, $p(x_k|Y_k)$. According to Bayes theorem, this probability distribution can be expressed as:

$$p(x_k|Y_k) = \frac{p(Y_k|x_k) \cdot p(x_k)}{p(Y_k)} \tag{A.16}$$

We can extract last measure y_k from the whole set, $Y_k = y_k, Y_{k-1}$, and reorganize terms as following:

$$\begin{aligned} p(x_k|Y_k) = \frac{p(Y_k|x_k) \cdot p(x_k)}{p(Y_k)} &= \frac{p(y_k, Y_{k-1}|x_k) \cdot p(x_k)}{p(y_k, Y_{k-1}} \\ &= \frac{p(y_k|x_k, Y_{k-1}) \cdot p(Y_{k-1}|x_k) \cdot p(x_k)}{p(y_k|Y_{k-1}) \cdot p(Y_{k-1})} \end{aligned} \tag{A.17}$$

Applying Bayes theorem to $p(Y_{k-1}|x_k)$, we have:

$$\begin{aligned} p(x_k|Y_k) = \ldots = \frac{p(y_k|x_k, Y_{k-1}) \cdot p(Y_{k-1}|x_k) \cdot p(x_k)}{(p(y_k|Y_{k-1}) \cdot p(Y_{k-1})} \\ = \frac{p(y_k|x_k, Y_{k-1}) \cdot p(x_k|Y_{k-1}) \cdot p(Y_{k-1}) \cdot p(x_k)}{p(y_k|Y_{k-1}) \cdot p(Y_{k-1}) \cdot p(x_k)} \end{aligned} \tag{A.18}$$

We can cancel common terms in numerator and denominator. Furthermore, observation depend only in the state, so we can remove Y_{k-1} from the condition of the first term $(p(y_k|x_k, Y_{k-1}) = p(y_k|x_k)$. This gives the final result:

$$p(x_k|Y_k) = \frac{p(y_k|x_k) \cdot p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})} \tag{A.19}$$

Where:

- $p(y_k|x_k)$ is the likelihood of an observation given current state. It is defined by the observation model.

- $p(x_k|Y_{k-1})$ is state prior probability subject to the set of measures until the previous time step. Can be expressed as $\int p(x_k|x_{k-1}) \cdot p(x_{k-1}|Y_{k-1})\,\mathrm{d}x_{k-1}$, being $p(x_k|x_{k-1})$ the state probability distribution at current time step, conditioned to the state at previous step. This last probability distribution is defined by the prediction function of the system. Finally, $p(x_{k-1}|Y_{k-1})$ is the posterior probability of the state of the system at the previous time step.

- $p(y_k|Y_{k-1})$ is the evidence. Can be calculated as $\int (y_k|x_k) \cdot p(x_k|Y_{k-1})\,\mathrm{d}x_k$.

The optimal solution to this problem involves solving some integrals. This is the point where Monte Carlo theory can be used to make the problem tractable.

A.3.1.2 Perfect Monte Carlo method

We want to estimate a probabilistic state x_k. With that purpose, we can evaluate the density function $p(x_k|Y_k)$ in concrete places of x_k, but it is not possible to get an analytic expression where we could apply continuous calculus tools:

$$E[x_k] = \int (x_k) \cdot p(x_k|Y_k)\,\mathrm{d}x_k \tag{A.20}$$

Monte Carlo theory transforms integrals into finite sums of random samples. We could approximate the integral above by extracting random samples from $^{i}_{k}$, so that they are mutually independent and identically distributed to the probability distribution $p(x_k|Y_k)$, and calculate their average value:

$$E[x_k] = \int x_k \cdot p(x_k|Y_k)\,\mathrm{d}x_k \approx \frac{1}{N}\sum_{i=1}^{N} f(x_k^i), \quad x_k^i \leftarrow i.i.d \rightarrow p(x_k|Y_k) \tag{A.21}$$

This value minimizes the mean square error to the random samples drawn and, thus, minimizes the distance to the real state. This is illustrated in figure A.3.

A.3.1.3 Importance Sampling

Most times, however, generating samples according to posterior probability is either very costly or not possible. "Importance sampling" is a suitable alternative: samples are extracted from a different probability distribution $q(x_k|Y_k)$ (receives the name of "proposal distribution") that is easier to sample. Then, instead of taking the simple average of the samples, it calculates a

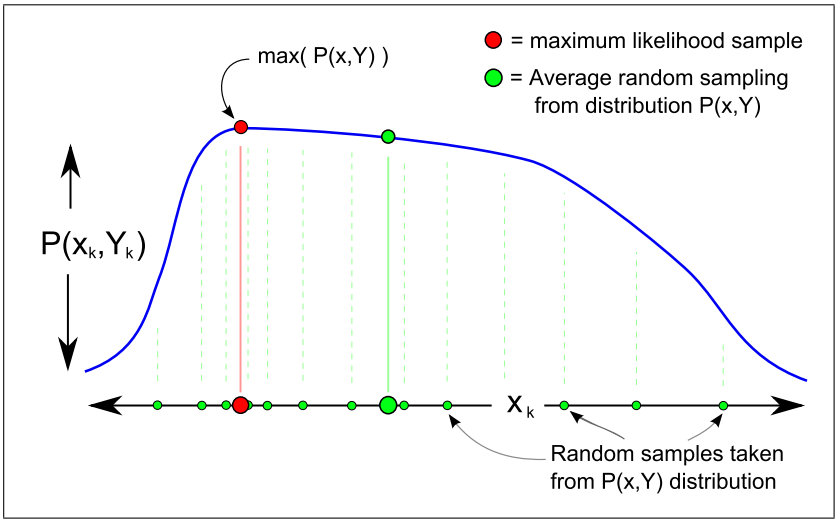


Figure A.3: State estimation calculated as the mean value of random samples, drawn according to the posterior probability distribution. Other solution, as the sample with a higher density (trying to find the peak of the real density function) offers poor results with asymmetric probability distribution.

weighted mean according to the importance of each sample. Next we will develop this idea, starting from the formulation above exposed for the estimation $E[x_k]$:

$$E[x_k] = \int f(x_k) \cdot p(x_k|Y_k)\, \mathrm{d}x_k = \int \frac{f(x_k) \cdot p(x_k|Y_k) \cdot q(x_k|Y_k)}{q(x_k|Y_k)}\, \mathrm{d}x_k \tag{A.22}$$

Applying Bayes to $p(x_k|Y_k)$ yields:

$$E[x_k] = \int \frac{f(x_k) \cdot p(x_k|Y_k) \cdot q(x_k|Y_k)}{q(x_k|Y_k)}\, \mathrm{d}x_k = \int \frac{f(x_k) \cdot p(Y_k|x_k) \cdot p(x_k) \cdot q(x_k|Y_k)}{p(Y_k) \cdot q(x_k|Y_k)}\, \mathrm{d}x_k \tag{A.23}$$

At this point, we are going to define weight (or importance) of a sample x_k as:

$$w_k(x_k) = \frac{p(Y_k|x_k) \cdot p(x_k)}{q(x_k|Y_k)} \tag{A.24}$$

Which allows to simplify the expression for the estimate into:

$$E[x_k] = \int \frac{f(x_k) \cdot w_k \cdot x_k \cdot q(x_k|Y_k)}{p(Y_k)}\, \mathrm{d}x_k \tag{A.25}$$

Taking out parts that do not depend on the integration variable:

$$E[x_k] = \frac{1}{p(Y_k)} \int f(x_k) \cdot w_k \cdot x_k \cdot q(x_k|Y_k)\, \mathrm{d}x_k \tag{A.26}$$

Now we can apply the substitution $p(Y_k) = \int p(Y_k|x_k) \cdot p(x_k)\ \mathrm{d}x_k$, adding the term $q(x_k|Y_k)/q(x_k|Y_k)$ for the sake of simplicity. Result is:

$$p(Y_k) = \int \frac{p(Y_k|x_k) \cdot p(x_k) \cdot q(x_k|Y_k)}{q(x_k|Y_k)}\ \mathrm{d}x_k = \int w_k(x_k) \cdot q(x_k|Y_k)\ \mathrm{d}x_k \tag{A.27}$$

Which, applied to state estimation x_k results in:

$$E[x_k] = \frac{\int f(x_k) \cdot w_k \cdot x_k \cdot q(x_k|Y_k)\ \mathrm{d}x_k}{\int w_k \cdot x_k \cdot q(x_k|Y_k)\ \mathrm{d}x_k} \tag{A.28}$$

Monte Carlo theory transforms integrals into finite sums, as defined in the previous section:

$$E[x_k] = \frac{\int f(x_k) \cdot w_k \cdot x_k \cdot q(x_k|Y_k)\ \mathrm{d}x_k}{\int w_k \cdot x_k \cdot q(x_k|Y_k)\ \mathrm{d}x_k} \approx \frac{\frac{1}{N}\sum_{i=1}^{N} \int f(x_k^i) \cdot w_k \cdot x_k^i}{\frac{1}{N}\sum_{i=1}^{N} w_k \cdot x_k^i} \tag{A.29}$$

Although in this case the samples x_k^i must be distributed according to the proposal $q(x_k|Y_k)$. Using properties of sums, the expression can be transformed into:

$$\begin{aligned} E[x_k] &\approx \frac{\frac{1}{N}\sum_{i=1}^{N} f(x_k^i) \cdot w_k \cdot x_k^i}{\frac{1}{N}\sum_{i=1}^{N} w_k \cdot x_k^i} \\ &= \frac{\sum_{i=1}^{N} f(x_k^i) \cdot w_k \cdot x_k^i}{\sum_{i=1}^{N} w_k \cdot x_k^i} \\ &= \sum_{i=1}^{N} f(x_k^i) \cdot \tilde{w_k} \cdot x_k^i \end{aligned} \tag{A.30}$$

Being $\tilde{w_k} \cdot x_k^i$ the normalized weights $\frac{w_k \cdot x_k^i}{\sum_{i=1}^{N} w_k \cdot x_k^i}$. Thanks to working with Markovian processes, we can transform weight calculation into a recursive formulation that does not need the full series of measures but only the last one.

$$w_k = w_{k-1} \frac{p(y_k|x_k) \cdot p(x_k|x_{k-1})}{q(x_k|Y_k)} \tag{A.31}$$

As summary:

$$E[x_k] = \sum_{i=1}^{N} f(x_k^i) \cdot \tilde{w_k} \cdot x_k^i \tag{A.32}$$

$$\tilde{w_k}(x_k^i) = \frac{w_k \cdot x_k^i}{\sum_{i=1}^{N} w_k \cdot x_k^i} \tag{A.33}$$

$$w_k = w_{k-1} \frac{p(y_k|x_k) \cdot p(x_k|x_{k-1})}{q(x_k|Y_k)} \tag{A.34}$$

The last step is to select a suitable proposal distribution $q(x_k|Y_k)$. The prior distribution

of prediction function $p(x_k|x_{k-1})$ is a very popular choice, because it simplifies weight update expression into:

$$w_k = w_{k-1}\frac{p(y_k|x_k)\cdot p(x_k|x_{k-1})}{q(x_k|Y_k)} = w_{k-1}\cdot p(y_k|x_k) \tag{A.35}$$

However, its accuracy is low compared with other alternatives, because it does not incorporate information from the last observation y_k. The application of this distribution can increase the variance of the population far above the optimal value, taking a large number of them into zones with a very low density. As a result, the interesting parts of the probability function are described by just a few particles, with a negative impact on the quality of results.

A.3.1.4 Resampling

The goal of a particle filter is to express the state probability distribution with maximum accuracy, while reducing the number of particles to the minimum possible for keeping computational requirements low. On the other hand, the variance of particle population grows unbounded with time, leading to the problem of low-weighted particles we mentioned before. To avoid this problem, the original design of Particle Filters include a step called *population resampling*. We are going to explain this process using figure A.4, which is an adaptation of the classical work taken from(Van der Merwe et al., 2000).

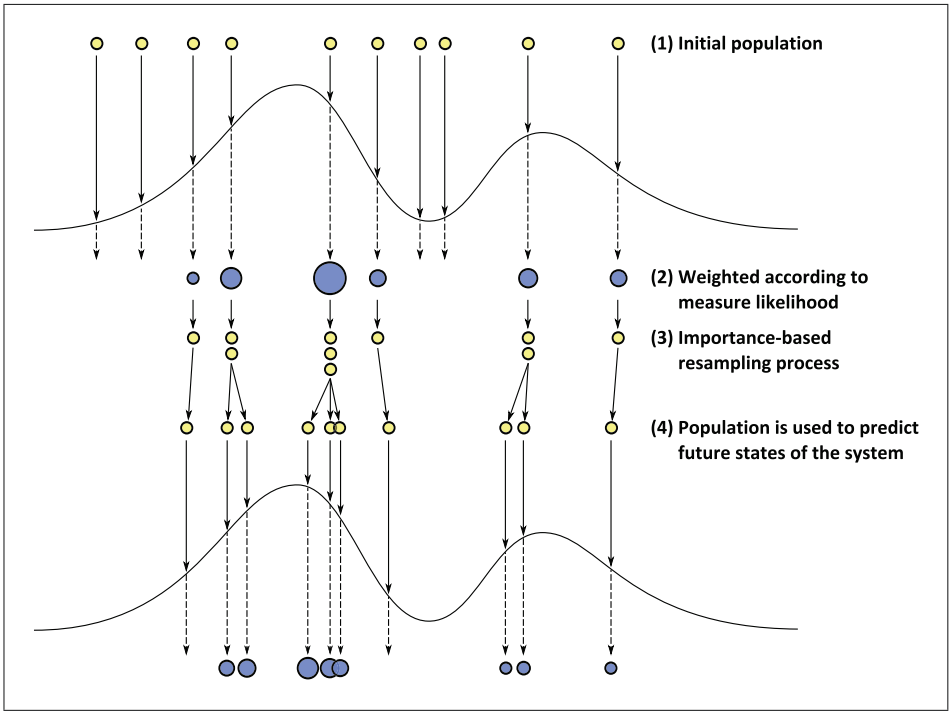


Figure A.4: Illustration of the sampling importance resampling process, inside the working cycle of a particle filter. Adapted from (Van der Merwe et al., 2000).

The population is resampled when weights variance grows over an acceptable limit. This limit is often determined using the concept of "effective sample size" N_{eff}, applied when:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2} > 0.5 \tag{A.36}$$

There are numerous numerical methods that decide how population is resampled. Among the most important, we can find *systematic resampling, stratified resampling* and *residual resampling*. For more details, check(Douc and Cappe, 2005).

During the development of this thesis, we deepened in the study of resampling processes for small-sized populations, applied to problems where process noise is small compared with the density of particles. In those scenarios, particle filters suffer a problem known as *sample depletion*, where particles no longer describe the state of the system with a sufficient accuracy. For solving this problem, we proposed a technique called "Neighborhood-based Regularization resampling" (Martí et al., 2011b,d). It improves the distribution of particles by filling empty spaces between neighbor samples, reducing the required number of particles and avoiding sample depletion in some scenarios.

References

Angerman, W. S. (2004). *Coming Full Circle with Boyd's Ooda Loop Ideas: An Analysis of Innovation Diffusion and Evolution*. PhD thesis, Air Force Institute of Technology at Wright-Patterson Air Force Base, Ohio.

Antos, S. A., Albert, M. V., and Kording, K. P. (2014). Hand, belt, pocket or bag: Practical activity tracking with mobile phones. *Journal of neuroscience methods*, 231:22–30.

Azimirad, E. and Haddadnia, J. (2015). he Comprehensive Review on JDL Model in Data Fusion Networks: Techniques and Methods. *International Journal of Computer Science and Information Security*, 13(1):53–60.

Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263.

Barbera, H., Skarmeta, A., Izquierdo, M., and Blaya, J. (2000). Neural networks for sonar and infrared sensors fusion.

Bazire, M. and Brézillon, P. (2005). Understanding context before using it. In Dey, A., Kokinov, B., Leake, D., and Turner, R., editors, *CONTEXT'05 Proceedings of the 5th international conference on Modeling and Using Context*, volume 3554 of *Lecture Notes in Computer Science*, pages 29–40, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bedworth, M. and O'Brien, J. (2000). The Omnibus model: a new model of data fusion? *IEEE Aerospace and Electronic Systems Magazine*, 15(4):30–36.

Benaskeur, A., Mcguire, P., Brennan, R., Liggins, G., and Wojcik, P. (2007). A Distributed Intelligent Tactical Sensor Management System. *International Journal of Intelligent Control and Systems*, 12(2):97–106.

Blasch, E., Garcia Herrero, J., Snidaro, L., Llinas, J., Seetharaman, G., and Palaniappan, K. (2013). Overview of contextual tracking approaches in information fusion. In Pellechia, M. F., Sorensen, R. J., and Palaniappan, K., editors, *SPIE Defense, Security, and Sensing*, page 87470B. International Society for Optics and Photonics.

Blasch, E. P. and Plano, S. (2002). JDL level 5 fusion model: user refinement issues and applications in group tracking. In Kadar, I., editor, *Proceedings of SPIE*, volume 4729, pages 270–279. International Society for Optics and Photonics.

Blázquez Gil, G., Berlanga, A., and Molina, J. M. (2012a). InContexto: Multisensor Architecture to Obtain People Context from Smartphones. *International Journal of Distributed Sensor Networks*, 2012(2012):1–15.

Blázquez Gil, G., Luis Bustamante, A., Berlanga, A., and Molina, J. M. (2012b). ContextCare: Autonomous Video Surveillance System Using Multi-camera and Smartphones. In Casillas, J., Martínez-López, F. J., and Corchado Rodríguez, J. M., editors, *Management Intelligent Systems. Advances in Intelligent Systems and Computing*, volume 171 of *Advances in Intelligent Systems and Computing*, pages 47–56, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boström, H., Andler, S. F., Brohede, M., Johansson, R., Karlsson, E., Van Laere, J., Niklasson, L., Nilsson, M., Persson, A., and Ziemke, T. (2007). On the definition of information fusion as a field of research. Technical report, School of Humanities and Informatics, University of Skövde, Skövde, Sweden.

Brickley, D. and Guha, R. (2004). RDF Vocabulary Description Language 1.0: RDF Schema.

Broekstra, J., Kampman, A., and Harmelen, F. V. (2002). Sesame: An Architecture for Storing and Querying RDF Data and Schema Information. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 54–68. Springer-Verlag London, UK.

Bürger, T. and Simperl, E. (2008). Measuring the Benefits of Ontologies. In Meersman, R., Tari, Z., and Herrero, P., editors, *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, volume 5333 of *Lecture Notes in Computer Science*, pages 584–594. Springer Berlin Heidelberg, Berlin, Heidelberg.

Caron, F., Duflos, E., Pomorski, D., and Vanheeghe, P. (2006). GPS/IMU data fusion using multisensor Kalman filtering: introduction of contextual aspects. *Information Fusion*, 7(2):221–230.

Carpenter, G. A., Martens, S., and Ogas, O. J. (2005). Self-organizing information fusion and hierarchical knowledge discovery: a new framework using ARTMAP neural networks. *Neural Networks*, 18(3):287–295.

Chae, H., Christiand, Choi, S., Yu, W., and Cho, J. (2010). Autonomous navigation of mobile robot based on DGPS/INS sensor fusion by EKF in semi-outdoor structured environment. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1222–1227. IEEE.

Chan, H. (2004). *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Mariland, Baltimore County.

Chao, H., Coopmans, C., Di, L., and Chen, Y. (2010). A comparative evaluation of low-cost IMUs for unmanned autonomous systems. In *2010 IEEE Conference on Multisensor Fusion and Integration*, pages 211–216. IEEE.

Chen, H., Finin, T., and Joshi, A. (2003). An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(3):197–207.

Cilla, R., Patricio, M. a., Berlanga, A., and Molina, J. M. (2011). Improving the accuracy of action classification using view-dependent context information. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6679 LNAI(PART 2):136–143.

Cohen, O. and Edan, Y. (2004). Adaptive fuzzy logic algorithms for sensor fusion mapping. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 3, pages 2326–2331. IEEE.

Cohen, O. and Edan, Y. (2008). A sensor fusion framework for online sensor and algorithm selection. *Robotics and Autonomous Systems*, 56(9):762–776.

Costa, P. D. (2007). *Architectural Support for Context-Aware Applications*. PhD thesis, Telematica Instituut.

da Rocha, R. C. A., Endler, M., and de Siqueira, T. S. (2008). Middleware for ubiquitous context-awareness. In *Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing - MPAC '08*, pages 43–48, New York, New York, USA. ACM Press.

D'Aquin, M., Nikolov, A., and Motta, E. (2010). How much semantic data on small devices? In *EKAW'10 Proceedings of the 17th international conference on Knowledge engineering and management by the masses*, pages 565–575. Springer-Verlag.

Dasarathy, B. (1997). Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38.

de Ley, E. and Jacobs, D. (2011). Rules-based analysis with JBoss Drools: adding intelligence to automation. In *Proceedings of ICALEPCS 2011*.

Dey, A. K. (2000). *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology.

Ditzel, M., van der Broek, S., Hanckmann, P., and van Iersen, M. (2011). Dafne – a distributed and adaptive fusion engine. In Corchado, E., Kurzyński, M., and Woźniak, M., editors, *Proceedings 6th of the Hybrid Artificial Intelligence Systems International Conference (HAIS), part 2*, volume 6679, pages 100–109, Wroclaw, Poland. Springer Berlin Heidelberg.

Dixon, C., Mahajan, R., Agarwal, S., Brush, A. J., Lee, B., Saroiu, S., and Bahl, P. (2012). An operating system for the home. In *NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, page 25, Boston, MA. USENIX Association.

Douc, R. and Cappe, O. (2005). *Comparison of resampling schemes for particle filtering*. IEEE.

Durrant-Whyte, H. F. (1988). Sensor Models and Multisensor Integration. *The International Journal of Robotics Research*, 7(6):97–113.

Dwiggins, B. H. (1968). *Automotive steering systems*. Delmar Publishers.

Engelmore, R. and Morgan, T. (1988). *Blackboard Systems*. Addison-Wesley Longman Publishing Co., Inc.

Franz, A., Mista, R., Bakken, D., Dyreson, C., and Medidi, M. (2002). Mr. Fusion: a programmable data fusion middleware subsystem with a tunable statistical profiling service. In *Proceedings International Conference on Dependable Systems and Networks*, pages 273–278. IEEE Comput. Soc.

Garcia, J., Gomez-Romero, J., Patricio, M., Molina, J., and Rogova, G. (2011). On the representation and exploitation of context knowledge in a harbor surveillance scenario.

Gomez-Romero, J., Garcia, J., Kandefer, M., Llinas, J., Molina, J., Patricio, M., Prentice, M., and Shapiro, S. (2010). Strategies and techniques for use and exploitation of Contextual Information in high-level fusion architectures.

Gomez-Romero, J., Patricio, M., Garcia, J., and Molina, J. (2009). Ontological representation of context knowledge for visual data fusion.

Gomez-Romero, J., Serrano, M. A., Garcia, J., Molina, J. M., and Rogova, G. (2014). Context-based multi-level information fusion for harbor surveillance. *Information Fusion*, 21:173–186.

Gómez-Romero, J., Serrano, M. A., García, J., Molina, J. M., and Rogova, G. (2015). Context-based multi-level information fusion for harbor surveillance. *Information Fusion*, 21:173–186.

Gomez-Romero, J., Serrano, M. A., Patricio, M. A., Garcia, J., and Molina, J. M. (2011). Context-based scene recognition from visual data in smart homes: an Information Fusion approach. *Personal and Ubiquitous Computing*, 16(7):835–857.

Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113.

Gribble, S. D., Welsh, M., Brewer, E. A., and Culler, D. (1999). The multispace: an evolutionary platform for infrastructural services. In *ATEC '99 Proceedings of the annual conference on USENIX Annual Technical Conference*, page 12. USENIX Association Berkeley.

Hafner, P., Wieser, M., and Kühtreiber, N. (2011). Quality Assessment of Different GNSS/IMS-Integrations. *Österreichische Zeitschrift für Vermessung & Geoinformation*, pages 89–99.

Hall, D. D. L. and McMullen, S. A. H. (2004). *Mathematical techniques in multisensor data fusion*. Artech House.

Han, H., Yu, J., Zhu, H., Chen, Y., Yang, J., Zhu, Y., Xue, G., and Li, M. (2014). SenSpeed: Sensing Driving Conditions to Estimate Vehicle Speed in Urban Environments. In *Proceedings of the IEEE International Conference on Computer Communications*.

Henricksen, K., Indulska, J., and A., R. (2002). Modeling Context Information in Pervasive Computing Systems. In *First International Conference on Pervasive Computing*, pages 167–180. Springer-Verlag.

Higgins, W. (1975). A Comparison of Complementary and Kalman Filtering. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11(3):321–325.

Hilal, A. R. and Basir, O. A. (2014). A Scalable Sensor Management Architecture Using BDI Model for Pervasive Surveillance. *IEEE Systems Journal*, PP(99):1–13.

Hong, J. and Landay, J. (2001). An Infrastructure Approach to Context-Aware Computing. *Human-Computer Interaction*, 16(2):287–303.

Indulska, J. and Sutton, P. (2003). Location management in pervasive systems. In *Conferences in Research and Practice in Information Technology Series; Vol. 21*, pages 143–151. Australian Computer Society, Inc.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35.

Klein, L. a. (1999). *Sensor and Data Fusion Concepts and Applications*. Society of Photo-Optical Instrumentation Engineers (SPIE).

Kumar, R., Wolenetz, M., Agarwalla, B., Shin, J., Hutto, P., Paul, A., and Ramachandran, U. (2003). DFuse : A Framework for Distributed Data Fusion. In *Sensing Systems 2003*, pages 114–125, Los Angeles, California. ACM.

Lacy, L. W. (2005). *Owl: Representing Information Using the Web Ontology Language*. Trafford Publishing.

Lee, H. and Zeigler, B. P. (2010). System Entity Structure Ontological Data Fusion Process Integrated with C2 Systems. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 7(4):206–225.

Li, M., Cai, W., and Tan, Z. (2006). A region-based multi-sensor image fusion scheme using pulse-coupled neural network. *Pattern Recognition Letters*, 27(16):1948–1956.

Li, W. and Leung, H. (2003). Constrained unscented Kalman filter based fusion of GPS/INS/digital map for vehicle localization. *Intelligent Transportation Systems, 2003. . . .* , pages 1362–1367.

Liggins, M. E., Llinas, J., and Hall, D. L. (2008). *Handbook of Multisensor Data Fusion: Theory and Practice*. CRC Press, 2nd edition.

Liu, G. and Gingras, D. (2007). Adaptive and Reconfigurable Data Fusion Architectures in Vehicle Positioning Navigation Systems. Technical report, Society of Automotive Engineers (SAE).

Llinas, J. (2010). A Survey and Analysis of Frameworks and Framework Issues for Information Fusion Applications. In Graña Romay, Manuel and Corchado, Emilio and Garcia Sebastian, M., editor, *Hybrid Artificial Intelligence Systems*, pages 14–23. Springer Berlin Heidelberg.

Llinas, J., Bowman, C., Rogova, G., Steinberg, A., Waltz, E., and White, F. (2000). Revisiting the JDL Data Fusion Model II. *Processing*.

Markin, M., Harris, C., Bernhardt, J., Austin, M., Bedworth, P., Greenway, R., Johnston, R., Little, A., and Lowe, D. (1997). Technology Foresight on Data Fusion and Data Processing. Technical report, Royal Aeronautical Society.

Martí, E., García, J., and Molina, J. (2011a). Opportunistic multisensor fusion for robust navigation in smart environments. In *User-Centric Technologies and Applications. Proceedings of CONTEXTS 2011 Workshop*. Springer.

Martí, E., García, J., and Molina, J. M. (2011b). A Regularised Particle Filter for Context-Aware Sensor Fusion Applications. In *INFORMATIK 2011 - 41th Annual Conference of the Gesellschaft für Informatik*.

Martí, E. D., García, J., and Molina, J. M. (2011c). Context-awareness at the service of Sensor Fusion Systems: Inverting the Usual Scheme. In *Proceedings of the ISCIF Conference*.

Martí, E. D., García, J., and Molina, J. M. (2011d). Neigborhood-based Regularization of Proposal Distribution for Improving Resampling Quality in Particle Filters. In *14th International Conference on Information Fusion (Fusion 2011)*, page 7, Chicago, USA.

Martí, E. D., Martín, D., García, J., de la Escalera, A., Molina, J. M., and Armingol, J. M. (2012). Context-aided sensor fusion for enhanced urban navigation. *Sensors (Basel, Switzerland)*, 12(12):16802–37.

Maslov, I. V. and Gertner, I. (2006). Multi-sensor fusion: an Evolutionary algorithm approach. *Information Fusion*, 7(3):304–330.

Morales, Y. (2008). Vehicle localization in outdoor woodland environments with sensor fault detection. In *2008 IEEE International Conference on Robotics and Automation*, pages 449–454. IEEE.

Morrison, A., Renaudin, V., Bancroft, J. B., and Lachapelle, G. (2012). Design and testing of a multi-sensor pedestrian location and navigation platform. *Sensors (Basel, Switzerland)*, 12(3):3720–38.

Murphy, R. (1998). Dempster-Shafer theory for sensor fusion in autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 14(2):197–206.

Petovello, M. G., O'Driscoll, C., and Lachapelle, G. (2007). Ultra-Tight GPS/INS for Carrier Phase Positioning in Weak Signal Environment. In *Proceedings of the NATO RTO SET-104 Symposium on Military Capabilities Enabled by Advances in Navigation Sensors*.

Pozo, A., García, J., Patricio, M. A., and Molina, J. M. (2011). Group Behavior Recognition in Context-Aware Systems. In Cabestany, J., Rojas, I., and Joya, G., editors, *Advances in Computational Intelligence*, chapter Lecture No, pages 645–652. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ravindra, B. and Wang, J. (2005). Ultra-tight GPS/INS/PL Integration: A System Concept and Performance Analysis. *GPS Solutions*, 13(1):75–82.

Rezaei, S. and Sengupta, R. (2007). Kalman Filter-Based Integration of DGPS and Vehicle Sensors for Localization. *IEEE Transactions on Control Systems Technology*, 15(6):1080–1088.

Ricquebourg, V. and Delahoche, L. (2008). Anomalies recognition in a context aware architecture based on TBM approach. In *Proceedings of the 11th International Confeerence on Information Fusion*, pages 63–70.

Salber, D., Dey, A. K., and Abowd, G. D. (1999). The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, pages 434–441, New York, New York, USA. ACM Press.

Sanchez, A., Patricio, M., Garcia, J., and Molina, J. (2007). Video tracking improvement using context-based information. In *2007 10th International Conference on Information Fusion*, pages 1–7. IEEE.

Sasiadek, J., Wang, Q., and Zeremba, M. (2000). Fuzzy adaptive Kalman filtering for INS/GPS data fusion. In *Proceedings of the 2000 IEEE International Symposium on Intelligent Control. Held jointly with the 8th IEEE Mediterranean Conference on Control and Automation (Cat. No.00CH37147)*, pages 181–186. IEEE.

Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32.

Shadbolt, N., Berners-Lee, T., and Hall, W. (2006). The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3):96–101.

Sharaf, R., Noureldin, A., Osman, A., and El-Sheimy, N. (2005). Online INS/GPS integration with a radial basis function neural network. *IEEE Aerospace and Electronic Systems Magazine*, 20(3):8–14.

Sheng, Q. and Benatallah, B. (2005). ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development. In *International Conference on Mobile Business (ICMB'05)*, pages 206–212. IEEE.

Shladover, S. E. (2009). Cooperative (Rather Than Autonomous) Vehicle-highway Automation Systems. *IEEE Intelligent Transportation Systems Magazine*, 1(1).

Shulsky, A. M. and Schmitt, G. J. (2002). *Silent Warfare: Understanding the World of Intelligence, 2nd ed*. Brassey S, 3rd edition.

Snidaro, L., Visentini, I., Llinas, J., and Foresti, G. L. (2013). Context in fusion: some considerations in a JDL perspective. In *Proceedings of the 16th International Conference on Information Fusion*, pages 115–120, Istanbul. IEEE.

Solaiman, B. (1998). Multisensor/Contextual Data fusion through membership functions revision. Application to land-cover SAR data classification. In *Proceedings of the 1st International Conference on Multisource-Multisensor Information Fusion*, pages 673–680, Las Vegas. International Society of Information Fusion.

Solano, M. A., Ekwaro-Osire, S., and Tanik, M. M. (2012). High-Level fusion for intelligence applications using Recombinant Cognition Synthesis. *Information Fusion*, 13(1):79–98.

Steinberg, A. and Rogova, G. (2008). Situation and context in data fusion and natural language understanding. In *11th International Conference on Information Fusion*, pages 1–8, Cologne. IEEE.

Stover, J., Hall, D., and Gibson, R. (1996). A fuzzy-logic architecture for autonomous multisensor data fusion. *IEEE Transactions on Industrial Electronics*, 43(3):403–410.

Strang, T. and Linnhoff-Popien, C. (2004). A Context Modeling Survey. In *Workshop on Advanced Context Modelling, Reasoning and Management, Sixth International Conference on Ubiquitous Computing*, pages 1–8.

Uhlmann, J. K. and Julier, S. J. (1997). A New Extension of the Kalman Filter to Nonlinear Systems. In *AeroSense: The 11th Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, Orlando, FL. SPIE.

Van der Merwe, R., de Freitas, N., Doucet, A., and Wan, E. (2000). The unscented particle filter. Technical report, Cambridge University Engineering Department.

Waltz, E. L. and Llinas, J. (1990). *Multisensor Data Fusion*. Artech House, Inc.

Wan, E. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158. IEEE.

Wang, J.-H. and Gao, Y. (2005). Multi-sensor data fusion for land vehicle attitude estimation using a fuzzy expert system. *Data Science Journal*, 4:127–139.

Wang, Z., Ma, Y., and Gu, J. (2010). Multi-focus image fusion using PCNN. *Pattern Recognition*, 43(6):2003–2016.

Weiser, M. (1999). The computer for the 21 st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11.

Welch, G. and Bishop, G. (1995). An Introduction to the Kalman Filter.

Wendel, J. and Trommer, G. F. (2004). Tightly coupled GPS/INS integration for missile applications. *Aerospace Science and Technology*, 8(7):627–634.

Winograd, T. (2001). Architectures for Context. *Human-Computer Interaction*, 16(2):401–419.

Wu, H. (2003). *Sensor Data Fusion for Context-Aware Computing Using Dempster-Shafer Theory*. PhD thesis, Carnegie Mellon University.

Zhang, P., Gu, J., Milios, E., and Huynh, P. (2005). Navigation with IMU/GPS/digital compass with unscented Kalman filter. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 3, pages 1497–1502. IEEE.