An Introduction to Sensor Fusion

Research Report 47/2001

Wilfried Elmenreich

Institut für Technische Informatik
Vienna University of Technology, Austria
wil@vmars.tuwien.ac.at

November 19, 2002

Abstract

This paper gives an overview over the basic concepts of sensor fusion. First we investigate on definitions and terminology and then discuss motivations and limitations of sensor fusion. The next sections present a survey on architectures for sensor fusion and describe algorithms and methods like the Kalman Filter, inference methods, and the application of sensor fusion in robotic vision. Sensor fusion offers a great opportunity to overcome physical limitations of sensing systems. An important point will be the reduction of software complexity, in order to hide the properties of the physical sensors behind a sensor fusion layer.

Keywords: sensor fusion, information fusion, terminology, fusion model, Kalman Filter, inference, occupancy grids.

1 Introduction

An animal recognizes its environment by the evaluation of signals from multiple and multifaceted sensors. Nature has found a way to integrate information from multiple sources to a reliable and feature-rich recognition. Even in case of sensor deprivation, systems are able to compensate for lacking information by reusing data obtained from sensors with an overlapping scope. Humans for example combine signals from the five body senses (sight, sound, smell, taste, and touch) with knowledge of the environment to create and update a dynamic model of the world. Based on this information the individual interacts with the environment and makes decisions about present and future actions [33].

This natural ability to fuse multi-sensory data has evolved to a high degree in many animal species and is in use for millions of years. Today the application of fusion concepts in technical areas has constituted a new discipline, that spans over many fields of science.

The objective of this paper is to give an overview on principles, architectures and methods of sensor fusion. Section 2 will first investigate on definitions and

terminology and then discuss motivations and limitations of sensor fusion. Section 3 presents a survey on architectures for sensor fusion. Section 4 describes algorithms and methods like the Kalman Filter, inference methods, sensor fusion in robotic map-building, and the construction of reliable abstract sensors. The paper is concluded in section 5.

2 Principles of Sensor Fusion

There is some confusion in the terminology for fusion systems. The terms "sensor fusion", "data fusion", "information fusion", "multi-sensor data fusion", and "multi-sensor integration" have been widely used in the technical literature to refer to a variety of techniques, technologies, systems, and applications that use data derived from multiple information sources. Fusion applications range from real-time sensor fusion for the navigation of mobile robots to the off-line fusion of human or technical strategic intelligence data [59].

Several attempts have been made to define and categorize fusion terms and techniques. In [72], Wald proposes the term "data fusion" to be used as the overall term for fusion. However, while the concept of data fusion is easy to understand, its exact meaning varies from one scientist to another. Wald uses "data fusion" for a formal framework that comprises means and tools for the alliance of data originating from different sources. It aims at obtaining information of superior quality; the exact definition of *superior quality* depends on the application. The term "data fusion" is used in this meaning by the Geoscience and Remote Sensing Society[1], by the U. S. Department of Defense [69], and in many papers regarding motion tracking, remote sensing, and mobile robots. Unfortunately, the term has not always been used in the same meaning during the last years [64]. In some fusion models, "data fusion" is used to denote fusion of raw data [18].

There are classic books on fusion like "Multisensor Data Fusion" [74] by Waltz and Llinas and Hall's "Mathematical Techniques in Multisensor Data Fusion" [34] that propose an extended term, "multisensor data fusion". It is defined there as *the technology concerned with the combination of how to combine data from multiple (and possible diverse) sensors in order to make inferences about a physical event, activity, or situation* [34, page *ix*]. However, in both books, also the term "data fusion" is mentioned as being equal with "multisensor data fusion" [34].

To avoid confusion on the meaning, Dasarathy decided to use the term "information fusion" as the overall term for fusion of any kind of data [20]. The term "information fusion" had not been used extensively before and thus had no baggage of being associated with any single aspect of the fusion domain. The fact that "information fusion" is also applicable in the context of data mining and data base integration is not necessarily a negative one as the effective meaning is unaltered: information fusion is an all-encompassing term covering all aspects of the fusion field (except nuclear fusion or fusion in the music world).

[1] http://www.dfc-grss.org

A literal definition of information fusion can be found at the homepage of the International Society of Information Fusion[2]:

Information Fusion *encompasses theory, techniques and tools conceived and employed for exploiting the synergy in the information acquired from multiple sources (sensor, databases, information gathered by human, etc.) such that the resulting decision or action is in some sense better (qualitatively or quantitatively, in terms of accuracy, robustness, etc.) than would be possible if any of these sources were used individually without such synergy exploitation.*

By defining a subset of information fusion, the term *sensor fusion* is introduced as:

Sensor Fusion is the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually.

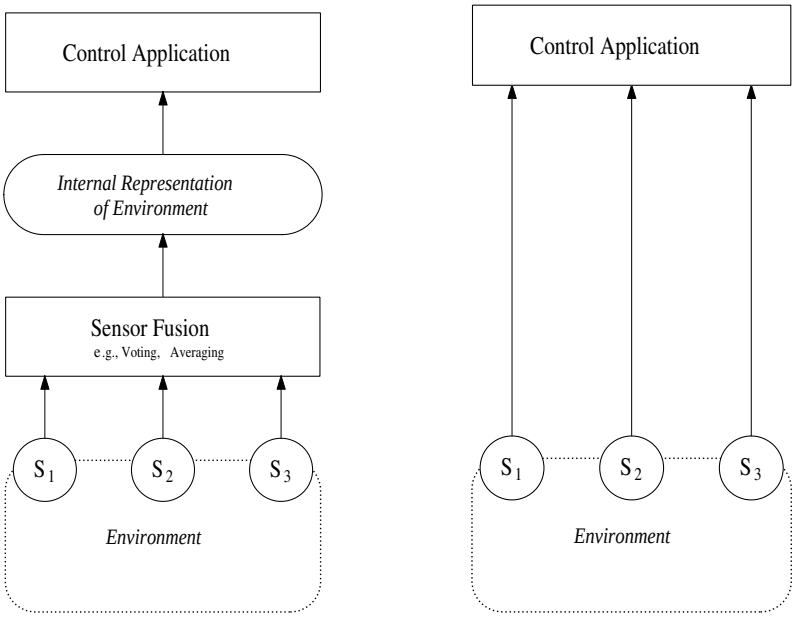
The data sources for a fusion process are not specified to originate from identical sensors. McKee distinguishes *direct fusion*, *indirect fusion* and fusion of the outputs of the former two [49]. Direct fusion means the fusion of sensor data from a set of heterogeneous or homogeneous sensors, soft sensors, and history values of sensor data, while indirect fusion uses information sources like a priori knowledge about the environment and human input. Therefore, *sensor fusion* describes direct fusion systems, while *information fusion* also encompasses indirect fusion processes.

Since "data fusion" still is a standard term in the scientific community for earth image data processing, it is recommended not to use the stand-alone term "data fusion" in the meaning of "low-level data fusion". Thus, unless "data fusion" is meant as proposed by the earth science community, a prefix like "low-level" or "raw" would be adequate.

The sensor fusion definition above does not require that inputs are produced by multiple sensors, it only says that sensor data or data derived from sensor data have to be combined. For example, the definition also encompasses sensor fusion systems with a single sensor that take multiple measurements subsequently at different instants which are then combined.

Another frequently used term is *multisensor integration*. Multisensor integration means the synergistic use of sensor data for the accomplishment of a task by a system. Sensor fusion is different to multisensor integration in the sense that it includes the actual combination of sensory information into one representational format [63, 44]. The difference between sensor fusion and multisensor integration is outlined in figure 1. The circles S_1, S_2, and S_3 depict physical sensors that provide an interface to the process environment. Block diagram 1(a) shows that the sensor data is converted by a sensor fusion block into a respective representation of the variables of the process environment.

[2] http://www.inforfusion.org/mission.htm



(a) Sensor fusion (b) Multisensor integration

Figure 1: Block diagram of sensor fusion and multisensor integration

These data is then used by a control application. In contrast, figure 1(b) illustrates the meaning of multisensor integration, where the different sensor data are directly processed by the control application.

2.1 Motivation for Sensor Fusion

Systems that employ sensor fusion methods expect a number of benefits over single sensor systems. A physical sensor measurement generally suffers from the following problems:

Sensor Deprivation: The breakdown of a sensor element causes a loss of perception on the desired object.

Limited spatial coverage: Usually an individual sensor only covers a restricted region. For example a reading from a boiler thermometer just provides an estimation of the temperature near the thermometer and may fail to correctly render the average water temperature in the boiler.

Limited temporal coverage: Some sensors need a particular set-up time to perform and to transmit a measurement, thus limiting the maximum frequency of measurements.

Imprecision: Measurements from individual sensors are limited to the precision of the employed sensing element.

Uncertainty: Uncertainty, in contrast to imprecision, depends on the object being observed rather than the observing device. Uncertainty arises when features are missing (e. g., occlusions), when the sensor cannot measure all relevant attributes of the percept, or when the observation is ambiguous [51]. A single sensor system is unable to reduce uncertainty in its perception because of its limited view of the object [30].

As an example, consider a distance sensor mounted at the rear of a car in order to assist backing the car into a parking space. The sensor can only provide information about objects in front of the sensor but not beside, thus the spatial coverage is limited. We assume that the sensor has an update time of one second. This is a limited temporal coverage that is significant for a human driver. Finally, the sensor does not provide unlimited precision, for example its measurements could be two centimeters off the actual distance to the object. Uncertainty arises, if the object behind the rear of the car is a small motorcycle and the driver cannot be sure, if the sensor beam hits the object and delivers a correct measurement with the specified precision or if the sensor beam misses the object, delivering a value suggesting a much different distance.

One solution to the listed problems is to use sensor fusion. The standard approach to compensate for sensor deprivation is to build a fault-tolerant unit of at least three identical units with a voter [71] or at least two units showing fail-silent behavior [42]. Fail-silent means that a component produces either correct results or, in case of failure, no results at all. In a sensor fusion system robust behavior against sensor deprivation can be achieved by using sensors with overlapping views of the desired object. This works with a set of sensors of the same type as well as with a suite of heterogeneous sensors.

The following advantages can be expected from the fusion of sensor data from a set of heterogeneous or homogeneous sensors [10, 33]:

Robustness and reliability: Multiple sensor suites have an inherent redundancy which enables the system to provide information even in case of partial failure.

Extended spatial and temporal coverage: One sensor can look where others cannot respectively can perform a measurement while others cannot.

Increased confidence: A measurement of one sensor is confirmed by measurements of other sensors covering the same domain.

Reduced ambiguity and uncertainty: Joint information reduces the set of ambiguous interpretations of the measured value.

Robustness against interference: By increasing the dimensionality of the measurement space (e. g., measuring the desired quantity with optical sensors and ultrasonic sensors) the system becomes less vulnerable against interference.

Improved resolution: When multiple independent measurements of the same property are fused, the resolution of the resulting value is better than a single sensor's measurement.

In [58], the performance of sensor measurements obtained from an appropriate fusing process is compared to the measurements of the single sensor. According to this work, an optimal fusing process can be designed, if the distribution function describing measurement errors of one particular sensor is precisely known. This optimal fusing process performs at least as well as the best single sensor.

A further advantage of sensor fusion is the possibility to reduce system complexity. In a traditionally designed system the sensor measurements are fed into the application, which has to cope with a big number of imprecise, ambiguous and incomplete data streams. In a system where sensor data is preprocessed by fusion methods, the input to the controlling application can be standardized independently of the employed sensor types, thus facilitating application implementation and providing the possibility of modifications in the sensor system regarding number and type of employed sensors without modifications of the application software [28].

2.2 Limitations of Sensor Fusion

Evolution has developed the ability to fuse multi-sensory data into a reliable and feature-rich recognition. Nevertheless, sensor fusion is not an omnipotent method. Fowler stated a harsh criticism in 1979:

One of the grabbiest concepts around is synergism. Conceptual application of synergism is spread throughout military systems but is most prevalent in the "multisensor" concept. This is a great idea provided the input data are a (sic!) *good quality. Massaging a lot of crummy data doesn't produce good data; it just requires a lot of extra equipment and may even reduce the quality of the output by introducing time delays and/or unwarranted confidence.* [...] *It takes more than correlation and fusion to turn sows' ears into silk purses.* [31, page 5]

Since this has been published, many people tried to prove the opposite. Nahin and Pokoski [52] presented a theoretical proof that the addition of sensors improves the performance in the specific cases for majority vote and maximum likelihood theory in decision fusion. Performance was defined as probability of taking the right decision without regarding the effort on processing power and communication.

In contrast, Tenney and Sandell considered communication bandwidth for distributed fusion architectures. In their work they showed that a distributed system is suboptimal in comparison to a completely centralized processing scheme with regard to the communication effort [67].

An essential criterium for the possible benefit of sensor fusion is a comprehensive set of performance measures. Theil, Kester, and Bossé presented measures of performance for the fields of detection, tracking, and classification. Their work suggests measuring the quality of the output data and the reaction time [68].

Dasarathy investigated the benefits on increasing the number of inputs to a sensor fusion process in [19]. Although the analysis is limited on the augmentation of two-sensor systems by an extra sensor, the work shows that increasing the number of sensors may lead to a performance gain or loss depending on the

sensor fusion algorithm.

It can be concluded from the existing knowledge on sensor fusion performance that in spite of the great potentials of sensor fusion slight skepticism on "perfect" or "optimal" fusion methods is appropriate.

2.3 Types of Sensor Fusion

The following paragraphs present common categorizations for sensor fusion applications by different aspects.

2.3.1 C^{3}I versus embedded real-time applications

There exists an important dichotomy in research on sensor fusion for C^{3}I (*command, control, communications, and intelligence*) oriented applications and sensor fusion which is targeted at real-time embedded systems. The C^{3}I oriented research focuses primarily on intermediate and high level sensor fusion issues while onboard applications concentrate on low-level fusion. Table 1 compares some central issues between C^{3}I and embedded fusion applications (cf. [59]).

2.3.2 Three-Level Categorization

Fusion processes are often categorized in a three-level model distinguishing *low*, *intermediate*, and *high level fusion*.

Low-level fusion or *raw data fusion* (confer to section 2 on the double meaning of "data fusion") combines several sources of raw data to produce new data that is expected to be more informative than the inputs.

Intermediate-level fusion or *feature level fusion* combines various features such as edges, corners, lines, textures, or positions into a feature map that may then be used for segmentation and detection.

High-level fusion, also called *decision fusion* combines decisions from several experts. Methods of decision fusion include voting, fuzzy-logic, and statistical methods.

	Onboard fusion	C^{3}I fusion
Time scale	milliseconds	seconds... minutes
Data type	sensor data	also linguistic or symbolic data
Man-machine interaction	optional	frequently
Database size	small to moderate	large to very large
Level of abstraction	low	high

Table 1: Comparison between C^{3}I and embedded fusion applications

2.3.3 Categorization Based on Input/Output

Dasarathy proposed a refined categorization based on the three-level model in [18]. It categorizes fusion processes derived from the abstraction level of the processes' input and output data.

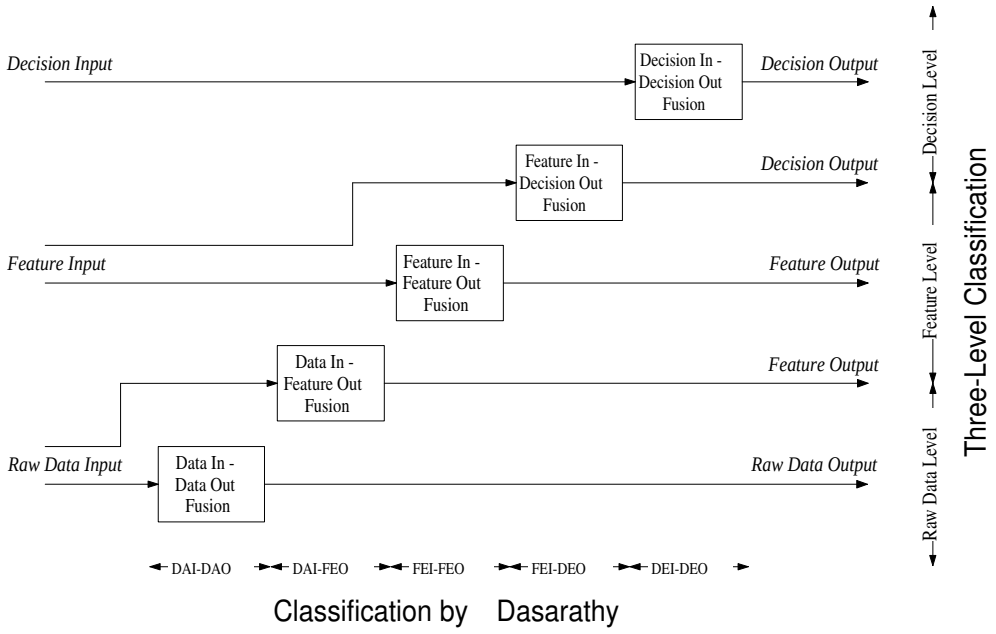


Figure 2: Alternative fusion characterizations based on input/output

The reason for the Dasarathy model was the existence of fusion paradigms where the input and output of the fusion process belong to different levels. Examples are feature selection and extraction, since the processed data comes from the raw data level and the results belong to the feature level. For example, pattern recognition and pattern processing operates between feature and decision level. These ambiguous fusion paradigms sometimes have been assigned according to the level of their input data and sometimes according to the level of their output data.

To avoid these categorization problems, Dasarathy extended the three-level view to five fusion categories defined by their input/output characteristics. Figure 2 depicts the relation between the three-level categorization and the Dasarathy model.

2.3.4 Categorization Based on Sensor Configuration

Sensor fusion networks can also be categorized according to the type of sensor configuration. Durrant-Whyte [23] distinguishes three types of sensor configuration:

Complementary: A sensor configuration is called *complementary* if the sensors do not directly depend on each other, but can be combined in order to give a more complete image of the phenomenon under observation. This

resolves the incompleteness of sensor data. An example for a complementary configuration is the employment of multiple cameras each observing disjunct parts of a room as applied in [35]. Generally, fusing complementary data is easy, since the data from independent sensors can be appended to each other [12].

Sensor S_2 and S_3 in figure 3 represent a complementary configuration, since each sensor observes a different part of the environment space.

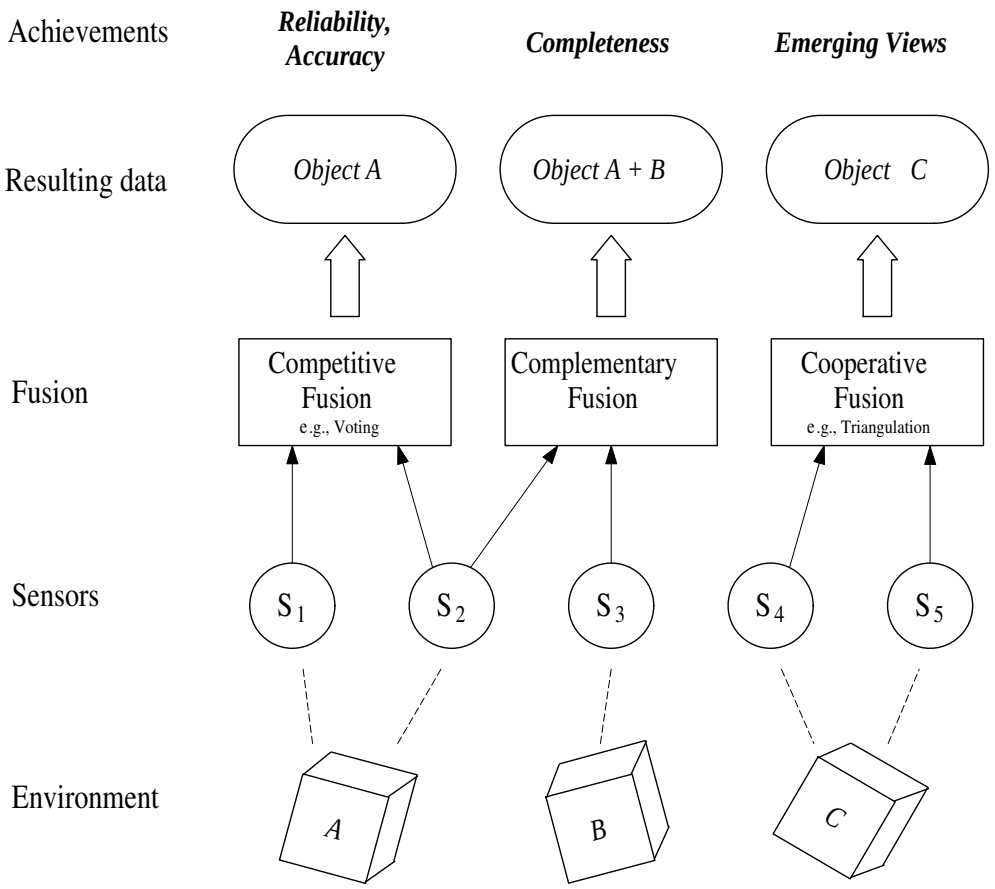


Figure 3: Competitive, complementary, and cooperative fusion

Competitive: Sensors are configured *competitive* if each sensor delivers independent measurements of the same property. Visser and Groen [70] distinguish two possible competitive configurations – the fusion of data from different sensors or the fusion of measurements from a single sensor taken at different instants. Competitive sensor configuration is also called a *redundant* configuration [44].

A special case of competitive sensor fusion is *fault tolerance*. Fault tolerance requires an exact specification of the service and the failure modes of the system. In case of a fault covered by the fault hypothesis, the system still has to provide its specified service. Examples for fault-tolerant

configurations are N-modular redundancy [53] and other schemes where a certain number of faulty components are tolerated [55, 47].

In contrast to fault tolerance, competitive configurations can also provide *robustness* to a system. Robust systems provide a degraded level of service in the presence of faults. While this *graceful degradation* is weaker than the achievement of fault tolerance, the respective algorithms perform better in terms of resource needs and work well with heterogeneous data sources [4]. An example for architectures that supports heterogeneous competitive sensors can be found in [54] and [27] where confidence tags are used to indicate the dependability of an observation.

Sensor S_1 and S_2 in figure 3 represent a competitive configuration, where both sensors redundantly observe the same property of an object in the environment space.

Cooperative: A *cooperative* sensor network uses the information provided by two independent sensors to derive information that would not be available from the single sensors. An example for a cooperative sensor configuration is stereoscopic vision – by combining two-dimensional images from two cameras at slightly different viewpoints a three-dimensional image of the observed scene is derived. According to Brooks and Iyengar, cooperative sensor fusion is the most difficult to design, because the resulting data are sensitive to inaccuracies in all individual participating sensors [12]. Thus, in contrast to competitive fusion, cooperative sensor fusion generally decreases accuracy and reliability.

Sensor S_4 and S_5 in figure 3 represent a cooperative configuration. Both sensors observe the same object, but the measurements are used to form an emerging view on object C that could not have been derived from the measurements of S_4 or S_5 alone.

These three categories of sensor configuration are not mutually exclusive. Many applications implement aspects of more than one of the three types. An example for such a hybrid architecture is the application of multiple cameras that monitor a given area. In regions covered by two or more cameras the sensor configuration can be competitive *or* cooperative. For regions observed by only one camera the sensor configuration is complementary.

3 Architectures for Sensor Fusion

Due to the fact that sensor fusion models heavily depend on the application, no generally accepted model of sensor fusion exists until today [7]. According to Kam, Zhu, and Kalata, it is unlikely that one technique or one architecture will provide a uniformly superior solution [41]. In this survey, we focus on architectures which have been known for a relatively long period of time.

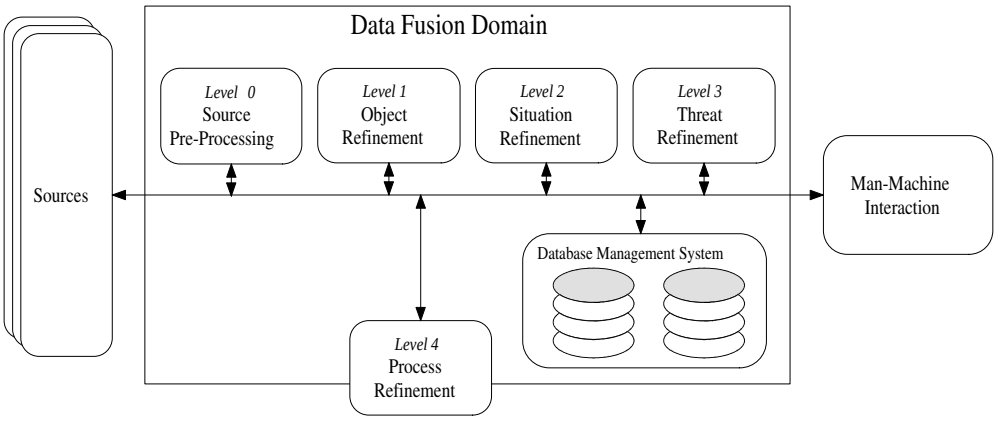


Figure 4: JDL fusion model (from [43])

3.1 The JDL Fusion Architecture

A frequently referred fusion model originates from the US Joint Directors of Laboratories (JDL). It was proposed in 1985 under the guidance of the Department of Defense (DoD). The *JDL model* [74] comprises five levels of data processing and a database, which are all interconnected by a bus. The five levels are not meant to be processed in a strict order and can also be executed concurrently. Figure 4 depicts the top level of the JDL data fusion process model. The elements of the model are described in the following:

Sources: The sources provide information from a variety of data sources, like sensors, *a priori* information, databases, human input.

Source preprocessing (Level 0): The task of this element is to reduce the processing load of the fusion processes by prescreening and allocating data to appropriate processes. Source preprocessing has later been labelled *level 0* [7].

Object refinement (Level 1): This level performs *data alignment* (transformation of data to a consistent reference frame and units), *association* (using correlation methods), *tracking* actual and future positions of objects, and *identification* using classification methods.

Situation refinement (Level 2): The situation refinement attempts to find a contextual description of the relationship between objects and observed events.

Threat refinement (Level 3): Based on *a priori* knowledge and predictions about the future situation this processing level tries to draw inferences about vulnerabilities and opportunities for operation.

Process refinement (Level 4): Level 4 is a meta process that monitors system performance (e. g., real-time constraints) and reallocates sensor and sources to achieve particular mission goals.

Database management system: The task of the database management system is to monitor, evaluate, add, update, and provide information for the fusion processes.

Man-machine interaction: This part provides an interface for human input and communication of fusion results to operators and users.

The JDL model has been very popular for fusion systems. Despite its origin in the military domain it can be applied to both military and commercial applications. The JDL model also has categorized processes related to a fusion system. However, the model suffers from the following drawbacks:

- It is a data-centered or information-centered model, which makes it difficult to extend or reuse applications built with this model.
- The model is very abstract, which makes it difficult to properly interpret its parts and to appropriately apply it to specific problems.
- The model is helpful for common understanding, but does not guide a developer in identifying the methods that should be used [43] – thus, the model does not help in developing an architecture for a real system.

The basic JDL model has also been improved and extended for various applications. Waltz showed, that the model does not address multi-image fusion problems and presented an extension that includes the fusion of image data [73]. Steinberg, Bowman, and White proposed revisions and expansions of the JDL model involving broadening the functional model, relating the taxonomy to fields beyond the original military focus, and integrating a data fusion tree architecture model for system description, design, and development [65].

3.2 Waterfall Fusion Process Model

The waterfall model, proposed in [45], emphasizes on the processing functions on the lower levels. Figure 5 depicts the processing stages of the waterfall model. The stages relate to the levels 0, 1, 2, and 3 of the JDL model as follows: Sensing and signal processing correspond to source preprocessing (level 0), feature extraction and pattern processing match object refinement (level 1), situation assessment is similar to situation refinement (level 2), and decision making corresponds to threat refinement (level 3).

Being thus very similar to the JDL model, the waterfall model suffers from the same drawbacks. While being more exact in analyzing the fusion process than other models, the major limitation of the waterfall model is the omission of any feedback data flow. The waterfall model has been used in the defense data fusion community in Great Britain, but has not been significantly adopted elsewhere [7].

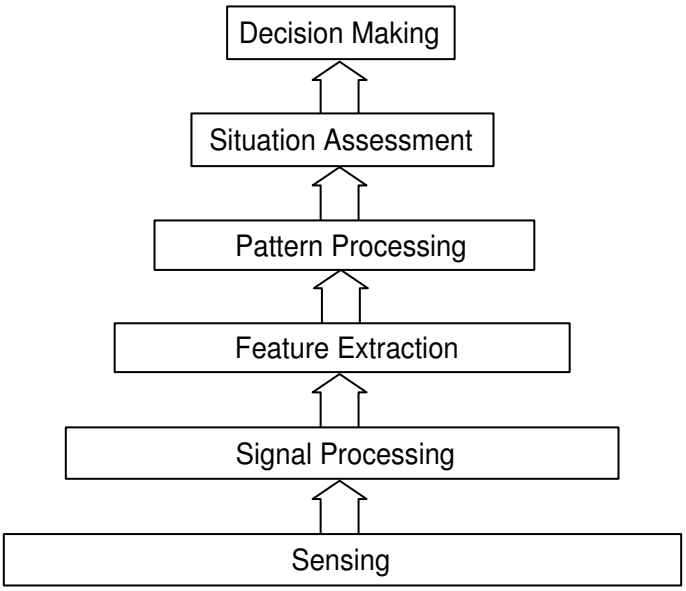


Figure 5: The waterfall fusion process model (from [45])

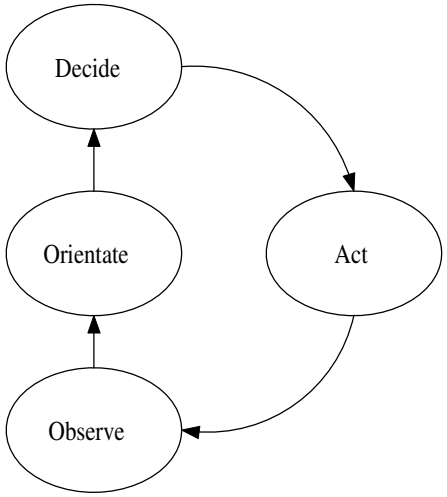


Figure 6: The Boyd (or OODA) loop

3.3 Boyd Model

Boyd has proposed a cycle containing four stages [11]. This *Boyd control cycle* or *OODA loop* (depicted in figure 6) represents the classic decision-support mechanism in military information operations. Because decision-support systems for situational awareness are tightly coupled with fusion systems [5], the Boyd loop has also been used for sensor fusion.

Bedworth and O'Brien compared the stages of the Boyd loop to the JDL [7]:

Observe: This stage is broadly comparable to source preprocessing in the JDL model.

Orientate: This stage corresponds to functions of the levels 1, 2, and 3 of the JDL model.

Decide: This stage is comparable to level 4 of the JDL model (Process refinement).

Act: Since the JDL model does not close the loop by taking the actuating part of the interaction into account, this stage has no direct counterpart in the JDL model.

The Boyd model represents the stages of a closed control system and gives an overview on the overall task of a system, but the model lacks of an appropriate structure for identifying and separating different sensor fusion tasks.

3.4 The LAAS Architecture

The LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes) architecture [1] was developed as an integrated architecture for the design and implementation of mobile robots with respect to real-time and code reuse. Due to the fact that mobile robot systems often employ sensor fusion methods, we briefly discuss the elements of the LAAS architecture (depicted in figure 7).

The architecture consists of the following levels [1]:

Logical robot level: The task of the logical robot level is to establish a hardware independent interface between the physical sensors and actuators and the functional level.

Functional level: The functional level includes all the basic built-in robot action and perception capabilities. The processing functions, such as image processing, obstacle avoidance, and control loops, are encapsulated into separate controllable communicating modules.

Execution control level: The execution control level controls and coordinates the execution of the functions provided by the modules according to the task requirements.

Decision level: The decision level includes the capabilities of producing the task plan and supervising its execution while being at the same time reactive to other events from the execution control level. Depending on the application, the decision level can be composed of several layers that provide different representation abstractions and have different temporal properties.

The LAAS architecture maps low-level and intermediate-level sensor fusion to modules at the functional level. High-level sensor fusion is represented in the decision level. The timing requirements are different at the decision level and the functional level (confer to section 2.3 on the difference between C^3I and real-time applications). While the architecture provides a good means for the partitioning of large systems into modules, it does not provide an appropriate real-time communication and representation of the fused data at the levels above

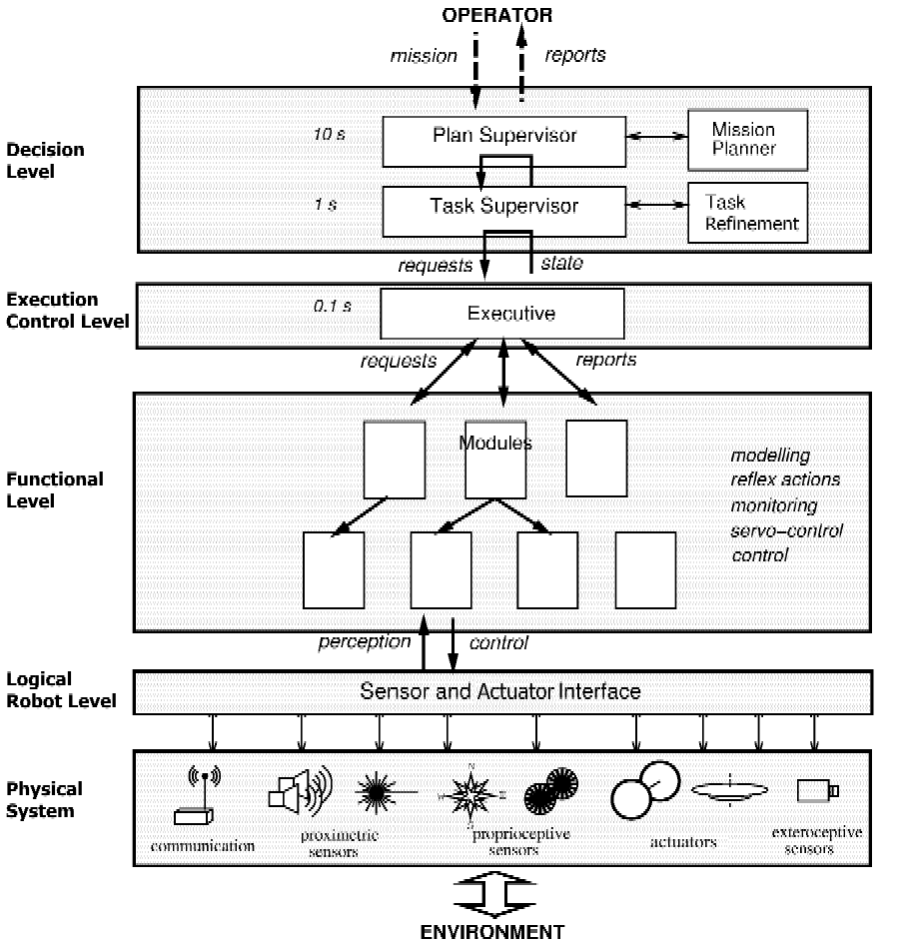


Figure 7: LAAS Architecture (from [1])

the functional level. In contrast to the JDL model, the LAAS architecture guides a designer well in implementing reusable modules as part of a real-time application.

3.5 The Omnibus Model

The omnibus model [7] has been presented in 1999 by Bedworth and O'Brien. Figure 8 depicts the architecture of the omnibus model. Unlike the JDL model, the omnibus model defines the ordering of processes and makes the cyclic nature explicit. It uses a general terminology that does not assume that the applications are defense-oriented. The model shows a cyclic structure comparable to the Boyd loop, but provides a much more fine-grained structuring of the processing levels. The model is intended to be used multiple times in the same application recursively at two different levels of abstraction. First, the model is used to characterize and structure the overall system. Second, the same structures are used to model the single subtasks of the system.

Although the hierarchical separation of the sensor fusion tasks is very so-

phisticated in the omnibus model, it does not support a horizontal partitioning into tasks that reflect distributed sensing and data processing. Thus, the model does not support a decomposition into modules that can be separately implemented, separately tested, and reused for different applications.

4 Methods and Applications

This section investigates on sensor fusion methods and applications that are related to this thesis.

4.1 Smoothing, Filtering, and Prediction

Generally, the task of a sensor is to provide information about a process variable in the environment by taking measurements. Since these measurements can be noisy and are – at least in digital systems – taken at discrete points in time, it is necessary to fuse multiple measurements to reconstruct the parameter of interest.

Given an observation vector $\vec{y}_k$ corresponding to time k, we want to estimate a process state vector $\vec{x}_{k+m}$. Depending on the time $k+m$, Åström and Wittenmark distinguish the following three cases [3]:

Smoothing ($m < 0$): The change of a process entity shall be reconstructed after a series of measurements has been performed. For each instant of interest, several measurements from previous, actual, and following instants are used in order to estimate the value of the process variable.

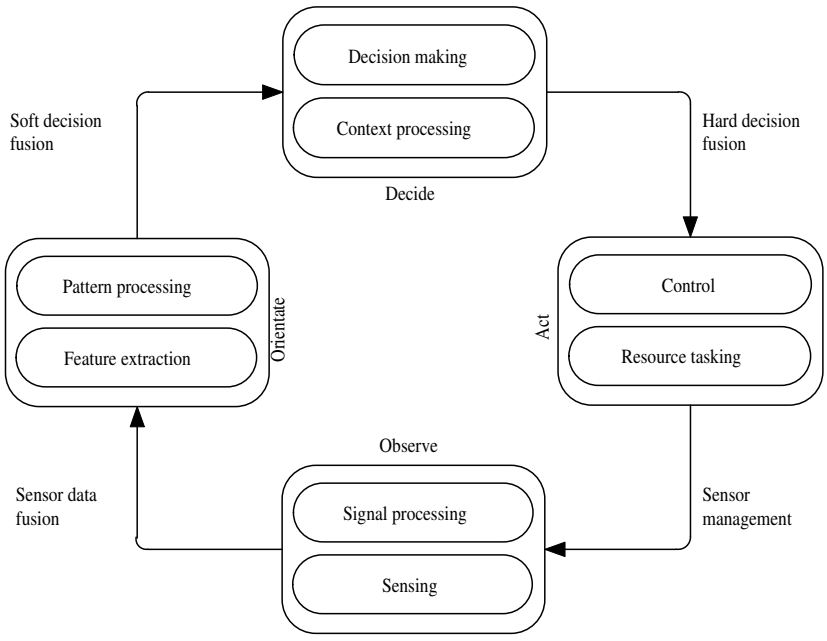


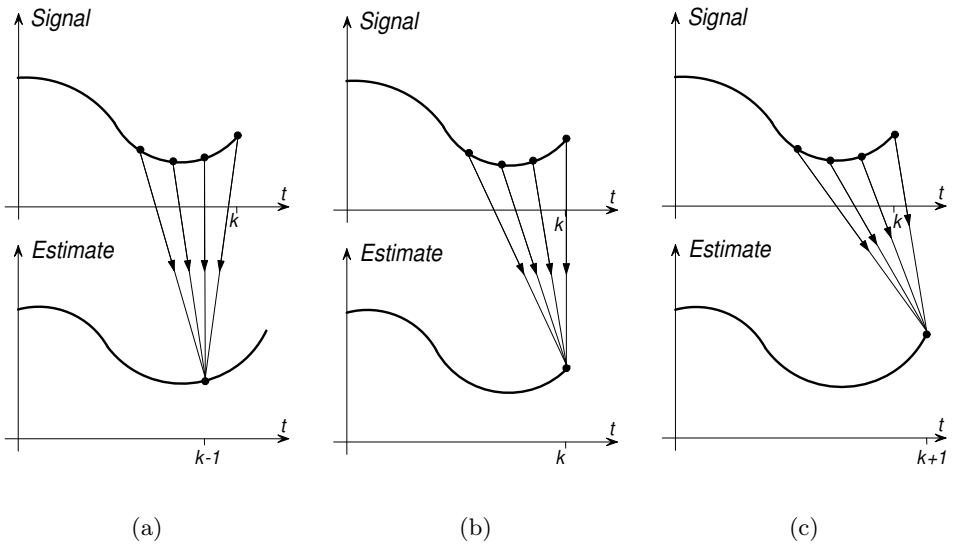
Figure 8: The omnibus model (from [7])

While the measurements have to be recorded in real time, the smoothing algorithm can be performed offline.

Filtering ($m = 0$): The actual state of a process entity shall be estimated by using an actual measurement and information gained from previous measurements. Usually, filtering is performed in real time.

Prediction ($m > 0$): The actual state of a process entity shall be estimated by using a history of previous measurements. The prediction problem requires an adequate system model in order to produce a meaningful estimation. Typically, prediction is performed in real time.

Figure 9 illustrates the different cases. Many filtering algorithms cover all three aspects. Filtering and prediction are fundamental elements of any tracking



(a) (b) (c)

Figure 9: Smoothing (a), filtering (b), and prediction (c)

system. They are used to estimate present and future kinematic quantities such as position, velocity, and acceleration [60].

4.2 Kalman Filtering

The stochastic *Kalman Filter* uses a mathematical model for filtering signals using measurements with a respectable amount of statistical and systematical errors. The method was developed by Kalman and Bucy in 1960 [39, 40].

Generally, a Kalman Filter fuses data measured in successive time intervals providing a maximum likelihood estimate of a parameter. It is also possible to relate inputs from multiple sensors to a vector of internal states containing the parameters of interest as long as there are only linear dependencies between inputs and system states [66].

The filter uses a discrete-time algorithm to remove noise from sensor signals in order to produce fused data that, for example, estimate the smoothed values of position, velocity, and acceleration at a series of points in a trajectory.

The standard Kalman Filter model is described by two linear equations. The first equation describes the dynamics of the system:

$$\vec{x}_{k+1} = \mathrm{A} \cdot \vec{x}_k + \mathrm{B} \cdot \vec{u}_k + w \tag{1}$$

where $\vec{x}_k$ is a vector that contains the system state at time k, A is the non-singular state transition matrix. Vector $\vec{u}_k$ describes the input to the system at time k. The relation between the input vector $\vec{u}_k$ and the system state vector $\vec{x}_{k+1}$ is defined by matrix B. w is a random variable that stands for the system noise, modelled as white noise $\sim N(0, \mathrm{Q})$, where Q is the covariance matrix.

The second linear equation describes the noisy observations of the system:

$$\vec{y}_k = \mathrm{C} \cdot \vec{x}_k + v \tag{2}$$

where each element of vector $\vec{y}_k$ contains a sensor observation at time k, the matrix C relates the measurements to the internal state, and v is the measurement noise, also modelled as white noise $\sim N(0, \mathrm{R})$ with the covariance matrix R.

The model described by equations 1 and 2 represents a very general model. For example, if one uses the identity matrix as state transition matrix ($\mathrm{A} \equiv I$) and sets the input to zero ($\vec{u} \equiv \vec{0}$), the model describes the standard sensor fusion case, where some internal state of a system can be reconstructed using subsequent more or less distorted measurements [36]. Another special case is given if the desired states can be measured directly. Hence, C is equivalent to the identity matrix or a permutation matrix.

The Kalman Filter is applied by doing the following steps: First an a priori estimator $\hat{\vec{x}}_{k+1|k}$ of the system state $\vec{x}$ for time $k+1$ is computed using the best system estimation at time k (equation 1):

$$\hat{\vec{x}}_{k+1|k} = \mathrm{A} \cdot \hat{\vec{x}}_{k|k} + \mathrm{B} \cdot \vec{u}_k \tag{3}$$

Then we compute the predicted error covariance matrix P for instant $k+1$:

$$\mathrm{P}_{k+1|k} = \mathrm{A} \cdot \mathrm{P}_{k|k} \cdot \mathrm{A}^T + \mathrm{Q} \tag{4}$$

and the Kalman gain matrix K:

$$\mathrm{K}_{k+1} = \frac{\mathrm{P}_{k+1|k} \cdot \mathrm{C}^T}{\mathrm{C} \cdot \mathrm{P}_{k+1|k} \cdot \mathrm{C}^T + \mathrm{R}} \tag{5}$$

Now the estimator $\hat{\vec{x}}$ can be updated by a new process observation $\vec{y}_{k+1}$:

$$\hat{\vec{x}}_{k+1|k+1} = \hat{\vec{x}}_{k+1|k} + \mathrm{K}_{k+1} \cdot (\vec{y}_{k+1} - \mathrm{C} \cdot \hat{\vec{x}}_{k+1|k}) \tag{6}$$

and the new error covariance matrix $P_{k+1|k+1}$ is derived by

$$\mathrm{P}_{k+1|k+1} = (\mathrm{I} - \mathrm{K}_{k+1} \cdot \mathrm{C})\mathrm{P}_{k+1|k}(\mathrm{I} - \mathrm{K}_{k+1} \cdot \mathrm{C})^T + \mathrm{K}_{k+1} \cdot \mathrm{R} \cdot \mathrm{K}_{k+1}^T \tag{7}$$

where I is the identity matrix. After calculation of equation 7 the iteration restarts with equation 3 and $k := k + 1$.

For start-up initialization of the Kalman Filter, one has to provide an estimate $\hat{\vec{x}}_0$ and an estimate of its covariance $P_{0|0}$. $P_{0|0}$ can be initialized with an estimated inaccurate start value, since the subsequent application of the Kalman Filter will let P approach its exact value.

Since each iteration has approximately the same effort, the Kalman Filter is well-suited for real-time applications. The first field of application was aerospace engineering. It was used, for example, in the Ranger, Mariner, and Apollo missions of the 1960s. Today, Honeywell's fault-tolerant gyro system on the Boeing 777 uses a Kalman Filter [17]. Kalman Filters have often been used in the field of robotics. For instance, Wen and Durrant-Whyte applied a Kalman Filter for a robot arm that uses sonar and infrared sensors to locate a specific object [76]. In mobile robotics, Kalman Filters are used for correction of localizations based on two or more different types of input [16, 66, 29]. Furthermore, the Kalman Filter has been applied in control theory and image processing.

Although the above described standard Kalman Filter performs well for many applications, the standard Kalman Filter approach had to be extended for several applications in order to achieve satisfying results [77]. Following, some extensions to the Kalman Filter are listed:

Non-linear systems: Linear modelling of the system is not always feasible. Although in many cases, linear behavior around some operating point can be assumed, there are still problems that cannot be described accurately by a linear model. Therefore, the Extended Kalman Filter (EKF) has been derived that uses non-linear stochastic difference equations for the system model [75]. However, while the standard Kalman Filter is *optimal* for linear systems, the solution provided by the EKF is only approximate. A major shortcoming of the EKF is that the distributions of the random variables are no longer normal after undergoing their nonlinear transformations. Attacking this problem, Julier et al. published a variant of EKF that uses a different parametrization of mean and covariance values that provides more accurate results than the original EKF [38].

Estimating system parameters: The statistical parameters are not always known *a priori* or constant over time. Hence, in this case a sophisticated version of the Kalman Filter also has to estimate the statistical parameters. Åström and Wittenmark describe an approach with time-varying matrices in [3].

Alternatives to least mean square optimization: The Kalman Filter approach minimizes the error using a least mean square approach. Depending on the application, other criteria, such as the H_∞ norm [62], perform better.

Reducing computational load: With respect to an implementation in embedded microcontrollers, the computational effort is of great interest.

Kalman filtering requires matrix multiplication and matrix inversion. If there are no dedicated vector processing units available on the microprocessor, it is important to find efficient implementations in order to achieve adequate performance and timely results. Gan and Harris showed that in a multi-sensor environment with identical measurement matrices the measurements can be handled separately in order to get computations of lower complexity [32]. In case of differing measurement matrices, using a merged input vector containing the information of all sensors is preferable. Generally, large systems can be modelled with the *information form* of the Kalman Filter. This variant is functionally identical, but has computational advantages for high-dimensional data [32].

4.3 Inference Methods

Merriam Webster's college dictionary defines inference as *the act of passing from one proposition, statement, or judgment considered as true to another whose truth is believed to follow from that of the former.* Inference methods are used for *decision fusion*, i. e., to take a decision based on given knowledge. A possible application could be the decision if the road in front of a car is blocked or free, given measurements of multiple sensors.

Classical inference methods perform tests on an assumed hypothesis versus an alternative hypothesis. As a test of significance, it yields the probability that the actually observed data would be present, if the chosen hypothesis were true. However, classical inference does not support the usage of *a priori* information about the likelihood of a proposed hypothesis [34]. This *a priori* probability is taken into account when using Bayesian inference, which is named after the English clergyman Thomas Bayes. In a paper published after his death in the Philosophical Transactions of the Royal Society of London [6] Bayes stated the rule known today as Bayes' theorem:

$$\mathbb{P}(H|E) = \frac{\mathbb{P}(E|H)\,\mathbb{P}\,(H)}{\mathbb{P}(E)} \tag{8}$$

Bayes' theorem quantifies the probability of hypothesis H, given that an event E has occurred. $\mathbb{P}(H)$ is the *a priori* probability of hypothesis H, $\mathbb{P}(H|E)$ states the *a posteriori* probability of hypothesis H. $\mathbb{P}(E|H)$ is the probability that event E is observed given that H is true.

Given multiple hypotheses, Bayesian inference can be used for classification problems. Then, Bayes' rule produces a probability for each hypotheses H_i. Each H_i corresponds to a particular class:

$$\mathbb{P}(H_i|E) = \frac{\mathbb{P}(E|H_i)\,\mathbb{P}\,(H_i)}{\Sigma_i\,\mathbb{P}\,(E|H_i)\,\mathbb{P}\,(H_i)} \tag{9}$$

Examples for applications based on Bayesian inference can be found in [24] for merging multiple sensor readings, in automatic classification of sensor inputs (e. g., the computer program AUTOCLASS [14] developed by the NASA), or in map building for mobile robots [25].

However, when Bayesian inference is used for sensor fusion, certain drawbacks can emerge [12]: One problem is the required knowledge of the *a priori* probabilities $\mathbb{P}(E|H_i)$ and $\mathbb{P}(E)$, which may not always be available [8]. Thus, it is often necessary to make subjective judgements on some of the probabilities. Although it is possible to use subjective probabilities [34], Bayesian inference requires that all probabilities are at the same level of abstraction. Another problem arrives when different sensors return conflicting data about the environment [9].

Therefore, Dempster developed the mathematical foundations for a generalization of Bayesian theory of subjective probability [22]. The result was Dempster's rule of combination, which operates on belief or mass functions like Bayes' rule does on probabilities. Shafer, a student of Dempster, extended Dempster's work and developed a Mathematical Theory of Evidence [61]. This theory can be applied for representation of incomplete knowledge, belief updating, and for combination of evidence [56]. The choice between Bayesian and Dempster-Shafer inference methods for decision algorithms is non-trivial and has been subject to heated debates over the last several years [13, 21].

4.4 Occupancy Maps

An occupancy map is a usually two-dimensional raster image uniformly distributed over the robot's working space. Each map pixel contains a binary value indicating whether the according space is free or occupied by an obstacle. There are two main perceptual paradigms for occupancy map algorithms [35]:

Image → objects: In the image → objects perceptual paradigm, a predefined empty area is assumed in which the observation of objects by the sensors populate the occupancy map [2]. Object tracking algorithms, for example triangulation [57], are used to obtain the positions of the objects.

Image → free space: In the image → free space perceptual paradigm, one assumes a predefined occupied area in which the observation of sensors creates free space in the occupancy map. Hoover and Olsen presented an application where a set of video cameras are used to detect free space in the vicinity of a robot [35]. They use multiple viewing angles to overcome the problem of occlusion and to increase performance. However, their application depends on correct measurements from all sensors.

Although the first paradigm of object tracking is the more common approach, the image → free space approach has some advantages over the image → objects approach. Hoover lists reduced complexity, (no tracking is necessary to create the map), robustness, and safety (a sensor breakdown causes a loss of perception of the free space, not on the objects) as advantages [35]. However, the occupancy map approach, as described here, suffers from a major problem regardless of the used perception paradigm: map pixels, for which either none or multiple contradicting measurements are given, can be misinterpreted since the pixels of an occupancy map cannot reflect uncertainty. This problem can be

overcome by extending an occupancy map with additional information about the reliance of a pixel value – this approach is known as *certainty grid*.

4.5 Certainty Grid

A *certainty grid* is a special form of an occupancy map. Likewise, it is a multi-dimensional (typically two- or three-dimensional) representation of the robot's environment. The observed space is subdivided into square or cube cells like in the occupancy map, but each cell now contains a value reflecting a measure of probability that an object exists within the related cell [78].

The information of a cell can be "free" if the corresponding space appears to be void; or "occupied" if an object has been detected at that place. Cells not reached by sensors are in an "uncertain" state. All values of the grid are initially set to this uncertain state. Each sensor measurement creates either free space or objects in the grid. Thus, the certainty grid approach is a hybrid approach between the image → objects and the image → free space perceptual paradigms of the occupancy grid. Basically, it is assumed that the certainty grid application has no *a priori* knowledge on the geometry of its environment and that objects in this environment are mostly static. The effect of occasional sensor errors can be neglected, because they have little effect on the grid [46].

In general, sensor information is imperfect with respect to restricted temporal and spatial coverage, limited precision, and possible sensor malfunctions or ambiguous measurements. To maximize the capabilities and performance it is often necessary to use a variety of sensor devices that complement each other. Mapping such sensor measurements into the grid is an estimation problem [26].

Matthies and Elfes [48] propose a uniform method for integrating various sensor types. Each sensor is assigned a spatial interpretation model, which is developed for each kind of sensor, that maps the sensor measurement into corresponding cells. When sensor uncertainties are taken into account, we arrive at a probabilistic sensor model.

The calculation of new grid values is usually done by Bayesian inference. Suppose that two sensors S_1 and S_2 give two occupancy probability values for a particular grid element *cell*. Using Bayes' rule, the updated probability of the cell being occupied by an obstacle can be calculated as:

$$\mathbb{P}(cell.occ|S_1, S_2) = \frac{\mathbb{P}(S_1|cell.occ, S_2)\,\mathbb{P}\,(cell.occ|S_2)}{\mathbb{P}(S_1|cell.occ, S_2)\,\mathbb{P}\,(cell.occ|S_2) + \mathbb{P}(S_1|cell.emp, S_2)\,\mathbb{P}\,(cell.emp|S_2)} \tag{10}$$

where *cell.occ* and *cell.emp* are the probabilities of the cell being occupied or empty, respectively. *Conditional independence* for the two sensors is defined in the following relation:

$$\mathbb{P}(S_1|cell.occ, S_2) = \mathbb{P}(S_1|cell.occ) \tag{11}$$

Furthermore, we assume $\mathbb{P}(cell.occ) = 1 - \mathbb{P}(cell.emp)$. Assuming, that the

prior probability had been equal to 0.5 (*maximum entropy assumption* [50]), the fusion formula can be expressed by:

$$\mathbb{P}(cell.occ|S_1, S_2) = \frac{\mathbb{P}(cell.occ|S_1)\,\mathbb{P}\,(cell.occ|S_2)}{\mathbb{P}(cell.occ|S_1)\,\mathbb{P}\,(cell.occ|S_2) + (1 - \mathbb{P}(cell.occ|S_1))(1 - \mathbb{P}(cell.occ|S_2))} \tag{12}$$

Equation 12 can be extended to the case of several sensors $S_1, S_2, \ldots, S_n$ by induction. Hence, the fusion formula for n sensors can be expressed as follows:

$$\frac{1}{\mathbb{P}(cell.occ|S_1, \ldots, S_n)} - 1 = \prod_{i=1}^{n} \left(\frac{1}{\mathbb{P}(cell.occ|S_i)} - 1 \right) \tag{13}$$

Equations 12 and 13 show fully associative and commutative behavior. Thus, the order of processing does not influence the result.

4.6 Reliable Abstract Sensors

The term "reliable abstract sensors" has been coined by Marzullo [47], who has elaborated a method for fault tolerance in multisensor environments. The main idea in Marzullo's work is a geometric derivation for *fault masking*. He defines sensors as piecewise continuous functions characterized by two parameters: *shape* and *accuracy range*. Shape defines the form taken by the uncertainty around the measurement value returned by the sensor. Since Marzullo addresses only the one-dimensional case, thus assuming that all shapes are linear. The accuracy range defines the interval that contains the true value of the measured entity. By combining measurements from single sensors, an improved abstract sensor can be built. The range of this abstract sensor is derived by finding the intersections of the ranges of the single sensors.

Fault-tolerant sensor averaging [47] is introduced by regarding at most t sensors to be faulty.[3] Faulty sensors deliver an improper interval that may not contain the true value. Therefore, a *fault-tolerant* sensor averaging algorithm returns an interval that contains the true value for sure, even if arbitrary t out of $2t + 1$ readings are faulty. Marzullo presents an algorithm for such a *reliable abstract sensor* in [47, page 290]:

> *Let $\mathcal{I}$ be a set of values taken from n abstract sensors, and suppose the abstract sensors are of the same physical state variable where their values were read at the same instant. Assuming that at most t of these sensors are incorrect, calculate $\bigcap_{t,n}(\mathcal{I})$ which is the smallest interval that is guaranteed to contain the correct physical value.*
> *Implementation: Let l be the smallest value contained in at least*

[3] "t" is the preferred letter in the literature for the number of faults to be tolerated. It derives from the concept of a *traitor* in the Byzantine Generals scenario.

$n-t$ of the intervals in $\mathcal{I}$ and h be the largest value contained in at least $n-t$ of the intervals in $\mathcal{I}$ then $\bigcap_{t,n}(\mathcal{I})$ will be the interval $[l...h]$.

Marzullo's original work has been extended to sensors working with any number of dimensions by Chew [15]. While in the linear or single-dimensional case the abstract sensor will always deliver correct results for at most t faulty sensors out of $2t+1$ sensors, in the multi-dimensional case there must be at least $2tD+1$ sensors in order to tolerate t faulty sensors, where D is the number of dimensions.

Jayasimha [37] has extended Marzullo's work with a better detection of faulty sensor's for the linear case. Both, Marzullo's and Jayasimha's algorithm have a run time complexity of $O(n \log n)$.

5 Summary

Sensor fusion offers a great opportunity to overcome physical limitations of sensing systems. An important point will be the reduction of software complexity, in order to hide the properties of the physical sensors behind a sensor fusion layer.

This paper contained a survey on sensor fusion architectures and methods. There is no common model of sensor fusion until today, however there are a host of models that propose similar partitioning into source preprocessing, feature extraction and pattern processing, situation assessment and decision making.

Besides the JDL model, the waterfall model will be an interesting alternative for embedded real-time fusion systems since it emphasizes the low-levels of fusion processing. However, like the JDL model, the waterfall model has its drawbacks, i. e. it lacks of guiding the developer in selecting the appropriate methods and concretely designing an implementation. Moreover, many typical fusion applications will have a closed control loop, thus the expression of a cyclic data processing will be advantageous as implemented in the intelligence cycle, the Boyd loop and the Omnibus model.

The section on fusion methods is incomplete – as a survey on fusion methods will always be. However, currently the Kalman Filter and Bayesian reasoning are the most frequently used tools in sensor fusion. The certainty grid for robotic vision is a nice example for a combination of complementary and competitive fusion based on Bayes' rule. Beyond the presented methods, techniques based on soft decisions and Fuzzy logic will certainly provide a good means for particular sensor fusion problems.

References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *International Journal of Robotics Research*, 17(4):315–337, April 1998.

[2] C. S. Andersen, C. B. Madsen, J. J. Sørensen, N. O. S. Kirkeby, J. P. Jones, and H. I. Christensen. Navigation using range images on a mobile robot. *Robotics and Autonomous Systems*, 10:147–160, 1992.

[3] K. J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design.* Prentice-Hall International Editions, Englewood Cliffs, NJ, USA, 1984.

[4] D. E. Bakken, Z. Zhan, C. C. Jones, and D. A. Karr. Middleware support for voting and data fusion. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 453–462, Gothenburg, Sweden, 2001.

[5] T. Bass. Intrusion detection systems and multisensor data fusion: Creating cyberspace situational awareness. *Communications of the ACM*, 43(4):99–105, May 2000.

[6] T. Bayes. Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763. Reprinted in *Biometrika*, 45:293–315, 1958.

[7] M. D. Bedworth and J. O'Brien. The omnibus model: A new architecture for data fusion? In *Proceedings of the 2nd International Conference on Information Fusion (FUSION'99)*, Helsinki, Finnland, July 1999.

[8] S. S. Blackman. *Introduction to Sensor Systems*, chapter Multiple Sensor Tracking and Data Fusion. Artech House, Norwood, Massachusetts, 1988.

[9] P. L. Bogler. Shafer-dempster reasoning with applications to multisensor target identification systems. *IEEE Transactions on Systems, Man and Cybernetics*, 17(6):968–977, November–December 1987.

[10] E. Bosse, J. Roy, and D. Grenier. Data fusion concepts applied to a suite of dissimilar sensors. *Canadian Conference on Electrical and Computer Engineering, 1996*, 2:692–695, May 1996.

[11] J. R. Boyd. A discourse on winning and losing. Unpublished set of briefing slides, Air University Library, Maxwell AFB, AL, USA, May 1987.

[12] R. R. Brooks and S. S. Iyengar. *Multi-Sensor Fusion: Fundamentals and Applications.* Prentice Hall, New Jersey, 1998.

[13] D. M. Buede. Shafer-dempster and bayesian reasoning: A response to 'Shafer-Dempster Reasoning with Applications to Multisensor Target Identification Systems'. *IEEE Transactions on Systems, Man and Cybernetics*, 18(6):1009 –1011, November–December 1988.

[14] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining.* AAAI Press/MIT Press, 1996.

[15] P. Chew and K. Marzullo. Masking failures of multidimensional sensors. In *Proceedings of the 10th Symposium on Reliable Distributed Systems*, pages 32–41, Pisa, Italy, October 1991.

[16] H. Chung, L. Ojeda, and J. Borenstein. Sensor fusion for mobile robot dead-reckoning with a precision-calibrated fiber optic gyroscope. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3588–3593, Seoul, Korea, May 2001.

[17] B. Cipra. Engineers look to kalman filtering for guidance. *SIAM News*, 26(5), August 1993.

[18] B. V. Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85:24–38, January 1997.

[19] B. V. Dasarathy. More the merrier ... or is it? - sensor suite augmentation benefits assessment. In *Proceedings of the 3rd International Conference on Information Fusion*, volume 2, pages 20–25, Paris, France, July 2000.

[20] B. V. Dasarathy. Information fusion - what, where, why, when, and how? *Information Fusion*, 2(2):75–76, 2001. Editorial.

[21] F. Daum. Book review on: Handbook of multisensor data fusion. *IEEE Aerospace and Electronic Systems Magazine*, 16(10):15–16, October 2001.

[22] A. P. Dempster. Upper and lower propabilities induced by a multi-valued mapping. *Annual Mathematical Statistics*, 38:325–339, 1967.

[23] H. F. Durrant-Whyte. Sensor models and multisensor integration. *International Journal of Robotics Research*, 7(6):97–113, December 1988.

[24] H. F. Durrant-Whyte. Toward a fully decentralized architecture for multi-sensor data fusion. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1331–1336, Cincinnati, OH, USA, 1990.

[25] A. Elfes. A sonar-based mapping and navigation system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, 1986.

[26] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46–57, 1989.

[27] W. Elmenreich and P. Peti. Achieving dependability in a time-triggered network by sensor fusion. In *Proceedings of the 6th IEEE International Conference on Intelligent Engineering Systems (INES)*, pages 167–172, Opatija, Croatia, May 2002.

[28] W. Elmenreich and S. Pitzek. Using sensor fusion in a time-triggered network. In *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 369–374, Denver, CO, USA, November–December 2001.

[29] E. Fabrizi, G. Oriolo, S. Panzieri, and G. Ulivi. Mobile robot localization via fusion of ultrasonic and inertial sensor data. In *Proceedings of the 8th International Symposium on Robotics with Applications*, Maui, USA, 2000.

[30] K. E. Foote and D. J. Huebner. Error, accuracy, and precision. Technical report, The Geographer's Craft Project, Department of Geography, University of Texas at Austin, 1995.

[31] C. A. Fowler. Comments on the cost and performance of military systems. *IEEE Transactions on Aerospace and Electronic Systems*, 15:2–10, January 1979.

[32] Q. Gan and C. J. Harris. Comparison of two measurement fusion methods for kalman-filter-based multisensor data fusion. *IEEE Transactions on Aerospace and Electronics*, 37(1):273–279, January 2001.

[33] P. Grossmann. Multisensor data fusion. *The GEC journal of Technology*, 15:27–37, 1998.

[34] D. L. Hall. *Mathematical Techniques in Multi-Sensor Data Fusion.* Artech House, Norwood, Massachusetts, 1992.

[35] A. Hoover and B. D. Olsen. Sensor network perception for mobile robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA*, pages 342–347, April 2000.

[36] H. Hyötyniemi. *Multimedia Applications in Industrial Automation – Collected Papers of the Spring 1997 Postgraduate Seminar*, chapter Modeling of High-Dimensional Data, pages 114–138. Helsinki University of Technology, Control Engineering Laboratory, 1997.

[37] D. N. Jayasimha. Fault tolerance in a multisensor environment. *Proceedings of the 13th Symposium on Reliable Distributed Systems*, pages 2–11, 1994.

[38] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of the 1995 American Control Conference*, pages 1628–1632, Seattle, WA, USA, 1995.

[39] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME, Series D, Journal of Basic Engineering*, 82:35–45, March 1960.

[40] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Transaction of the ASME, Series D, Journal of Basic Engineering*, 83:95–108, March 1961.

[41] M. Kam, X. Zhu, and P. Kalata. Sensor fusion for mobile robot navigation. *Proceedings of the IEEE*, 85(1):108–119, Jan. 1997.

[42] H. Kopetz, H. Kantz, G. Grünsteidl, P. Puschner, and J. Reisinger. Tolerating transient faults in MARS. In *Proccedings of the 20th. Symposium on Fault Tolerant Computing, Newcastle upon Tyne, UK*, June 1990.

[43] J. Llinas and D. L. Hall. An introduction to multi-sensor data fusion. *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, 6:537–540, May–June 1998.

[44] R. C. Luo and M. Kay. Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):901–930, September–October 1989.

[45] M. Markin, C. Harris, M. Bernhardt, J. Austin, M. Bedworth, P. Greenway, R. Johnston, A. Little, and D. Lowe. Technology foresight on data fusion and data processing. Publication, The Royal Aeronautical Society, 1997.

[46] M. C. Martin and H. P. Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, The Robotics Institute, Carneghie Mellon University, Pittsburgh, PA, USA, 1996.

[47] K. Marzullo. Tolerating failures of continuous-valued sensors. *ACM Transactions on Computer Systems*, 8(4):284–304, November 1990.

[48] L. Matthies and A. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 727–733, Philadelphia, PA, USA, 1988.

[49] G. T. McKee. What can be fused? *Multisensor Fusion for Computer Vision, Nato Advanced Studies Institute Series F*, 99:71–84, 1993.

[50] B. Moshiri, M. R. Asharif, and R. Hosein Nezhad. Pseudo information measure: A new concept for extension of Bayesian fusion in robotic map building. *Information Fusion*, 3(1):51–68, 2002.

[51] R. R. Murphy. Biological and cognitive foundations of intelligent sensor fusion. *IEEE Transactions on Systems, Man and Cybernetics*, 26(1):42–51, January 1996.

[52] P. J. Nahin and J. L. Pokoski. NCTR plus sensor fusion equals IFFN or can two plus two equal five? *IEEE Transactions on Aerospace and Electronic Systems*, 16(3):320–337, May 1980.

[53] V. P. Nelson. Fault-tolerant computing: Fundamental concepts. *IEEE Computer*, 23(7):19–25, July 1990.

[54] B. Parhami. A data-driven dependability assurance scheme with applications to data and design diversity. In A. Avizienis and J. C. Laprie, editors, *Dependable Computing for Critical Applications*, volume 4, pages 257–282. Springer Verlag, Vienna, 1991.

[55] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2), 1980.

[56] G. M. Provan. The validity of dempster-shafer belief functions. *International Journal of Approximate Reasoning*, 6:389–399, 1992.

[57] B. Y. S. Rao, H. F. Durrant-Whyte, and J. A. Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *International Journal of Robotics Research*, 12(1):20–44, 1993.

[58] N. S. V. Rao. A fusion method that performs better than best sensor. *Proceedings of the First International Conference on Multisource-Multisensor Information Fusion*, pages 19–26, July 1998.

[59] P. L. Rothman and R. V. Denton. Fusion or confusion: Knowledge or nonsense? *SPIE Data Structures and Target Classification*, 1470:2–12, 1991.

[60] V. V. S. Sarma and S. Raju. Multisensor data fusion and decision support for airborne target identification. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1224–1230, September–October 1991.

[61] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976.

[62] U. Shaked and Y. Theodor. H_∞-optimal estimation: A tutorial. In *Proceedings of the 31st IEEE Conference on Decision and Control*, volume 2, pages 2278–2286, Tucson, Arizona, USA, 1992.

[63] A. Singhal. Issues in autonomous mobile robot navigation. Survey paper towards partial fulfillment of MS degree requirements, Computer Science Department, University of Rochester, Rochester, NY 14627-0226, May 1997.

[64] T. Smestad. Data fusion – for humans, computers or both? Translated article from *Mikroskopet*, Norwegian Defence Research Establishment, February 2001.

[65] A. N. Steinberg, C. L. Bowman, and F. E. White. Revisions to the JDL data fusion model. In *Proceedings of the 1999 IRIS Unclassified National Sensor and Data Fusion Conference (NSSDF)*, May 1999.

[66] C. Tarín, H. Brugger, R. Moscardó, B. Tibken, and E. P. Hofer. Low level sensor fusion for autonomous mobile robot navigation. In *Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference (IMTC'99)*, volume 3, pages 1377–1382, 1999.

[67] R. R. Tenney and N. R. Sandell jr. Detection with distributed sensors. *IEEE Transactions on Aerospace and Electronic Systems*, 17(4):501–510, July 1981.

[68] A. Theil, L. J. H. M. Kester, and É. Bossé. On measures of performance to assess sensor fusion effectiveness. In *Proceedings of the 3rd International Conference on Information Fusion, Paris, France*, July 2000.

[69] U. S. Department of Defense (DoD), Data Fusion Subpanel of the Joint Directors of Laboratories, Technical Panel for C3. *Data Fusion Lexicon*, 1991.

[70] A. Visser and F. C. A. Groen. Organisation and design of autonomous systems. Textbook, Faculty of Mathematics, Computer Science, Physics and Astronomy, University of Amsterdam, Kruislaan 403, NL-1098 SJ Amsterdam, August 1999.

[71] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956.

[72] L. Wald. A european proposal for terms of reference in data fusion. *International Archives of Photogrammetry and Remote Sensing*, XXXII, Part 7:651–654, 1998.

[73] E. Waltz. The principles and practice of image and spatial data fusion. In *Proceedings of the 8th National Data Fusion Conference, Dallas*, 1995.

[74] E. Waltz and J. Llinas. *Multisensor Data Fusion*. Artech House, Norwood, Massachusetts, 1990.

[75] G. Welch and G. Bishop. An introduction to the kalman filter. In *SIGGRAPH 2001 Conference Proceedings*, 2001. Tutorial 8.

[76] W. Wen and H. F. Durrant-Whyte. Model-based multi-sensor data fusion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1720–1726, Nice, France, 1992.

[77] L. Wenzel. Kalman-filter. *Elektronik*, (6,8,11,13), 2000.

[78] L. Yenilmez and H. Temeltas. Real time multi-sensor fusion and navigation for mobile robots. *9th Mediterranean Electrotechnical Conference*, 1:221–225, May 1998.