Research Report 13/2001

Proceedings of the 5th IEEE International Conference on Intelligent Engineering Systems
Helsinki, September 2001

The Time-Triggered Sensor Fusion Model

W. Elmenreich and S. Pitzek

Institut für Technische Informatik
TU Vienna, Austria
Tel: +43 (1) 58801-18234
wil@vmars.tuwien.ac.at

September 14, 2001

Abstract

This paper compares current sensor fusion models and proposes a new time-triggered model for sensor fusion.

The time-triggered sensor fusion model decomposes a real time system into three levels, a node level, containing the sensors and the actuators, a cluster level that gathers measurements and performs sensor fusion, and an application level where an application program makes control decisions based on environmental information provided by the cluster level. Because the application code is independent of the employed sensors, the system is open to sensor reconfigurations and reuse of the application code. Furthermore, the model contains a hardware-independent application interface and a time-triggered smart transducer network.

An application of the presented concept is shown with a mobile robot controlled by a time-triggered communication network.

Keywords: *sensor fusion, modelling, time-triggered systems, architecture, interface.*

1 Introduction

Many real-time systems reach decisions based on sensor measurements on their environment. Unfortunately, sensors, in their current form, are not reliable interfaces for several reasons [BI98, Elm00]. Methods for enhancing sensor properties have emerged in the past decades in form of sensor replication and sensor fusion. Intelligent sensor fusion aids in the development of systems that "see" and "comprehend" their environment using computational methods to get a picture from the data collected by sensors. Much research on sensor-fusion and according models originated in the military domain.

Decomposing a system into subsystems with a high degree of inner connectivity, and defining adequate interfaces between these subsystems is a well-known design principle [Krü97]. Finding an adequate model for a real-time sensor-fusion application that allows such a decomposition can be a difficult task, because many existing fusion models rely heavily on military applications and, thus, incorporate a structure that does not map well to civil applications.

The objective of this paper is to provide a generic architectural model of a time-triggered sensor fusion system. The aim of this model is to provide a framework for the decomposition of real-time applications. Such a decomposition eases implementation, reconfiguration, code reuse, testing, and debugging.

The rest of the paper is organized as follows: Section 2 describes existing models related to sensor fusion and real-time systems. Section 3 explains the time-triggered sensor fusion model with its computational stages and its communication interfaces. Section 4 describes a case study based on the time-triggered sensor fusion model. The paper is concluded in Section 5.

2 Related Work

Fusion Models

Due to the fact that data fusion models heavily depend on the application, no generally accepted model of data fusion exists until today [BO00].

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). This *JDL model* [WL90] consists of five levels of fusion processing and a database, which are all interconnected by a bus. The five levels (preprocessing, single object refinement, situation refinement, implication refinement, and process refinement) are not meant to be processed in a strict order. The JDL model is very popular for data fusion systems, but is not always appropriate for other than defense data fusion systems [BO00].

Another approach to model a fusion application is to line out its cyclic character. Representatives of such an approach are the intelligence cycle [Shu91] and the Boyd control loop [Boy87].

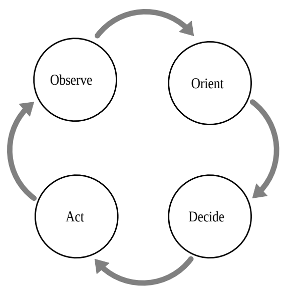

Figure 1: The Boyd (or OODA) Loop

The *Intelligence Cycle* [Shu91] comprises the following stages: (i) Planning and Direction determinate intelligence requirements, (ii) Collection performs a gathering of appropriate information, (iii) Collation lines up the collected information, (iv) the Evaluation fuses and analyzes information, and (v) Dissemination distributes the fused intelligence.

John Boyd has proposed a cycle of observation–orientation–decision–action [Boy87]. The *Boyd control cycle* or *OODA loop* (see Figure 1) represents the classic decision-support mechanism in military information operations. Because decision-support systems for situational awareness are tightly coupled with data fusion systems [Bas00], the Boyd loop has also been used for data fusion.

The Time-Triggered Computation Model and Client-Server Architectures

The time-triggered model of computation [Kop98] is a model for the representation and analysis of real-time systems. It is intended for the representation and analysis of the design of large hard real-time systems. The model consists of four building blocks: (i) interfaces that contain temporally accurate data, (ii) a communication subsystem that connects interfaces, (iii) a host computer that reads input data from interfaces and writes output data to interfaces, and (iv) a transducer that transforms the information representation in the environment into the digital form of the interface and vice versa. The main idea is a compositional design of a real-time system, supported by well-defined interfaces to separate the design of the interaction pattern among components from the design of the components themselves. All instants of communication are defined *a priori* in a configuration phase. Sporadic requests are converted into periodic time-triggered messages. Thus, a time-triggered scheduling is designed for the worst-case scenario and will always hold its deadlines, for the cost of high average communication load.

The time-triggered model of computation stands in contrast to classical client-server architectures. In a client-server model services are requested by a client from a server. Communications take place in a request-response pattern. A typical client server architecture is CORBA, that introduces client-server interfaces to interconnect heterogeneous computer systems [Vin97].

3 The Time-Triggered Sensor Fusion Model

Because none of the existing fusion models was adequate for modelling a real-time system with data acquisition, fusion processing, and control decisions, the concept of a time-triggered sensor fusion model has been developed.

The Time-Triggered System Model incorporates properties of the approaches described in Section 2, like a cyclic processing and a composable design introducing well-defined interfaces between its subsystems.

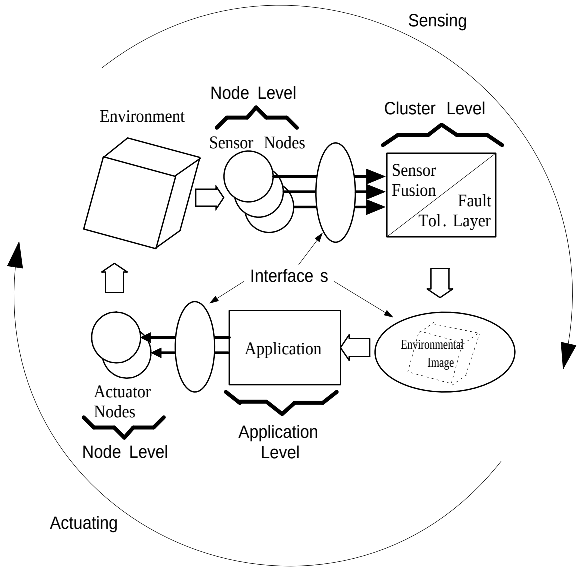

Figure 2: The Time-Triggered Sensor Fusion Model

The model, as depicted in Figure 2 identifies three levels of data processing with well-defined interfaces between them.

Three-Level Abstraction

The sensors and actuators that interface the environment refer to the *node level* of the system. Each sensor can be seen as a window showing a small part of the environment. It is the task of the node level to provide each window view. The node level physically interacts with the environment by performing measurements with sensors or actions with actuators. To support maximum modularity, the nodes are build as smart transducers. The smart transducer technology offers a number of advantages from the point of view of technology, cost, and complexity management [Kop99, KHE00].

The *cluster level* contains the hardware and software that act as a glue between transducers and control program. It integrates the measurements from the sensors into a unified view, the environment image. If necessary, this image is made independent of incomplete, erroneous or missing measurements by implementing sensor fusion algorithms and a fault tolerance layer.

The *control application level* contains the control intelligence to make decisions based on the environment image. The environment image may be any feasible abstraction of the environmental properties of interest. The properties of the environment image like content, resolution, update timing, and data structure depend on the application type. The control application influences the environment via actuators and closes the control loop.

Interfaces

The breakdown into these three levels is justified by different tasks they have to fulfill and the different knowledge necessary for implementing the corresponding software (see Table 1). To support composability the model uses well-defined interfaces between the levels:

Smart Transducer Interface: This interface connects the transducers (sensors or actuators) with the cluster level. The

Level	Task	Implementer	Knowledge
Node	Provide Transducer Information	Transducer Manufacturer	Internals of the Node
Cluster	Gather and represent Sensor Information	System Integrator	Sensor Fusion Algorithms, Fault Tolerant Concepts
Control Application	Control System	Application Programmer	Control/Navigation, Decision finding

Table 1: Properties of Node, Cluster, and Control Application Level

smart transducer interface exports the transducer properties in a unified manner and hides the node's internals. This interface appears twice in the model, as a sensor and as an actuator interface.

Environmental Image: While sensors often provide incomplete or faulty measurements for several reasons the environmental image is intended to provide a firewall against malfunctions of single sensors. A malfunction of a sensor may affect the quality of the environmental image, but may not require a change in the behavior of the control application.

The motivation for the introduction of this system of interfaces and components was the reduction of system complexity at cluster and control application level and the possibility of software reuse. Because the control application does not directly rely on the sensor measurements, the same control program can be used with different sensor configurations. The user interface is also part of the application interface.

Timing

In the abstract, the model described above could use any possible communication and computation schedule. Nevertheless, there are several reasons, that drove us towards a time-triggered communication and computation pattern.

Sensor fusion is the combination of two or more sensor measurements to form a new enhanced value. To find a sensor agreement of some sensor readings of a real-time object, either the measurements are performed and delivered so frequently, that the actual instant of measurement is of no concern or the sensor fusion process must know the timely relation between measurements, e.g. by knowing the instants of measurement [EP01]. If all measurements are performed simultaneously the timely relation between measurements is implicitly defined. This leads us to the requirement of a globally synchronized timebase available to all nodes in the system.

Two-sensor fusion algorithms require the presence of both sensor's measurements to compute an agreed value. Although it makes some sense to fuse values from alternately measuring sensors, most fusion calculation can be kept simpler, if both values are guaranteed to be measured at the same time. However, such simultaneously measured values can be transmitted at different instants, e. g. by serial communication line.

In a Time-Triggered Architecture [SHS+97] all instants for measurement and transmission are a priori known to all components. All components have access to a global time and communication can be synchronized with measurement instants known by all nodes. Furthermore, deterministic timing behavior is a key feature for replication of systems or subsystems to increase dependability [Pol94].

4 Case Study

The time-triggered system model introduced in the previous Section was used for the implementation of a model car, which acts as an autonomous robot with sensory inputs [Pet01].

The model comprises a mobile robot ("smart car") equipped with a suit of pivoted distance sensors, an electric drive, and a steering unit. Distance sensors, servo motors for sensor pivoting, driving and steering units are all separate nodes. Each node is implemented on a low-cost microcontroller and equipped with a smart transducer interface.

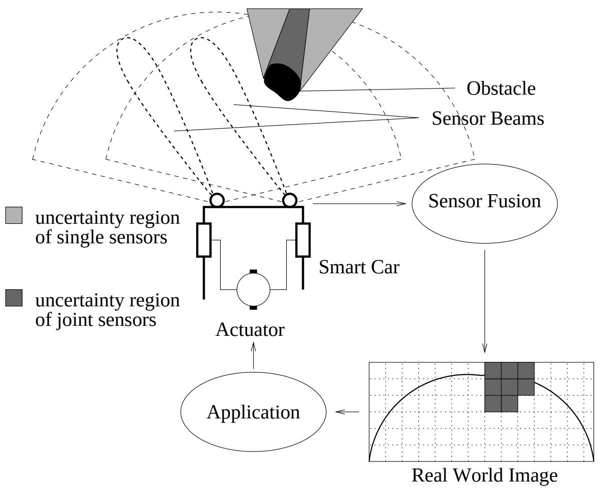

Figure 3: Smart Car

All nodes are build upon the Time-Triggered Protocol for SAE class A applications (TTP/A). TTP/A is a fieldbus protocol that supports the integration of smart transducer nodes with a predictable timing behavior. Furthermore, the protocol comprehends already a two level design methology with a cluster and a node level [PAG$^+$00]. By adding a control application level this model can easily be extended to fit into the proposed time-triggered system model. A detailed specification on TTP/A and its smart transducer interface can be found in [Kop00] and [HK01].

Figure 3 depicts the functionality of the smart car. At node level the distance sensors are swivelled around by servo motors so that they are able to scan the area in front of the robot. The sensors generate a value that corresponds to the distance of the object they are aimed at.

At fusion level, the data stream provided by the distance sensors is taken over by a data processing node that fuses the perceptions from the distance sensors and the directions they are aimed at with a model of the robot environment. In this model, the shapes of obstacles are stored and assigned with a probability value that decreases with the progression of time and increases when the object is re-scanned.

We used a sensor grid algorithm [BK91] to obtain the environment image. Based on these data, the control application makes decisions about direction and speed of further movement.

Although the control program is independent, it is also hosted in the data processing node so that the environment image does not have to be transmitted over the network. Thus, the necessary bus speed was kept low (about 20 KBit/s) which made it possible to use a low-cost single-wire bus.

5 Conclusion

We have proposed an architecture that incorporates smart transducer networks with sensor fusion processing and an environment image interface, which is sensor-independent.

The time-triggered sensor fusion model decomposes a real-time system into three levels: a node level, containing the sensors and the actuators; a cluster level that gathers measurements and performs sensor fusion; and a control application level where a control program makes control decisions based on environmental information provided by the cluster level. Because the control code is independent of the employed sensors, the system is open to sensor reconfigurations and reuse of the application control program.

The model has been already implemented in an autonomous mobile robot, where it helped to build a well-structured composable system. Furthermore this model will build the basis for a distributed time-triggered architecture for sensor fusion applications.

Acknowledgments

We would like to give special thanks to our colleague Michael Paulitsch for constructive comments on earlier versions of this paper. This work was supported in part by the Austrian Ministry of Science, project TTSB and by the European IST project DSoS under contract No IST-1999-11585.

References

[Bas00] T. Bass. Intrusion detection systems and multisensor data fusion: Creating cyberspace situational awareness. *Communications of the ACM*, 43(4):99–105, May 2000.

[BI98] R. R. Brooks and S. S. Iyengar. *Multi-Sensor Fusion: Fundamentals and Applications.* Prentice Hall, New Jersey, 1998.

[BK91] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, June 1991.

[BO00] M. Bedworth and J. O'Brien. The omnibus model: a new model of data fusion? *IEEE Aerospace and Electronics Systems Magazine*, 15(4):30–36, April 2000.

[Boy87] J. R. Boyd. A discourse on winning and losing. *Unpublished set of briefing slides available at Air University Library, Maxwell AFB, Alabama*, May 1987.

[Elm00] W. Elmenreich. An introduction to data fusion. Technical Report 11, Technische Universität Wien, Institut für Technische Informatik, July 2000.

[EP01] W. Elmenreich and S. Pitzek. Using sensor fusion in a TTP/A network. Technical Report 2, Technische Universität Wien, Institut für Technische Informatik, Jan. 2001.

[HK01] W. Elmenreich H. Kopetz, W.Haidinger. Specification of the smart sensor interface. Technical Report 7, IST-1999-11585 Dependable Systems of Systems (DSoS), 2001. Available at http://www.vmars.tuwien.ac.at.

[KHE00] H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. *Proceedings of the 3rd International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, March 2000.

[Kop98] H. Kopetz. The time-triggered model of computation. *Proceedings of the 19th IEEE Systems Symposium (RTSS98)*, Dec. 1998.

[Kop99] H. Kopetz. Do current technology trends enforce a paradigm shift in the industrial automation market? *Closing Keynote at the 7th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 99), Barcelona, Spain*, October 1999.

[Kop00] H. Kopetz et al. Specification of the TTP/A protocol. Technical report, Technische Universität Wien, Institut für Technische Informatik, March 2000. Available at http://www.ttpforum.org.

[Krü97] A. Krüger. *Interface Design for Time-Triggered Real-Time System Architectures.* PhD thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, April 1997.

[PAG+00] S. Poledna, H. Angelow, M. Glück, M. Pisecky, I. Smaili, G. Stöger, C. Tanzer, and G. Kroiss. TTP two level design approach: Tool support for composable fault-tolerant real-time systems. *SAE World Congress 2000, Detroit, Michigan, USA*, March 2000.

[Pet01] P. Peti. Monitoring and configuration of a TTP/A cluster in an autonomous mobile robot. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2001.

[Pol94] S. Poledna. Replica determinism in distributed real-time systems: A brief survey. *Real-Time Systems*, 6:289–316, 1994.

[SHS+97] C. Scheidler, G. Heiner, R. Sasse, E. Fuchs, H. Kopetz, and C. Temple. Time-Triggered Architecture (TTA). *Advances in Information Technologies: The Business Challenge, IOS Press*, 1997.

[Shu91] A. N. Shulsky. *Silent Warfare: Understanding the World of Intelligence.* Brassey's, New York, 1991.

[Vin97] S. Vinoski. CORBA: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications*, 35(2):46–55, Feb. 1997.

[WL90] E. Waltz and J. Llinas. *Multisensor Data Fusion.* Artech House, Norwood, Massachusetts, 1990.