

A Real-Time, Multi-Sensor Architecture for fusion of delayed observations: Application to Vehicle Localisation

C. Tessier, C. Cariou, C. Debain
CEMAGREF
24, avenue des Landais
BP 50085, 63172 Aubière, France
e-mail: cedric.tessier@cemagref.fr

F. Chausse, R. Chapuis
LASMEA - UMR 6602
24, avenue des Landais
63177 Aubière, France
e-mail: chausse@lasmea.univ-bpclermont.fr

C. Rousset
ECA
Rue des Frères Lumière BP 24
83078 Toulon Cedex 09, France
e-mail: cro@eca.fr

Abstract— This paper presents a software framework called AROCCAM that was developed to design and implement data fusion applications. This architecture permits to build applications in a very short time unburdening the user of sensor communication. Moreover, it manages unsynchronized sensors and delayed observations in an elegant manner that permits to the user to fuse those information easily, taking into account the environment perception date.

In this paper, a fusion methodology for delayed observations is first presented in order to point the problem of latency periods in a fusion system. These latency periods are then taking into account within our embedded architecture needing only a little effort from user. Finally, benefits of AROCCAM architecture are demonstrated via a real-time vehicle localisation experiment carried out with an outdoor robot.

I. INTRODUCTION

Nowadays, more and more vehicles are equipped with intelligent systems. These systems have to realize particular tasks such as: vehicle localisation, automatic guidance, obstacle avoidance, pedestrian detection,... To accomplish their tasks, they process sensor data. However, it is necessary to write piece of softwares to communicate with sensors. This task is tedious and most of all a lack of time. A solution to this problem is to use an embedded architecture. Such architectures permit to facilitate the development of algorithms by managing the communication between those algorithms and sensors. For instance, they can collect sensor data and control sensors without blocking algorithm mechanism.

The architecture proposed here has to respect several requirements:

- management of unsynchronized data and sensors latency,
- recording and replaying of sensor data in real-time,
- engineering requirements: re-usability, integration, maintenance, processing efficiency
- user requirements: easy of use, programming error detection.

When a system uses a single sensor, it can only sense a partial and incomplete part of the environment. By contrast, a multi-sensor approach is a way to improve environment perception. It's necessary to notify that a sensor provides an observation of the environment at a particular time. Another sensor provides a similar information at another time. The

difficulty of a multi-sensor system is to fuse sensor data with different dates.

- In some works [1], [2], [3], sensor data are fused without taking into account the fact that information are unsynchronized.
- In other works [4], the data sampling frequency is increased in order to consider that all information are synchronized each other. Unfortunately, this assumption is false since each sensor has a latency time, which can not be taken into account by this way.

Data fusion systems become more and more used, which motivated us to design an architecture to answer the problem of unsynchronized sensor data taking also into account sensor latency.

This paper is organized as follows. Section II depicts the works related to embedded architecture, emphasising assets and drawbacks of each approach. In section III, our architecture is described by presenting each module and their objectives. Functionalities that ease the development of embedded software are also enumerated. Section IV details the latency periods of observations: where they arise from? the consequences of fusion of delayed observations. A solution is suggested to deal with such observations in a data fusion system. Finally, a typical application involving our solution is given in the last section: real-time vehicle localisation.

II. RELATED WORKS

Since several decades, different embedded architectures have been developed, each answering to requirements of a particular field of applications. However, all these architectures are agree with the necessity to be divided in several components.

For instance, the LAAS architecture [5] has three hierarchical levels, having different temporal constraints and manipulating different data representations. The main asset of such a decomposition is the realisation of prototype software applications in a very short time. In the same manner, SCOOT-R [6], [7] the acronym for "Server and Client Object Oriented for the Real-Time", offers a framework for distributing tasks on multi-processing unit architecture. This software is in charge of communication between each

processing unit. This permits to realise time-consuming applications by distributing tasks on several computers.

Another feature pointed up in the SCOOT-R architecture is the real-time aspect. It consists in giving the sensor data to the algorithm as soon as it's available. In the case of this architecture, a communication between each components must be implemented, since it is a distributed application. In order to solve the problem of communication time in a real-time software, a real-time network is used. Moreover, the real-time aspect of the mechanism of SCOOT-R components obliges each component to work in a very short time not to be declared as defective. Unfortunately, the determination of the processing time of sensor data is sometimes not possible. The upper limit of the processing time can always be evaluated. In general, vision algorithms like road detection [8] can take more than 100ms per images. This reduce the utilisation of this architecture to mighty processing unit and to avoid time-consuming algorithms. In the same manner as SCOOT-R, RT-MAPS [9], [10], is divided into several components and date sensor data with an accurate clock as well. In that particular case, the aim of the dating is to use RT-MAPS like a numeric videotape recorder. Note that this date is the reception date of the sensor data. In a first time, it can record all data in a synchronized dated database. Then, it's possible to replay all these data later. The asset of this function is that it permits to improve an algorithm by testing it on the same databank or to compare several algorithms. Now, having compared several architectures for embedded applications, let's analyse the main functions of our architecture before discussing latency problem.

III. AROCCAM

In our architecture called AROCCAM, the acronym for "Architecture d'ordonnement de capteurs pour la création d'algorithmes modulaires", we pointed up the modular and simple aspect. As we want an easy to use architecture, AROCCAM is only divided into three components (Figure 1).

- A **driver module** is responsible for the communication with external entities like sensors, softwares, computers,... There is a particular driver module for a particular communication bus (IEEE, CAN network, Ethernet network, Serial ports,...).
- A **brik module** is an application algorithm. Thanks to a subscription to driver modules, a brik module receives directly sensor data without having to know the communication protocol. In general, the user has to write piece of software only in this area.
- **The heart module** is the final component. This component has not to be modified or adapted by the user. It is responsible for the communication between driver modules and brik modules, the threads creation, memory management, ...

Moreover, like RT-MAPS, our architecture dates accurately each sensor data gathered. This permits to the user to replay all the recorded sensor data at the desired speed. In order to replay exactly the sensor data in the same way

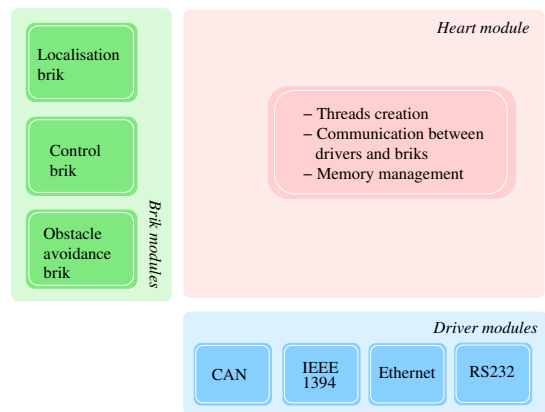


Fig. 1. AROCCAM Software Architecture.

that they were recorded (order of sensor data reception and interval time between two sensor data), the date of each sensor data corresponds to the reception date of the data by AROCCAM during recording.

After having described components involved in our architecture and their objectives, let's now analyse in details the problem of delayed observations fusion.

IV. A FUSION METHODOLOGY FOR DELAYED OBSERVATIONS

A. Presentation

The real-time aspect of embedded architectures is a feature pointed up. This feature consists in giving the sensor data to the algorithm as soon as it is available. However, as it was suggested by [11], most of the sensors have a latency time. Such architectures remain real-time sensor data collecting softwares but are not real-time environment perception softwares.

As sensor latency time can't be suppressed, it's therefore impossible to realise the real-time architecture last proposed. It means that the user must take these latenesses into account. In [12], a solution is suggested to the user. In this paper, we propose an elegant manner to manage this problem without making the development of an application algorithm more complex.

In the next section, is described in details the kind of latency periods that can appear.

B. Observation and latency period

The aim of the software that must be implemented in our architecture is to realize a particular task. In this section, we take the example of an accurate real-time positioning of a car on a digital map [13]. In this case, the vehicle is equipped with a video camera, oriented in the direction of the road, aiming at detecting the road. Then this detection permits to locate the vehicle thanks to a digital map listing road configurations.

An information that permits to participate to the achievement of the software task is called here **an observation**. It can be a sensor data or the processing result of this data. These observations are fusioned in the software. Let's

analyse in details the process to obtain an observation (Figure 2) in our example.

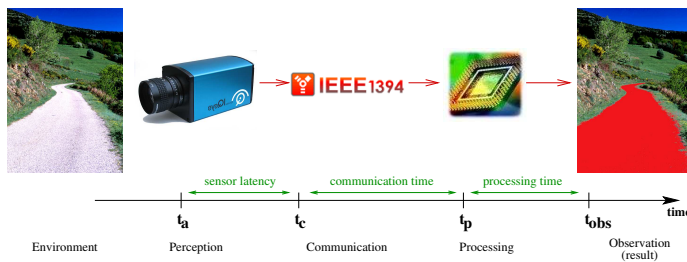


Fig. 2. The latency periods for obtaining an observation.

In general, the process to obtain an observation can be divided into three steps, each of them requiring a process time:

- *perception*. The sensor captures at time t_a a perception of the environment. Modern systems use smart sensors, i.e. an analog sensor linked to a local controller. The local controller gets the analog information, performs the analog to digital conversion, and sends the results through a digital bus like RS232 or CAN or Firewire. In the following of this paper, the word sensor will refer to a smart sensor. The time required by the local controller to prepare the result corresponds to the sensor latency.
- *communication*. As explained just above, it consists in sending the result through a digital bus at time t_c .
- *processing*. At time t_p , the embedded architecture, like AROCCAM, receives the sensor data. As our architecture works in real-time, we consider that the sensor data is dated as soon it's available by the computer. However, this information is not directly useable for the algorithm task. This last one has to process these data to extract a worth observation. The time required to treat the sensor data is the processing time.

In all embedded architectures, dating sensor data accurately, consists in timestamping those data with the date t_p like we do. However, the application algorithm can only fuse observations with the same date: the perception date. Sometimes, the data sheet of the sensor or directly the sensor during the experiment provides the sensor latency. When there is no information, a temporal calibration of sensors has to be done. Unfortunately, it's not always possible to estimate accurately all these parameters. We suggest to timestamp the observation with the date t_a and include with this date the dating precision.

C. Consequences of delayed observations fusion

To illustrate the fusion of delayed observations, let's keep our example of vehicle localisation. For vehicle localisation task, the estimated position is valuable only at a specific time. The robotic community calls this characteristic, the spatio-temporal localisation. It means that the estimated vehicle position is function of time. For each result produced by those systems, a time imprecision induced a spatial imprecision. A spatial time induced error of $10cm$ is given by an imprecision

of $1.5ms$ at $250km/h$ or $15ms$ at $25km/h$. To build an accurate localisation system, it's necessary to keep in mind the observations latency times.

D. Fusion of delayed observations

The aim of the system is to compute a result: the state vector, and this vector is function of time. In the previous section, the latency periods for obtaining an observation have been detailed. The problem here, is the fusion of a delayed observation with a particular state vector. It means that an observation received by system has to be used to update a particular state vector: the one that describes the system at observation's date. **AROCCAM offers an optimal estimation of the state vector at any time.**

V. EXPERIMENTATION AND RESULTS

The following section describes experimentations that were carried out to validate our approach. Thus, an outdoor localisation system was chosen.

A. Description

A terrestrial robot was used for the experiments. This research platform, of the Research Federation TIMS (Technologie de l'information, de la mobilité et de la sûreté), (Figure 3), was initially equipped with an on-board PC running Linux RTAI. Several sensors was added to it for the realisation of the system.

The objectives of this system is to locate the vehicle with $2m$ accuracy using only low-cost sensors.

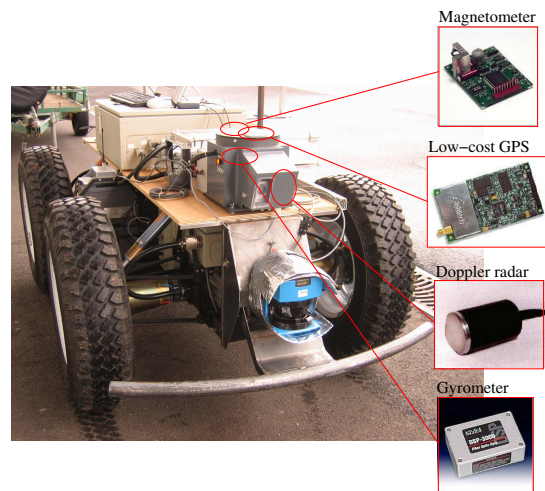


Fig. 3. The "RobucarTT" with its sensors

1) *Sensors*: Sensors used in this system are presented in Table I. The GPS system and the magnetometer permits to initialize the system: vehicle position and orientation. Then, two proprioceptive sensors: a Doppler radar and a gyrometer, and the GPS are used to locate the vehicle. A particle filter is employed to estimate the vehicle position.

Table I lists not only sensors but also their latency period and their acquisition frequency. These latency periods take into account the latency period of the sensor and also the

TABLE I
PROBABILITY VALUES FOR PERCEPTIVE TRIPLETS

sensor	latency period	acquisition frequency
magnetometer	5ms	16Hz
low-cost GPS	6 - 100ms*	10Hz
Doppler radar	-	10 - 77Hz
gyrometer	-	20Hz

*: supplied directly by the sensor.

communication time on the bus that links the sensor to the computer. We can also notice that all these sensors are not synchronized since they are affected by different latency period and different acquisition frequency. The use of AROCCAM seems to be necessary.

2) *Vehicle progress model*: In this part, we focus on small displacements. Assuming the flatness of the environment where the robot is running, position and attitude of the vehicle are resumed to the position of the vehicle in the 2D plane (Oxy) defined by $x(t)$ and $y(t)$ and orientation of the car with respect to the (Ox) axis given by $\theta(t)$. Measurements from the vehicle are the average speed $V(t)$ given by the Doppler radar and angular speed $\omega(t)$ given by the gyrometer.

Figure 4 shows the vehicle state (position and orientation) at two particular moments: t_k and t_{k+1} .

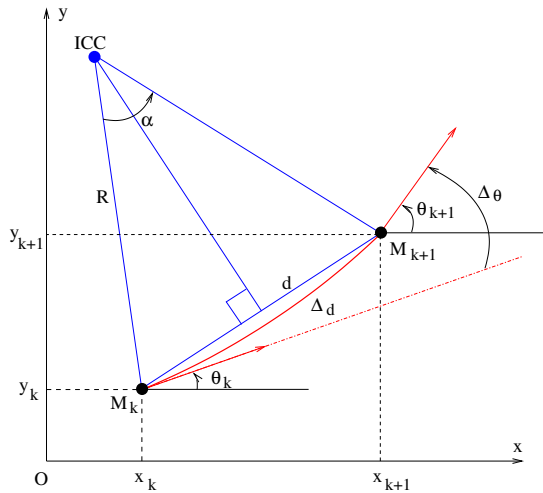


Fig. 4. Small displacement between two successive positions.

Relation between vehicle displacement Δ_d and average speed V is given by: $\Delta_d \approx d = Te \cdot V$ where Te is the sampling period. In the same way, relation between vehicle rotation Δ_θ and angular speed ω is given by: $\Delta_\theta = Te \cdot \omega$.

Thus, the vehicle progress model is:

$$\begin{cases} x_{k+1} = x_k + \Delta_d \cdot \cos(\theta_k + \Delta_\theta/2) \\ y_{k+1} = y_k + \Delta_d \cdot \sin(\theta_k + \Delta_\theta/2) \\ \theta_{k+1} = \theta_k + \Delta_\theta \end{cases} \quad (1)$$

3) *Why use AROCCAM*: It's necessary to use the AROCCAM architecture in this system since:

- **sensors are unsynchronized**: equation (1) supposed that the two proprioceptive sensors supply synchronized observations.

- **sensors have a latency period**.

An elegant solution to solve these difficulties is to use the architecture suggested in this paper.

B. Experimentation results

The scenario used to validate our architecture is presented in figure 5. As we can see, the trajectory is complex and contains several curves. This path have been obtained by a manual run. During the experiment, successive absolute positions giving by a GPS RTK (THALES Navigation) with 2cm accuracy, are recorded. The position estimated by our method is compared each time with the GPS-RTK reference trajectory by computing the difference (error) between them.



Fig. 5. Trajectory realised in the experiment.

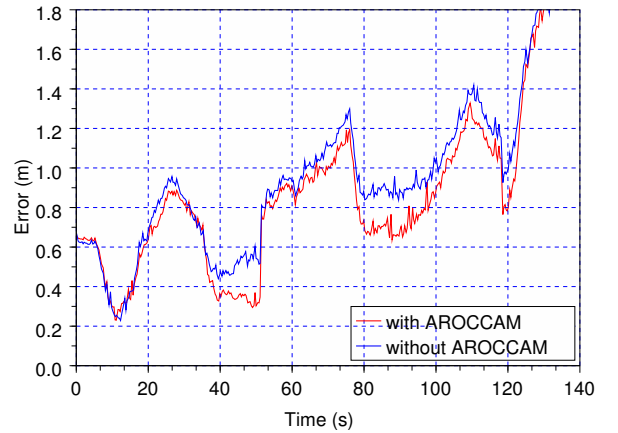


Fig. 6. Localisation error with two architectures.

On the following figure (figure 6), the localisation error is depicted for two experimentations:

- In red line: localisation error with the use of AROCCAM, taking into account the latency periods.
- In blue line: localisation error with a classical embedded architecture.

We can notice that, as expected, latency periods affect the localisation system results. In our example, a maximal error of 20cm is added to the system when a classical architecture is used. These results permit to show the benefit given by our approach.

VI. CONCLUSION

This paper has proposed an embedded architecture for the fusion of delayed observations. A fusion methodology has first been designed. First a description of the latency periods from the sensor to the data fused in the algorithm has been presented. The consequences of delayed observations fusion have been explained through the example of the vehicle localisation and a method has been suggested by the use of AROCCAM. Finally, a vehicle localisation application has been proposed to illustrate our architecture.

Even though the AROCCAM architecture allows to reduce errors in fusion of unsynchronized information, our aim here is to draw conclusions not from the demonstration but from the process of building embedded applications. We think that the AROCCAM architecture offers an elegant and easy manner to build fusion algorithm. The benefit was really substantial: during all the programming and debugging process, we never had problem with thread communication, real-time multithread management, transmission of data between different system,... All these aspects were dealt with by our software architecture.

Moreover several other applications were developed under AROCCAM with no difficulties and good results.

REFERENCES

- [1] Gianluca Ippoliti, Leopoldo Jetto, Alessia La Manna, and Sauro Longhi. Improving the robustness properties of robot localization procedures with respect to environment features uncertainties. In *International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, April 2005.
- [2] C. Kwok, D. Fox, and M. Meila. Real-time particle filters. *IEEE, Sequential State Estimation*, 92(2), 2004.
- [3] David Filliat. *Cartographie et estimation globale de la position pour un robot mobile autonome*. PhD thesis, LIP6/AnimatLab, Université Pierre et Marie Curie, Paris, France, December 2001. Spécialité Informatique.
- [4] Marc-Michael Meinecke and Marian-Andrzej Obojski. Potentials and limitations of pre-crash systems for pedestrian protection. In *International Workshop on Intelligent Transportation*, Hamburg/Germany, March 15-16 2005.
- [5] Rachid Alami, Raja Chatila, Sara Fleury, Matthieu Herrb, Felix Ingrand, Maher Khatib, Benoit Morisset, Philippe Moutarlier, and Thierry Siméon. Around the lab in 40 days... In *IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000.
- [6] Khaled Chaaban, Paul Crubillé, and Mohamed Shawky. *Computer Science*, chapter Real-Time Framework for Distributed Embedded Systems, pages 96–107. Springer-Verlag GmbH, 2004.
- [7] Khaled Chaaban, Paul Crubillé, and Mohamed Shawky. Real-time embedded architecture for intelligents vehicles. In *Proceeding of the 5th Real-time Linux workshop*, Valencia, Spain, November 2003.
- [8] P. Jeong and S. Nedeveschi. Efficient and robust classification method using combined feature vector for lane detection. *Ieee transactions on circuits and systems for video technology*, 15(4):528–537, 2005.
- [9] Iyad Abuhadrous, Fawzi Nashashibi, and Claude Laurgeau. Multi-sensor fusion (gps, imu, odometers) for precise land vehicle localisation using rtmmaps. In *11th International Conference on Advanced Robotics ICAR*, 2003.
- [10] Fawzi Nashashibi. Rtm@ps: a framework for prototyping automatic multisensor applications. In *IEEE Intelligent Vehicles Symposium*, October 3-5 2000.
- [11] Mikael Kais, Laurent Bouraoui, Steeve Morin, Arnaud Porterie, and Michel Parent. A collaborative perception framework for intelligent transportation system applications. In *Intelligent Transportation Systems Conference ITSC*, Vienna, Austria, September 13-16 2005.
- [12] Iyad Abuhadrous. *Système embarqué temps réel de localisation et de modélisation 3D par fusion multi-capteur*. PhD thesis, Ecole des Mines de Paris, January 2005.
- [13] Jean Laneurit, Roland Chapuis, and Frédéric Chausse. Accurate vehicle positioning on a numerical map. *International Journal of Control, Automation, and Systems*, 3(1):15–31, March 2005.