ABSTRACT
This paper presents the architecture of a multimedia sensor network, especially dedicated to video surveillance. The founding ideas of our architecture are: (a) continuous 3D real-time reconstruction of the monitored area in which streams originated from video sensors are merged. (b) locating parts of data analysis/extraction tasks on the sensors themselves in a way to reduce drastically the network bandwidth usage, traditionally consumed by video stream. Hence video sensors perform in addition to capturing tasks, data analysis and extraction of features needed for the 3D reconstruction. We validated our approach through real experimentations, in particular on the video sensor.

Keywords
Multimedia sensor networks, video capture devices, 3D real-time reconstruction.

1. INTRODUCTION
Recent decade has witnessed significant advances (miniaturized hardware components, radio frequency communication...) in wireless sensor networks which emerge as one of the most promising technologies for the 21st century [28]. In fact, they present huge potential in several application domains ranging from health care applications to military applications. In such networks several nodes, each of which composed of a processing unit, a sensing unit and a radio communication unit, collaborate with each others in order to collect data from physical environment (e.g., temperature, humidity, etc.) and to transmit it, usually by the means of a multi-hop radio communication, to a base station (sink) for processing.

At the same time, sensor networks raise many challenging research issues, especially because of their restricted computing capabilities and their limited lifetime, as nodes are driven by batteries. This latter feature makes energy optimization the first and the dominant studying factor no matter what the problem is.

More recently, the availability of new miniaturized multimedia hardware (CMOS cameras and microphones for instance) at reasonable prices has allowed the development of Wireless Multimedia Sensor Networks (WMSN) for a large number of real applications in several domains (home automation, old people assistance, etc.). What differs WMSN from classical sensor network is their ability to collect multimedia content (video and audio streams, still images, etc.) in addition to scalar sensor data. As noted in [26]: “In addition to the ability to retrieve multimedia data, WMSNs will also be able to store, process in real-time, correlate and fuse multimedia data originated from heterogeneous sources”.

1.1 Motivations
The management of multimedia data within limited environments as those of sensor networks poses new issues at all levels of the applications because on one hand of the huge volume of the produced data (the size of videos or images is far away much more important than scalar data) and on the other hand of the “continuous constraint”, also called real-time constraint, related to multimedia data i.e., their delivery is time sensitive. It is obvious that the management and the delivery of multimedia data require much more resources in terms of processing power as well as network resources, mainly bandwidth.

Furthermore, the process of multimedia data correlation and fusion differs fundamentally from that of scalar data in the sense that the latter is primarily concerned with average computing while the first is much more complex (to the best of our knowledge no well known approach is available1).

In this paper, we tackle the problem of managing WMSNs by specifically focusing on these two issues: (a) saving/reducing the network bandwidth required to deliver continuous multimedia data and (b) proposing a novel approach for multimedia data fusion. The targeted application is video surveillance sensor network.

1.2 Contributions
The key idea behind our proposition is to “continuously” construct a 3D representation of the monitored area, in which video streams originated from the video sensors are fused. In other words, the “views” of the sensor nodes are merged in the 3D scene of the monitored region. This approach presents many interesting features, in particular for resources limited environments like those of sensor networks.

The first important advantage of using 3D representation is its flexibility in the sense that it allows a comprehensive representation of the observed scene. In fact, it is commonly known that exploiting directly raw video data is very hard or even not possible if it is not pre-processed or annotated by experts/annotators [29]. Moreover, it is not obvious to get

1 In general, few approaches were developed to specific data and contexts [25].
Another advantage of 3D reconstruction is its ability to answer some spatio-temporal requests that are very hard to handle with raw video data. To illustrate, let’s consider the example of supermarkets in which managers are interested to know for example the impact of some new department dispositions on the behaviour of customers i.e., measuring the average waiting time of customers at the fish department. Unless explicitly “counting” customers throughout video surveillance system, no automatic approach is conceivable to answer such queries. Oppositely, it is relatively easier to handle such queries in a system based on 3D reconstruction even though its precision is approximate (the application does not require person recognition but just form recognition). More over 3D reconstruction allows some “angles of views” of the monitored scene that raw data sensor cannot offer (see our practical example presented in section 3.3 and in figure 7).

In a 3D reconstruction system, each node performs two main tasks: (a) data extraction (processing phase) during which the node analyses the captured stream and extracts the required information necessary for the 3D reconstruction. (b) Instead of sending captured stream, the node sends only the extracted information to the fusion server which in turn reconstructs the 3D view of the monitored area. By doing so, an important network bandwidth reduction is achieved, which constitutes an important requirement for the system scalability. It has to be noted that in our approach, it is possible, upon the request from the observer/manager, that one node sends the entire video stream in case that complementary information is needed.

However, some important questions have to be answered: first what kind of data extraction the node has to perform in order to ensure 3D real-time reconstruction by the fusion server? And more importantly is the sensor node able to handle such processing tasks? In this paper, we focus on the development of such a sensor node by adopting real configuration and measuring its performance.

The overall paper is organised as follow: Section 2 presents related works. Section 3 details our proposed architecture. In Section 4, we present the processing tasks that the sensor node has to perform. Section 5 resumes our preliminary experimental results and Section 6 concludes this paper.

2. RELATED WORKS

As an emerging research domain, wireless multimedia sensor networks present interesting research challenges, especially due to the several constraints that they impose. In [4] [5], the authors present the Cyclops image capturing and inference module. The latter has been designed for light-weight imaging and can be easily interfaced with other sensor nodes like MICA2 [27]. Other research works tackled the problem of developing multimedia capturing devices with the associated processing tasks [6]. They pointed out that using 32 bits processors is more convenient for this purposes than 8 bits processors. This conclusion has also been confirmed in terms of energy consumption by the work presented in [24].

On the other hand, other research works focused on the development of compression schemes, especially tailored for WMSN [7] [8] [9] [10] [11]. The common point between those works is that they are founded on JPEG 2000 compression scheme. In those works, instead of sending an entire video stream, they just send periodically images telling the fusion server what changes have been occurred in the observed area. Hence an important network bandwidth reduction is achieved.

Other works have addressed the network delivery issue with the associated quality of service requirement. Propositions have been made at the MAC level [16] [17] and at the network level [12] [13] [14] [15]. The objective remains to guarantee a certain quality of service by respecting the “real-time” constraint of the continuous data.

Experimental experiences have also been conducted, mainly in the development of video sensor networks [19] [20] [8]. For instance, in Panoptes [8] platform, PDAs with 64 MB of RAM equipped with video capture device have been used. The platform also includes spatial compression technique (but no temporal compression has been used) combined with distributed filters and video streams priority management.

We note here that our work could be seen as complementary to those works in the sense that the developed ideas could, in some cases be integrated in our approach.

3. SYSTEM ARCHITECTURE

Figure 1 shows the global architecture of our proposal. It is composed of three main elements named: the capture device, the fusion server and the end-user server. We detail the features of each component in the following sections.

![Figure 1: functional architecture.](image-url)
3.1 Capture devices
The capture device is a set of devices and software that allows to capture and process the images. In order to obtain a full visual surveillance of rooms and corridors, it is often necessary to use a huge number of sensors intelligently placed in the building.

The module « capture device » combines a digital camera and embedded software. The digital camera captures images that are processed by the program in order to produce vectorial data. These data describe under a compact form, the differences between the camera images and a referential image previously memorized when the place was free of people.

The goal of the pipeline of image processing tasks consists in sending only vectorial data instead of the captured images to the fusion server over the wireless network.

The size of the generated vectorial data is very small compared to the size of the original digital image. Then in the extreme case where the captured image is similar to the referential image, no data are sent by the device. More generally, when the program detects an object in the captured image only a set of points coordinates of the borders of the object image is sent. These data admit a small size compared to the original image size, easily compressible in order to reduce the cost of the transmission over the network.

In order to compare the two approaches, in terms of reduction of the size of transmitted data, we present the following example: A video camera (640x480 pixels) sends a reference image which is compressed using the JPEG algorithm with 50% quality level, and has a size of 30 Ko. The same image is processed in order to extract the border as 38 segments, stored in 4 short integers each (2 bytes), the total size is then 304 bytes! We hence obtain a ratio of about 1/100, which represents a substantial gain. But, we must remind that the price for this gain is the necessity of processing the video images at the sensor level (see next section).

3.2 Fusion server
The main functionality of the fusion server consists in merging information sent by the capture device in order to place the objects detected into the 3D environment.

The fusion server receives the segments extracted from all the sensors. This simple information seems not to be sufficient to place objects in the 3D scene, but combining these 2D data with precise knowledge about the position and the direction of the sensors (early computed and recorded), it allows the deduction of a probable position of the viewed object. Some simple geometrical rules may be applied to construct the extrapolation of 3D information from 2D segments data. For example, if we consider the simple hypothesis that a person viewed by the camera is standing on the floor, we can easily estimate its 3D position (cf. figure 2).

However, it is possible to realize this processing only if the system associates a referential to the scene captured for each sensor. These referential may be computed once and for all by a pseudo-automatic algorithm (cf. figure 3) of camera calibration.

The goal of this process is to precisely place the camera in the viewed scene by cliquing in well-known areas on the image. In the example shown in figure 3, the colored circles represent the points placed by the user on the image. The coordinates of these points must be well-known on the scene. In our case, we use the corner of the walls as origin of our referential and the axes defined by the intersections of walls and the floor as main axes. Moreover it is important to know the real distance between the points.

In the following computations, we consider the digital camera as a perfect sensor. In practice, we apply a geometrical correction to the captured image in order to reduce the deformations produced by the lens.

The position of the 3D referential is obtained using a method similar to the one proposed by Hirokazu Kato and Mark Billinghurst [1] for the placement of square marks. Considering O as the optical origin of the camera and that the rectangle ABCD (lengths l_{AB} and l_{AD} are known) is projected on the camera image, the points A’ B’ C’ D’ are the projected of ABCD (cf. figure 4).
Considering that the two vectors $\mathbf{AB}$ and $\mathbf{DC}$ are equals, we can deduce:

$$V_1 = \mathbf{OB} \times \mathbf{OA'}, \quad V_2 = \mathbf{OD'} \times \mathbf{OC'}, \quad \mathbf{AB} = \mathbf{DC} = l_{AB} \frac{V_1 \times V_2}{|V_1 \times V_2|} \quad (1)$$

Vector $\mathbf{AD}$ is obtained by the same way, and then we obtain the $Z$ direction of our referential by:

$$Z = \frac{\mathbf{AB} \times \mathbf{AD}}{\| \mathbf{AB} \times \mathbf{AD} \|}$$

Now, we have to place one of the points of the ABCD rectangle in the 3D scene. To do that, we must calculate the distance between a point of the rectangle and the origin of the camera. We need complementary information as the angle ($\alpha$) representing the camera’s field of view:

$$d = \frac{1}{2} \tan \left( \frac{\theta}{2} \right)$$

The following classic equation expresses the position of the projected $\mathbf{A'}$ of the point $\mathbf{A}$ in the camera referential. Figure 5 shows the principle of the projective camera i.e. the projection of a point $\mathbf{A}$ on the screen.

$$\begin{pmatrix} \frac{A_x}{d} \\ \frac{A_y}{d} \\ \frac{A_z}{d} \end{pmatrix} \quad \text{and then} \quad \begin{pmatrix} A_x' \\ A_y' \\ A_z' \end{pmatrix} = \begin{pmatrix} \frac{A_x A_z + d A_y}{d} \\ \frac{A_y A_z + d A_x}{d} \\ \frac{A_z A_x + d A_y}{d} \end{pmatrix} \quad (2)$$

The depth $Z_A$ is calculated using that $\mathbf{B'}$ is both the projected of $\mathbf{B}$ and the projected of $\mathbf{A'} + \mathbf{AB}$ ($\mathbf{AB}$ obtained by equation 1):

$$\begin{pmatrix} d_A + AB_x \\ d_A + AB_y \\ d_A + AB_z \end{pmatrix} = \begin{pmatrix} \frac{A_x A_z + d A_y}{d} \\ \frac{A_y A_z + d A_x}{d} \\ \frac{A_z A_x + d A_y}{d} \end{pmatrix} \quad (3)$$

Using the expression of the position of $\mathbf{A}$ (eq. 2) in the equation [3], we obtain the following equations system, which admits only one unknown $A_z$:

$$\begin{cases} B_x = \frac{A_x A_z + d A_y}{A_z + AB_z} \\ B_y = \frac{A_y A_z + d A_x}{A_z + AB_z} \\ B_z = \frac{A_z A_x + d A_y}{A_z + AB_z} \end{cases}$$

In order to increase the precision of the position $\mathbf{A}$, we use all the projected points of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ using the same method. The precision of the calculus is sensitive to the errors due to the image sampling process and by the camera calibration.

In Figure 3, we use the placement of the referential to show a meshing over each plane ($\mathbf{O}, \mathbf{X}, \mathbf{Y}$), ($\mathbf{O}, \mathbf{X}, \mathbf{Z}$), ($\mathbf{O}, \mathbf{Y}, \mathbf{Z}$). These meshings allow to check the calibration quality.

When the geometric links between the captured image and the 3D model is defined, it is possible to compute the position of the 2D vectors sent by each capture devices and merge these positions to obtain an average position. It is exactly the role of the “fusion server” component which then sends the result of this merging to the “end-user server” as 3D primitives.

### 3.3 The end-user server

The end-user server proposes an interface representing in 3D the geometrical model of the filmed scene and the set of complementary data associated to the detected elements. To realize this interface, we use classical software of the virtual reality domain, allowing to browse easily in the 3D space in order to interactively visualize the 3D model populated by the detected elements (images of the people moving in the building for instance) placed in the appropriate position in the 3D map.
We can observe in the synthesis image the avatar of the detected person mapped on a simple mesh and placed on the scene by each camera. Then we can notice the homogeneity of localization by the two captors, the small visible differences are due to the captured images sampling.

4. 3D CAPTURE DEVICE

In this section, we first detail the image processing tasks devolved in the sensor and then we discuss about implementation problems over the platform.

4.1 Processing tasks

The processing tasks running on the sensor have to extract geometrical primitives from the captured image. In this first version of the software, the primitives are only linear segments that allow the identification of all the linear part of the shapes in the image.

Many methods for primitive extraction from an image are available, mainly using the detection of interest points, like Canny edge operator [24] or Harris corners detection [2]. These methods are very efficient and strong but not efficient in our real time case because of their complexity.

We study here a particular case of the vectorisation in real time of the border of a 2D area obtained after the difference between two similar images. The pipeline of image processing tasks operates in three steps:

- The digital sensor which manages the capture of images as a matrix of pixels produces sampled data at about 30 images per second. These sensors may be remotely configured using many parameters (B&W or color images, saturation, contrast, etc.). It permits to ask different versions of the same image depending if the sensor produces images for the automatic watching process or for a visual identification by a human observer.

- The filtering module, applied on the sensor image, uses a pipeline of image processing tasks in order to extract the borders of detected elements. The first step (cf. figure 8a) consists in computing the difference between the last captured image and the reference one. More precisely this resulting image is compared to a threshold in order to produce an image only composed of white pixels (value 1) representing the detected objet over a black background (value 0). Then the application of a convolution filter allows to create a new image where the borders of the objects are drawn. The implemented filter is a simple 3x3 Laplace form filter [3], it produces a double line border where the interior is characterized by a light grey line (value over 4) and a dark grey exterior line (value<4), the value 4 is reserved for the background color (cf. figure 8b).

- The real time constraints impose to simplify the border vectorisation algorithm. It consists in constructing a chain of linear segments that follows the dark grey line of the previous processing. The linear segment grows while the curved border is similar as a line. When a curvature is detected, the segment is closed and a new approximation segment is created. This algorithm is repeated to construct a broken line following the entire border of the detected object.

The complexity of these two first steps is in $O(n)$ where $n$ is the number of pixels of the image.

4.2 Current material configuration

In [6], I. Downes and all present a study about image sensors developments. They show that using a 32 bits processor is more recommended than 8 bits processor for image processing tasks. For example, the time processing for an image convolution task is 16 times longer using a 8 bits processor (ATMEL ATmega128, at 4 Mhz) than using 32 bits processor (ARM7, at 48Mhz), while the energetic consumption is only 6 times higher.

In our experimentations, presented in the following section, we use a small Fox Board card (small computer: 66x72mm, 37gr) from ACME Systems [23]. The main feature of this card is that it includes open source codes and allows many hardware interfaces with other multimedia devices via USB ports and one network interface. Moreover the performance/price rate is very attractive (about 180€ for the entire capture module).

The Fox Board card is an embedded system, with an ETRAX 100LX processor (100MIPS, 32 bit, RISC, 100MHz) and 16Mo RAM (cf. figure 9). A native web server running on the card allows easily integration of it in our network. Our FoxBoard card is connected to a Labtec Pro webcam and a WIFI adapter via the two USB 1.1 connectors. We develop our real time code in C++ under Linux.

Figure 8: pipeline of the image processing.

Figure 9: our Fox Board capture device.
5. PRELIMINARY EXPERIMENTAL RESULTS

In our experiments, we mainly focused on the performance of our capture device. More specifically, we were interested in answering the following question: is the material configuration we adopted able to support both data acquisition and features extraction for 3D real-time reconstruction?

We characterized the performance of the capture device by varying the size of the captured images and measuring the processing time accordingly. The results are plotted in figure 10.

As shown in the figure, no results are plotted for big sized images (1024x768). The reason is that the capture device has limited memory size (16 MB only) which does not allow it to handle those images. We note that recent versions of the used Fox Board have more than 64 MB of memory size, which will be sufficient.

As expected, we observed that the processing time increases proportionally to the size of the captured images. Hence, for low resolution videos (320x200), we have obtained 30 processed images per second which means that the processing time has no negative effects on the produced streams. However, with higher resolutions the capture device was not able to satisfy the real-time constraint (e.g. 30 images/s). For example, the sensor node was able to process 16 images per second when their size was 640x480.

We also plotted in figure 10 the processing time of each task. We observed, as expected, that no difference is notable between filtering and convolution (these two tasks are intensive memory access tasks). The reason is that memory access are “expensive” in terms of access time on the used Fox Board because it uses flash memory. Hence, memory access time overlaps the processing time. This is not the case for vectorisation task since it does not require much memory access. To verify this point, we have used instead of the Fox board a PC laptop equipped with Intel Pentium M processor (800 MHz) and 512 MB memory. The obtained results are plotted in figure 11. These results confirmed the expected difference between filtering and convolution tasks. The vectorisation remains the less time processing consumption task.

With regard to these obtained results, we can deduce that the overall performances of the capture device can easily fulfill the target application requirements, namely video surveillance. Furthermore, as presented in the introduction section, we have also designed our architecture to handle some special situations in which more precision is needed (for instance to get details about an event). For this purpose, we have included a service that could be requested from any video sensor to deliver the captured video stream to the server instead of sending only extracted features. By doing so, our architecture is able to respond efficiently to some emergency situations (intrusion for example).

Finally, we believe that our approach presents an acceptable compromise between system resources optimization, mainly network bandwidth and the target application requirements (precision).

6. CONCLUSION AND FUTURE WORKS

Wireless multimedia sensor network is an emerging domain that poses a number of challenges, all the more multimedia data are resource consuming. In this paper, we presented the design and the preliminary performance of our system, especially tailored for video surveillance application. The main concerns that have guided our proposal are system resources optimization, particularly network bandwidth and allowing comprehensive video data fusion/exploitation. Our founding idea was real-time 3D reconstruction of the observed scene. While this architecture presents several advantages (as noted previously), it requires however that sensor nodes perform some additional tasks in addition to the capturing tasks.

In this paper, we particularly focused on this point by implementing and testing a real capture device. The latter uses Fox Board card (66x72mm, 37gr, 16MB) interfaced with a webcam and a wifi USB key. The experiments we conducted show that the proposed capture device can easily fulfill the target application requirements provided that low/medium resolution video are used. Furthermore, in spite of using an “old” Fox Board, we expect that recent versions can definitely handle high resolution videos.

This important result stimulates us to continue our work into two main directions: (a) first, we plan to use another material (e.g., Stargate Board [21] equipped with PXA-255 XScale 400 MHZ RISC processor) to implement our capture device. We expect better performances than Fox Board. (b) Second, we plan to validate our proposal by deeply studying the fusion server and the end-user server from two points of views: performance point of view in order to support the important workload generated by an important number of sensor nodes and exploitation point of view.
throughout the development of intuitive and interactive tools dedicated to data exploitation.

7. REFERENCES


[23] http://www.acmesystems.it


[33] http://www.acmesystems.it


