# Performance and results of the triple buffering built-in in a Raspberry PI to optimize the distribution of information from a Smart Sensor

Jose-Luis Jimenez-Garcia[1], Jose-Luis Poza-Lujan[2], Juan-Luis Posadas-Yagüe[2], David Baselga-Masia[1], José-Enrique Simó-Ten[2]

[1]School of Engineering in Computer Science (ETSINF)
[2]University Institute of Control Systems and Industrial Computing (ai2)
Universitat Politècnica de València (UPV, Camino de vera, s/n. 46022 Valencia (Spain)
[1]{jojigar1 ,dabama1}@inf.upv.es
[2]{jposadas,jopolu,jsimo}@ai2.upv.es

**Abstract.** Currently, 3D sensors can be considered an evolution of cameras by providing the image with its depth information. These sensors have a generic function and the programmer has to process the received information in order to be adapted and used in a specific environment. In robots navigation, the 3D information can be useful for basic behaviours such as “obstacles avoidance” or even more complex behaviours such as “maps generation”. In this article an image management system provided by the xTion intelligent sensor is presented. The xTion sensor provides a VGA image and a 3D depth, which allows it to be used for several purposes. In order to distribute the data, it is acquired, processed and sent to several clients with a triple buffer system modified to serve the most recent image to the client. The system is programmed in C for Linux and built-in in a Raspberry PI. The article exposes the performance and results from monitoring the frame's delay comparing it with a simple and a double buffer system widely used in this kind of systems.

**Keywords:** intelligent sensors, buffering, distributing information

## 1 Introduction

The use of sensors in robot navigation goes from basic survival behaviours like obstacle avoiding to complex task realization like map generation. In basic survival behaviours, commonly associated to reactive navigation, were commonly employed sensors with a specific task like depth or contact sensors. To complex behaviours, commonly associated to deliberative navigation has been used information from reactive information sensors. Currently, robots need sensors associated to behaviours. For example, distance sensors are utilized to avoid obstacles. However, this information is useful to build an environment map or to locate specific objects during the navigation. So, the information that a sensor produces is interesting it to be distributed to different behavioural processes.

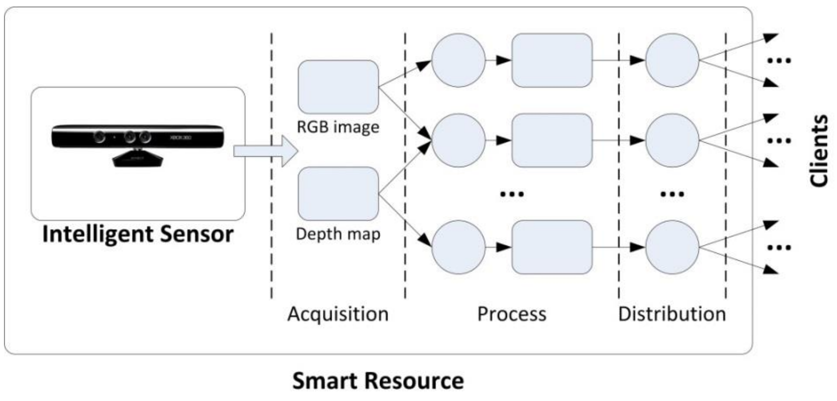

**Fig. 1.** Smart resource components.

The article is organized in this order: next section introduces the related work. Section 3 the implemented system is described, in section 4 the developed triple buffer variation is detailed. In section 5 the experiments and results obtained are described, finally, in section 6 the future working lines and conclusions are exposed.

## 2 Related work

An intelligent sensor is one that modifies its internal behaviour to optimize its ability to collect data from the physical world and communicate them in a responsive manner to a host system [1]. Nowadays intelligent sensors with great computation capabilities and large amount or information are offered by the technology, like image and depth of the provided frame.

There are several smart resources employing intelligent sensors [2] generally used on image-based systems. It seems convenient using the intelligent sensors both to reactive navigation to deliberative navigation [3] and [4]. However, using an intelligent sensor with several clients and with different client requirement implies offering the sensor information in the more suitable way to every client and distributes it so that the clients can control enough to perform their behaviours, especially if those behaviours are reactive.

The fact that the information transmitted by a smart sensor can be processed and adapted to the client needs turns it into a smart resource. [5]. Anyway, this process and its later distribution imply a processing cost which can decrease the information quality (figure 1).

In order to optimize the processing several methods to distribute internal information are used. Among them, the use of multiple buffers allows process synchronizing optimizing the speed [6]. The most used methods in graphic processing are the double and the triple buffering [7]. There are triple buffering implementations but they are usually oriented to only one consumer or they are simulations of theoretical models [8].

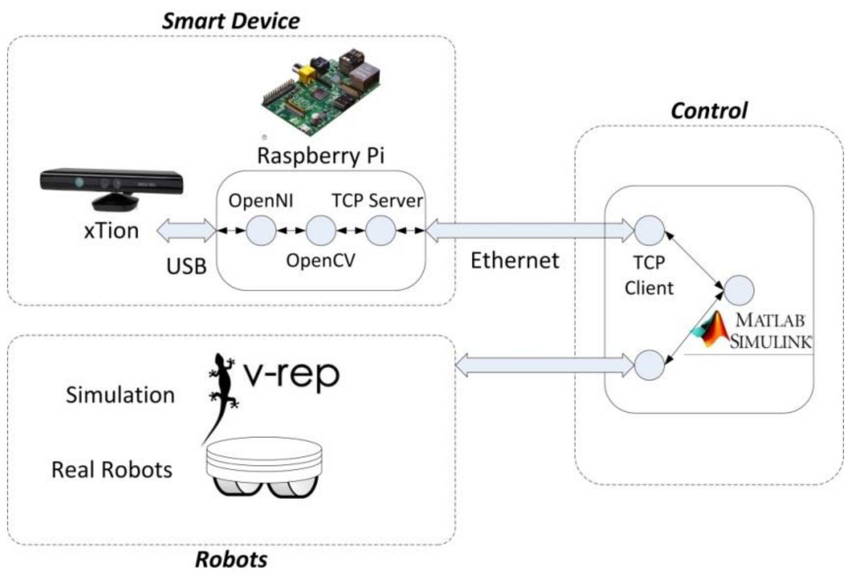

**Fig. 2.** System developed.

In this article it is described a smart device which involves the use of a smart sensor xTion [8] with a synchronized triple buffer with no waiting times. The algorithm's goal is providing frames at the speed (in frames per second or fps) that the client has requested, with minimum delay possible and without the need of synchronizing the data sending. In the article results of delay times measuring and quality of image sending frequency are presented.

## 3 System proposed description

The implemented smart resource acquires images from an xTion sensor to process it with a Raspberry Pi [9] system which distributes it with an Ethernet connexion. These dispositive were chosen because their low weight and ease of programming to make prototypes that can be easily embedded as soon as the system efficiency and strength has been checked. Nowadays the smart resource is been implemented to improve the efficiency in robot navigation control based on the improvement of QoS and QoC [10].

In Figure **2** it is showed the details of the system in which is used the smart resource. The smart device is using the OpenNI library [11] to the xTion sensor connection. The image processing is done using the OpenCV library [12], implementing the connection and distribution with the Linux socket library. The control algorithm are implemented in a server with MatLab [13] which processes the images to the generation of different behaviours that goes from the generation of trajectories to the map creation. Finally it is used the robot simulation environment V-REP [14] with the goal of including the smart resource in real robots.

The robot navigation requires the most recent information, according that, the triple buffer design presented in this article is oriented to sending the most recent frame employing the minimal processing time possible.

## 4 Method

Because of the robot control system needs the most recent image; the designed triple buffer optimizes the transmitted image making sure that it is the last one without employing more memory than a conventional triple buffer. In this article only has been experimented with a single image processing. Is planned to realize a multiprocessing with multiple clients in future researching. The triple buffer is managed by two threads: the acquisition and process thread (APT), and the sending thread (ST).

The buffer's behaviours are as follows: a buffer is used for the image being acquired. Another contains the image being sent and the third buffer is an auxiliary buffer to avoid bottleneck. The triple buffer is managed by two threads: the acquisition and process thread (APT), and the sending thread (ST). The APT changes between the acquisition buffer and the auxiliary while the ST ends sending its buffer. The Swap function does the buffer's pointers exchange which corresponds. When the exchange is done, the APT sends a signal to ST to pick the last image. The APT behaviour is as follows.

```
1: I ← 0
2: Adquisition() ← frame
3: while I < frame.MaxPixel do
4:    frame.Process()
5: end while
6: Swap()
7: Signal()
```

When the ST has sent an image it waits, using the wait() function, to be communicated by APT that the last image is available. As soon as the ST is unlocked by the APT from the wait function, the ST does the buffers pointer swap to send the last available image.

```
1: Wait()
2: Swap()
3: Send()
```

Because two threads are employed to decouple the acquisition from the distribution, we can find two scenarios: APT serves a frame faster than ST sends it and otherwise. In first case, the frame sequence is lost in exchange for gain immediacy in the information. In figure **3** the sending sequence is shown where it can be appreciated how the frame i+2 is lost because the frame i+3 is acquired before the frame i+1 ends being sent. In figure **4** the opposite case is shown, the ST is faster than the APT so no message in the sequence is lost.

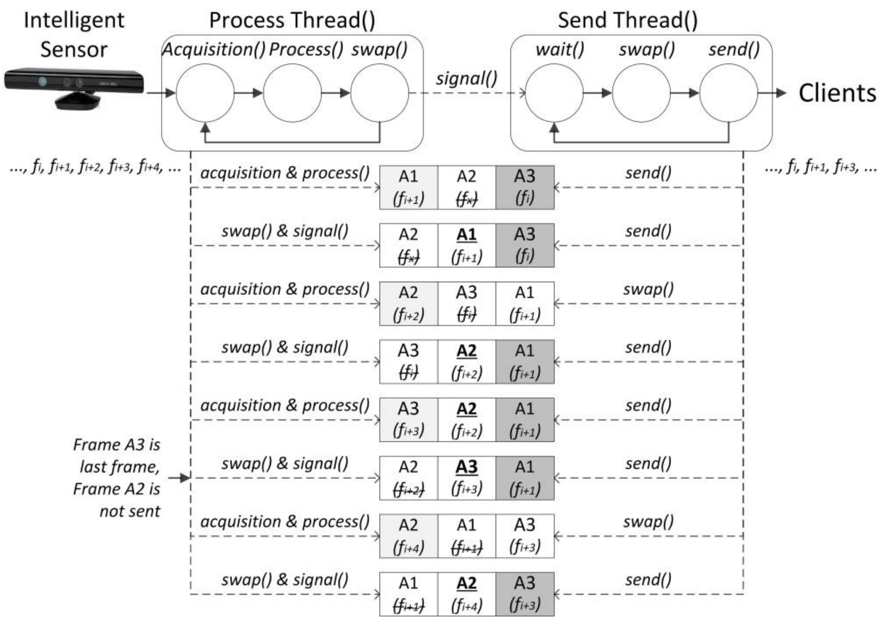
Intelligent Sensor
Process Thread()
Acquisition() Process() swap()
signal()
Send Thread()
wait() swap() send()
Clients
..., $f_i$, $f_{i+1}$, $f_{i+2}$, $f_{i+3}$, $f_{i+4}$, ...
..., $f_i$, $f_{i+1}$, $f_{i+3}$, ...
acquisition & process()
swap() & signal()
send()
swap()
Frame A3 is last frame, Frame A2 is not sent

**Fig. 3.** Sequence of operation of triple buffer with APT faster than the ST.

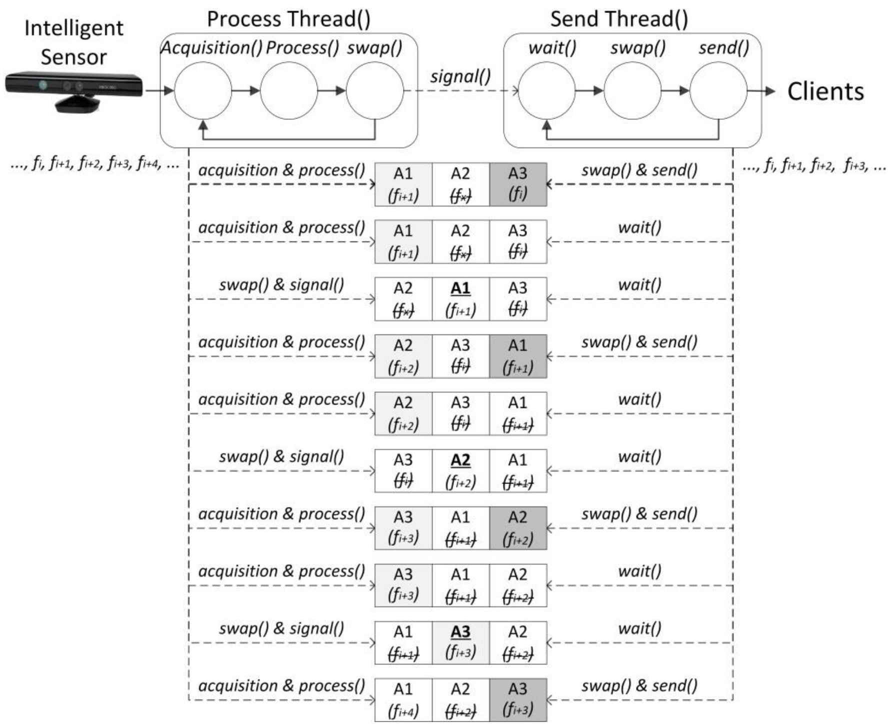

**Fig. 4.** Sequence of operation of triple buffer with ST faster than the APT.

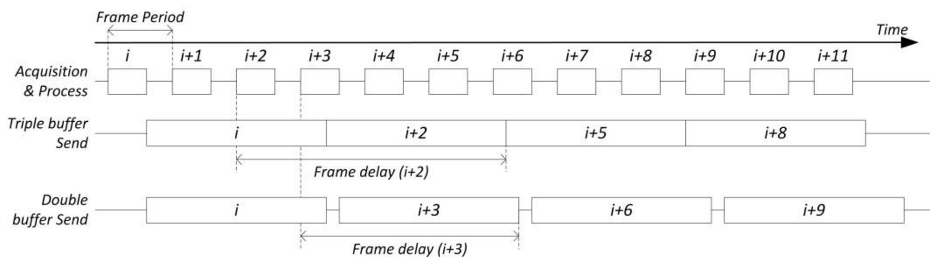

**Fig. 5. Variables measured at the experiments**

## 5 Experiments and results

To check the correct operation of triple buffer implemented, it has been tested using a computer as smart resource to check highest level of efficiency, and a raspberry PI on the final system. Because the system must operate in a distributed closed network, servers and clients work with fixed IP addresses, thereby, the effect of messages noise on network is reduced. All tests were performed with loads of 1000 frames and 30 frames per second (fps) as sampling rate. Previously, it has been verified that the number of frames not affect the operation. Has been measured times with one, two and four clients, comparing the performance of a simple, double and triple buffering based server. The measured variables are the average of the all the frame delays, the frame delay is defined in the equation 1.

$$T_{frame\_delay} = t_{adq} + t_{wp} + t_p + t_{ws} + t_s \tag{1}$$

In equation 1, $t_{adq}$ represents the time inverted to read a frame from the xTion, $t_{wp}$ represents the time that the APT is waiting to obtain the frame and the $t_p$ is the time used by the APT to process the frame, $t_{ws}$ is the time that the ST waits for the new frame, if it is necessary, and $t_s$ is the time inverted to send the frame to all clients. The results for the frame delay, expressed in microseconds, obtained are shown at table 1.

**Table 1.** Results of the experiments (in microseconds).

| | 1 client | | 2 clients | | 4 clients | |
|---|---|---|---|---|---|---|
| Variables | PC | RPI | PC | RPI | PC | RPI |
| Simple buffer | 67350 | 3176925 | 66936 | 592390 | 66945 | 1293636 |
| Double buffer | 67350 | 203086 | 66936 | 334824 | 66945 | 640506 |
| Triple buffer | 66940 | 199647 | 67386 | 332900 | 66930 | 611752 |

The main difference between double and triple buffering is that the triple buffer always has a frame ready to send. However, the triple buffer adds more delay time. These results are predictable but it's necessary that the smart resource can provide to the clients in order to improve the system with quality of service policies (figure 5).

## 6 Conclusions

In this article a system of acquisition, processing and sending of image frames from an intelligent sensor to several clients interested in a processed image has been presented. To provide the most recent image, a triple buffer that ensures that constraint has been developed.

As can be seen in the table 1, when the image server is a powerful computer (PC) differences between the types of buffer are not significant. However, an embedded system provides less efficient results for a simple buffer, for a double buffering and triple buffering delay times is less in the case of double buffering when we increase the client number. So that, triple buffer offers the same updated information with an added immediacy and more efficiently than double.

In triple buffering there is always a frame available to be sent. So that it is not necessary to wait for the image acquisition to send a new frame. With this method, an intelligent sensor can provide the same performance as the double buffering method, but performing others tasks, as image processing, image segmentation or image resizing.

Currently, we are studying how the buffer used affects to robot navigation. Reactive behaviours require information as quickly as possible, so the triple buffer seems the most appropriate method. However, the immediacy of the double buffer, can improve certain non-critical behaviours as tracking paths, due to they provide latest information, but with less immediacy.

As future work, it is planned to adapt the triple buffer to a system where the obtained data from the intelligent sensor would have several distinct parallel processes, in example, locating free paths to the robot trajectory at the same time that interesting objects to surrounding maps generation are detected like corners, walls or door steps.

**Acknowledgments**: Coordinated project COBAMI: Mission-based Hierarchical Control. Education and Science Department, Spanish Government. CICYT: MICINN: DPI2011-28507-C02-01/02 and project “Real time distributed control systems” of the Support Program for Research and Development 2012 UPV (PAID-06-12)

## 7 References

1. Brignell JE, The future of intelligent sensors: a problem of technology or ethics? Sensors and Actuators A56 (1996) 11-15
2. Fernandes, J. Laranjeira, J. Novais, P. Goreti Marreiros, J. N. A Context Aware Architecture to Support People with Partial Visual Impairments. Distributed Computing and Artificial Intelligence. Advances in Intelligent Systems and Computing Volume 217, 2013, pp 333-340

3. Chien-Hui, L. Kuan-Wei, L. Ting-Hua, C. Che-Chen, C. Charles H.-P. Fall Detection by a SVM-Based Cloud System with Motion Sensors. Advanced Technologies, Embedded and Multimedia for Human-centric Computing. Lecture Notes in Electrical Engineering Volume 260, 2014, pp 37-45
4. Grzejszczak, T. Mikulski, M. Szkodny, T. Jędrasiak, K. Gesture Based Robot Control. Computer Vision and Graphics. Lecture Notes in Computer Science Volume 7594, 2012, pp 407-413
5. Lee, C.S. ; Gyu Myoung Lee ; Woo Seop Rhee. (2013) Standardization and challenges of smart ubiquitous networks in ITU-T. IEEE Communications Magazine, 51(10), 102-110
6. Dan, A. D. Dias R. Mukherjee, D. Sitaram, R. Tewari. Buffering and Caching in Large-Scale Video Servers. Prac. of COMPCON, 1995.
7. Tagami, Y., Watanabe, M., & Yamaguchi, Y. (2013). Development Environment of 3D Graphics Systems. Fujitsu Scientific & Technical Journal, 49(1), 64-70.

**¡Error! No se encuentra el origen de la referencia.**. Khan, S. Bailey, D. Gupta, G. Simulation of Triple Buffer Scheme. 2009 Second International Conference on Computer and Electrical Engineering

8. ASUS: Xtion Pro Live. [Online]. Available: www.asus.com
9. Edwards, C. Not-so-humble raspberry pi gets big ideas. Engineering & Technology. Volume:8 (3). 2013. 30 - 33
10. Poza-Luján, J.L. Posadas-Yagüe, J.L. Simó-Ten, J.E. (2014) Quality of Control and Quality of Service in Mobile Robot Navigation. International Journal of Imaging and Robotics.. Volume 8 (1).
11. Norman Villaroman, N. Rowe, D. Swan, B. 2011. Teaching natural user interaction using OpenNI and the Microsoft Kinect sensor. In Proceedings of the 2011 conference on Information technology education (SIGITE '11). ACM, New York, NY, USA, 227-232.
12. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'reilly.
13. Ollero, A. (2005). Intelligent mobile robot navigation (Vol. 16). Springer.
14. Freese, M., Singh, S., Ozaki, F., Matsuhira, N.: Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator. In:SIMPAR 2010. LNCS, vol. 6472, pp. 51–62. Springer, Heidelberg (2010)