Towards a cloud-based automated surveillance system using wireless technologies

Javier J. Salmerón-García[1] · Sjoerd van den Dries[2] · Fernando Díaz-del-Río[1] · Arturo Morgado-Estevez[3] · Jose Luis Sevillano-Ramos[1] · M. J. G. van de Molengraft[2]

Abstract Cloud Computing can bring multiple benefits for Smart Cities. It permits the easy creation of centralized knowledge bases, thus straightforwardly enabling that multiple embedded systems (such as sensor or control devices) can have a collaborative, shared intelligence. In addition to this, thanks to its vast computing power, complex tasks can be done over low-spec devices just by offloading computation to the cloud, with the additional advantage of saving energy. In this work, cloud's capabilities are exploited to implement and test a cloud-based surveillance system. Using a shared, 3D symbolic world model, different devices have a complete knowledge of all the elements, people and intruders in a certain open area or inside a building. The implementation of a volumetric, 3D, object-oriented, cloud-based world model (including semantic information) is novel as far as we know. Very simple devices (orange Pi) can send RGBD streams (using kinect cameras) to the cloud, where all the processing is distributed and done thanks to its inherent scalability. A proof-of-concept experiment is done in this paper in a testing lab with multiple cameras connected to the cloud with 802.11ac wireless technology. Our results show that this kind of surveillance system is possible currently, and that trends indicate that it can be improved at a short term to produce high performance vigilance system using low-speed devices. In addition, this proof-of-concept claims that many interesting opportunities and challenges arise, for example, when mobile watch robots and fixed cameras would act as a team for carrying out complex collaborative surveillance strategies.

Keywords Surveillance · World modeling · Wireless communication · Computation offloading · RGBD cameras · Cloud robotics

✉ Fernando Díaz-del-Río
fdiaz@us.es

[1] Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, Sevilla, Spain

[2] Technische Universiteit Eindhoven, Eindhoven, The Netherlands

[3] Escuela Superior de Ingeniería, Universidad de Cádiz, Cádiz, Spain

1 Introduction

Nowadays, the development of automated surveillance systems has become more than widespread [7]. There are innumerable environments where their use has become critical, such as cities, households and corporate buildings, not only for security, but also for people location or customer queue analytic [38]. Consequently, there is a need to process, send and analyze the high volume of sensorial media produced by cameras, radars and other sensors. This task involves not only the communication infrastructure but also the necessary multimedia computing framework. More precisely, in a future smart city there would be innumerable devices (small sensors, computers, mobile robots, amongst others) that working together should perfectly create an extremely safe environment. However, several cutting-edge surveillance solutions rely on very expensive sensing technology, high performance dedicated hardware and wired connectivity, and are therefore unaffordable for many companies or scenarios. Hence, the question to be addressed in this paper would be: until what extent can all these heterogeneous devices collaborate to perform a multifaceted task-like surveillance using current low-cost available technologies?

A powerful tool to fulfill this challenge is the incorporation of Cloud Computing, because many of its properties can benefit such surveillance systems. Specifically, the following characteristics fit perfectly:

- On-demand resource provision: The cloud is able to provide both computing and storage resources. This is normally done in the form of the so-called containers. For instance, should a surveillance or "watch" robot require computing power to perform a certain task, then it would only have to spin up as much instances as required to complete it. Most commercial cloud platforms (for example, Amazon EC2, Windows Azure, Google App Engine, etc.) use a pay-as-you-go approach. Nevertheless, this should be much cheaper than all the capital expenditure required for a cluster, not to mention all the maintenance costs.
- Dynamic scalability: This property relates to the ability of the cloud to dynamically adapt to the user needs. A clear example is that of a multiple intrusion. If, suddenly, the number of requests dramatically increase due to this transgression, the service might not be able to process them in a reasonable time. To address this, the cloud could increase the number of computing resources devoted to this task. Moreover, once the threat has ceased and the number of requests decreases, the cloud could reduce the amount of computing resources. This relates to the "utility" concept behind Cloud Computing.

In fact, a great interest exists in applying and developing Cloud Computing frameworks to the surveillance area [17, 26]. However, many challenges still remain. For instance, the incorporation of the cloud paradigm forces developers to rethink their software architectures so, for instance, the dynamic scalability property could be exploited (see Sect. 4). Another major concern is that, as pointed out in [26], the cost for deploying a video surveillance system using cloud computing is not necessarily less expensive than implementing the hardware locally. Furthermore, legal and privacy issues must be addressed when using commercial cloud platforms.

An interesting approach is the use of a private cloud setting, so that critical surveillance data is always kept in the private cloud [17]. This approach also facilitates the integration of embedded devices that can successfully perform collaborative tasks such as localization, mapping [30], traffic-light status detection [4] or healthcare [10]. As a consequence, concepts such as "Internet of Robots" [13, 37] are now possible. Furthermore, a private cloud can be complemented with commercial cloud platforms when needed to optimize deployment and maintenance costs, an approach commonly known as hybrid cloud [17]. Therefore, using the cloud for a collaborative surveillance system is another interesting application that we address and evaluate in this work.

In a great extent, the answer to reduce costs and deploy more easily a vigilance system that integrates a number of embedded devices comes from the use of private/hybrid Cloud Computing and high bandwidth wireless technologies. Moreover, the cloud is the most suitable and cost-effective candidate for what is known as computation offloading. The idea is to free an embedded device from heavy computations, so that an external platform would do them and get the results back. This is especially interesting as it allows the devices to perform more complex tasks, overcoming their hardware limitations and their inherent difficult upgrading. Furthermore, offloading CPU-intensive tasks implies less power consumption in the devices and robots. Following this idea, building very simple devices that rely on computation offloading will set a trend in forthcoming years [31]. On the other hand, the use of high bandwidth wireless technologies is especially interesting for two reasons. First, it permits the use of mobile robots or UAVs (Unmanned Aerial Vehicles), bringing innumerable possibilities in surveillance. Second, the new standard 802.11ac promises bandwidths of the order of 1 Gbps [9]. Hence, eventually, current bottlenecks of WiFi connections could be overcome. Therefore, the convenient combination of Cloud Computing, wireless technologies and heterogeneous embedded devices (depending on the budget) could be a perfect candidate for creating a cost-effective surveillance and intruder detection solutions.

Besides, there is a problem that needs to be addressed to detect intruders: the correct interpretation and sharing of information of the environment. In this sense, the lack of semantic information provokes problems in this kind of tasks, for instance [5, 11, 22]:

- Some operations such as clearing could be done more efficiently if more information of the static environment was known.
- Without symbolic information, it is very difficult for the robot to make predictions based on the environment.

The system presented here is a first prototype of a cloud-based surveillance system using a shared, 3D symbolic world model; simple embedded devices that can offload computation to the cloud; mobile devices including "watch" robots; high-bandwdth wireless connections; and Kinect cameras as the sensing technology. The paper is organized as follows. First, next section presents the system model. In Sect. 3, a brief review of recent advances in the area is presented. Then, in Sect. 4 all the main elements of the prototype (world model, information storage, computation offloading) are described. To prove the efficacy of this

first prototype, some preliminary experimental results are included (especially focused on communication and computation performance) in Sect. 5. Finally, future lines of work and conclusions are presented in Sects. 6 and 7.

2 System model

In this paper, we propose a framework for giving support to a surveillance system via an Object Oriented World Model that is distributed among the cloud and the rest of devices. We use a 3D volumetric, object-oriented world model, called Environment Descriptor (ED), (developed by the "Control Systems Technology" in TU/e University [5, 11]) that allows the modules in the system to take the context of the environment. Such a fully offloaded cloud-based implementation of a volumetric, 3D, object-oriented world model (including semantic information) has never been carried out as far as we know.

Figure 1 shows an application scenario of the proposed Cloud-based World Model platform. Surveillance tasks can be shared by several fixed devices with different hardware configurations and sensing technologies: stereo cameras, RGBD cameras, range finders, laser, amongst others. In addition, each device can be assigned to different tasks. The common case would be that of camera devices installed in different places. Meanwhile, in addition to this, there can be mobile "Watch robots", which navigate the area or building looking for intruders. Then, as they all share the same world model, if a camera detected an intruder, then all devices would know their exact location. Consequently, "Watch robots" would transform their location coordinates to relate their own posture to that of the intruder, with the aim of pursuing him/her.

Figure 2 shows a conceptual diagram of the proposed Cloud-based World Model platform. In our implementation, task organization is flexible enough so that, depending on the hardware specifications, some devices can perform all the computations on their own (Case 2) or offload computations to the cloud (Case 1). Non-sensing devices that form part of the system such as domotic controllers can receive an order and react accordingly, for instance, by locking a door if an intruder were found (Case 3). Hence, using the cloud paradigm as a natural central interface for many devices, an extremely efficient and cooperative surveillance system can be developed. A detailed explanation of the blocks of this Fig. 2 will be described in further sections.

It is worth noting that in this kind of applications, concerns such as response time requirements or data safety and privacy may lead to an alternative paradigm to Cloud Computing called Edge Computing [34]. This term refers to processing the data at the edge of the network, that is, closer to the "things", in the IoT sense, or end devices. As mentioned before, in our experiments we will use a private cloud which could be considered as a kind of edge-computing system. The reason is that this way we avoid longer and more variable delays and can focus on implementation issues such as computation offloading or environment sensing and modeling. However, along the paper we prefer to use the more general model and terminology of Cloud Computing. Furthermore, the extension of our surveillance system to a larger Smart City would benefit from a more general Cloud Computing paradigm, for instance, gathering sensing data from different buildings.

3 Related work

There are numerous research works in the area of automated surveillance. In relation with this work, the most relevant can be arranged in those works that use the cloud for integrating sensor information, those that deal with the detection algorithms, those that test several sensor devices, and finally, others that study network technologies.

The idea of using the cloud for integrating multiple sensor information, including surveillance information, has also been explored in the literature. Moreover, a new term has been coined to define the application of Cloud Computing (in terms of ubiquity, storage, computation, privacy, scalability, amongst others) to this area: Video Surveillance as a Service [20, 29]. For instance, in [27] the authors deal with the problem of integrating heterogeneous sensor information by creating a database scheme able to integrate different types of sensor. However, most of the available solutions rely on wired networks, such as [14, 15, 21, 36]. Only a few works have developed some aspects of cloud-based surveillance platform using wireless networks. Authors of [38] presented Vigil, where some devices with cameras execute face recognition algorithms and send the results to the cloud. They also introduce the concept of intra-cluster processing, when there is a set of cameras with overlapping visions. Compared with Vigil, the work presented in this paper takes into consideration the use of very simple devices thanks to computation offloading. In [35] a Cloud Gateway for filtering the information from multiple sensors in Wireless Sensor Networks was implemented. This filtering is applied to a basic cloud surveillance system obtaining an 83% accuracy.

Most of the cloud-based surveillance systems have the following architecture: multiple cameras stream to the cloud, which is responsible of analyzing the footage in search of events. Only in the case of [14] some pre-processing is done before sending to the cloud. Compared to these works, this paper offers a more flexible approach, as

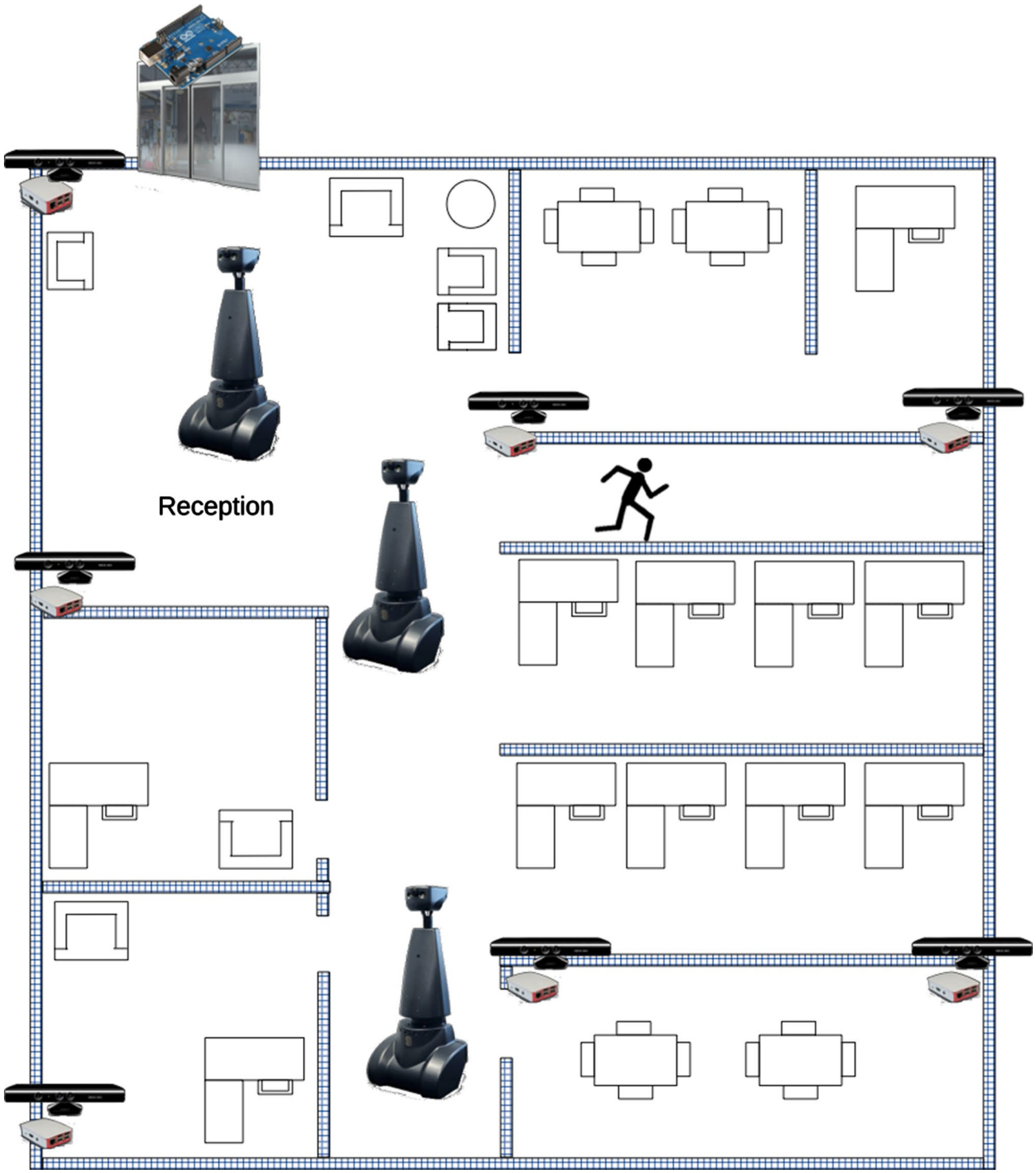


Fig. 1 Example of application in a building

the devices can decide whether to offload or not depending on their needs.

Another important element of Cloud Computing is Quality-of-Service (QoS) management. In this sense, in [3] a service QoS-adaptive configuration framework for optimizing this factor in surveillance systems is proposed.

Another study element corresponds to image analysis to detect intruders. Several algorithms have been devised, such as Latent Semantic Analysis (LSA), Kernel-based

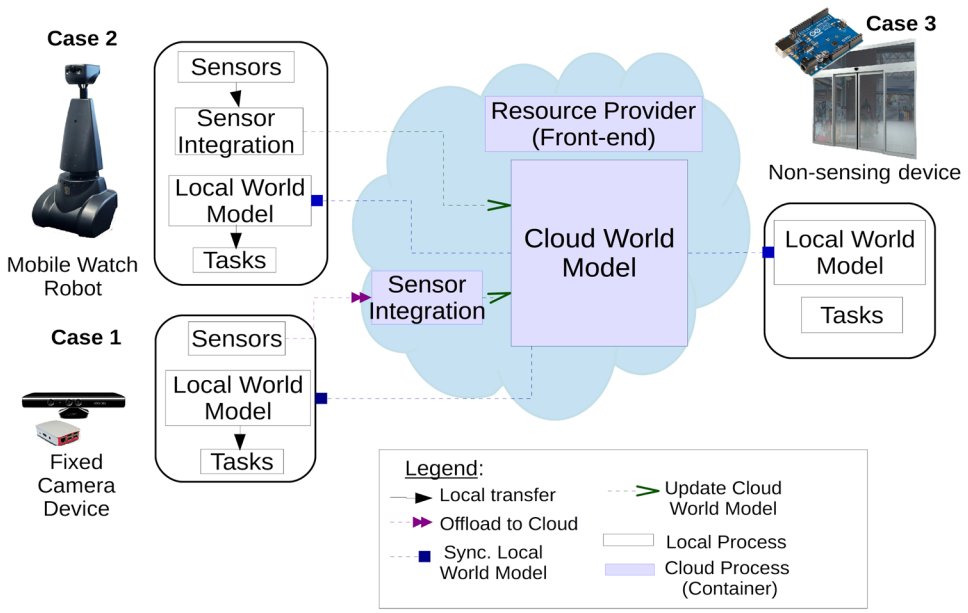


Fig. 2 Conceptual diagram of the cloud-based surveillance system

Online Anomaly Detection (KOAD) and Kernel Estimation-based Anomaly Detection (KEAD), Kernel Principal Component Analysis (KPCA) and One-Class Neighbor Machine (OCNM) [2], using machine learning techniques as well.

In [25], instead of regular video cameras, omnidirectional cameras are used, which permit the surveillance of a whole room or a bigger open space with only one camera. In [12], the sensor used is that of Microsoft Kinect, with satisfactory results. In this sense, the system presented here also makes use of Kinect Sensors which, at a reasonable price, provide a depth channel, extremely important for object detection and tracking.

As it can be seen, most of the surveillance systems available in the literature normally use fixed cameras. In this sense, one of the major assets of this work is the collaboration of different heterogeneous devices (such as mobile "watch" robots) thanks to the use of a shared world model and the adaptation of the cloud to each robot's needs. Moreover, the use of mobile robots brings some interesting opportunities and challenges for more complex surveillance strategies and systems.

The approach of using an edge-computing platform and wireless technologies is becoming increasingly attractive to researchers on surveillance systems [6]. For instance, [16] shows that an edge node can reduce the reaction time between edge and sensors, and can improve network stability as well by uploading only necessary data. Also, [23] improves the optimal bandwidth distribution and allocation in a video surveillance system using up-to-date wireless technologies. In the present paper, we follow this approach with the additional contributions of implementing a 3D, object-oriented, cloud-based world model that includes semantic information, as well as allowing the use of simple embedded devices and mobile "watch" robots through computation offloading.

4 Implementation of the platform

In this section, the implementation of the Cloud World Model based Surveillance Platform is detailed. The implementation of the Surveillance Platform consists of the following parts:

- Definition of the elements of World Model (Sect. 4.1)
- Definition and deployment of a World Model Storage System (Sect. 4.2)
- Implementation of the Sensor Integration Module, responsible of processing the image streams (Sect. 4.3).

4.1 Elements of the world model

World modeling is the process of creating a model of an environment (with its inherent complexity), which is a key task for intelligent systems. The more accurate the representation of the environment, the more successfully a task will be performed. Hence, as stated in Sect. 1, adding semantic information to each element of the environment implies a dramatic increase in accuracy. In the implementation shown in this paper, the main elements of the world model (simplifying from [5, 11]) are entities and transforms.

Entities: An entity is an element of the world. It has the following properties:

- ID: Unique identifier for an entity. No other entity can have the same identifier, even if the entity was removed.
- Type: Specifies the type of the entity. This allows to make classifications or create subsets of entities for different actions. For example, several actions can be done depending if the entity is a person, a furniture or a wall.
- Pose: Establishes the position and orientation of the entity in the world model. These coordinates are absolute.
- Shape or convex hull: Adds volumetric information to the entity. This shape can either be real or virtual. An entity can have both.
- Measurements: Information obtained by sensors that is associated to an entity in the world model. These measurements hold the sensor data and the time-stamp.

Transforms: defines the spatial relationship between two different entities in the world model. Multiple transformations for each parent–child pair can be stored for different timestamps which results in a graph of entities and transforms.

4.2 Cloud world model storage

As stated in previous sections, a collaboratively built world model offers several advantages. However, there are some bottlenecks that the cloud solution must address.

First, if the number of robots increases (and therefore, the number of Database read/write operations), the cloud might not be able to satisfy all queries in a reasonable time. To make matters worse, if the number of entities increases, the cloud resources assigned to the database may run out. Therefore, one of the main objectives of the cloud solution is to adapt computing resources at runtime depending on the needs.

To satisfy these requirements, Distributed Database Management Systems (DDBMS) are the perfect candidates for storing all the information, thanks to their massively parallel nature, ease of use and portability. Among all the available options, one of the best DDBMS paradigms is that of Bigtable. Its main advantage is that it uses a sparse, distributed and multidimensional sorted map, using the following data organization [8]:

> (row:string, column:string, tag:string, time:int64) → stringHence, there would only be one (or two) big tables that can be perfectly scattered among different containers in the cloud. Therefore, if the aforementioned bottlenecks occur, then more containers can be spun up at runtime. On the contrary, if the database is less loaded, then the cloud can scale back, thus saving resources for other tasks.

Having in mind this DB model, the next schema was designed to fit with the needs of a surveillance system. The idea is not only to store the current information about entities, but also their history. In this sense, we use the concept of *delta*, that is, a change in a certain entity (following the so-called event-driven mechanism [17]). To clarify the concept, an example follows (depicted in Fig. 3): suppose two robots A, B and an entity E. Both robots have synchronized their local world models at an instant t, finding that the most recent change was done at $t-1$ ("Last time" in Fig. 3). Therefore, they know all the information about E (shape, pose, type, etc.). At $t+1$, B registers changes in the pose of E, and therefore will update the world model with this new information. If nothing but the pose has changed, re-sending the information about the shape would be a waste of bandwidth. Therefore, B will only send the change (that is, the delta) of the pose. This same reasoning can be applied to A, which is only interested in the most recent changes (deltas) since database's most recent time-stamp $t-1$ (to save resources). Thanks to the use of timestamps, a robot can query all the new deltas since a given time t.

Taking this into account, there will be a table of deltas in the database with the following columns: pose, type, shape, convex hull and deleted (whether the entity was cleared or not). The information will be stored in the JSON (JavaScript Object Notation) format. The row ID will correspond to the entity's ID.

Among the available implementations of the Bigtable model, Hypertable has been chosen because it is written on C++ and promises higher performance results compared to competitors such as HBase [1]. This is of great importance when working with robots, and in some cases, near real-time conditions. The elements of a Hypertable system that were configured for our system are [19]: range server (for handling the read and write operations in the table), Master node (for creating/deleting tables), ThriftBroker (interface between the database and the clients) and FS Broker (normalized filesystem interface).

Figure 4 on the left shows the implementation of the world model using Hypertable (the FS Broker is not shown, as it is being used in all the nodes). On the right, an example of adaptation when a bigger number of robots demands more resources has been depicted. In this case, more instances of Range servers and ThriftBrokers are spin up by the cloud to satisfy this new load.

4.3 Sensor integration modules

The sensor integration offloading implemented here is a simplified version of [5, 11]. The workflow is depicted in Fig. 5. All modules are implemented using the Robotics Operating System (ROS) Framework. ROS is an Open

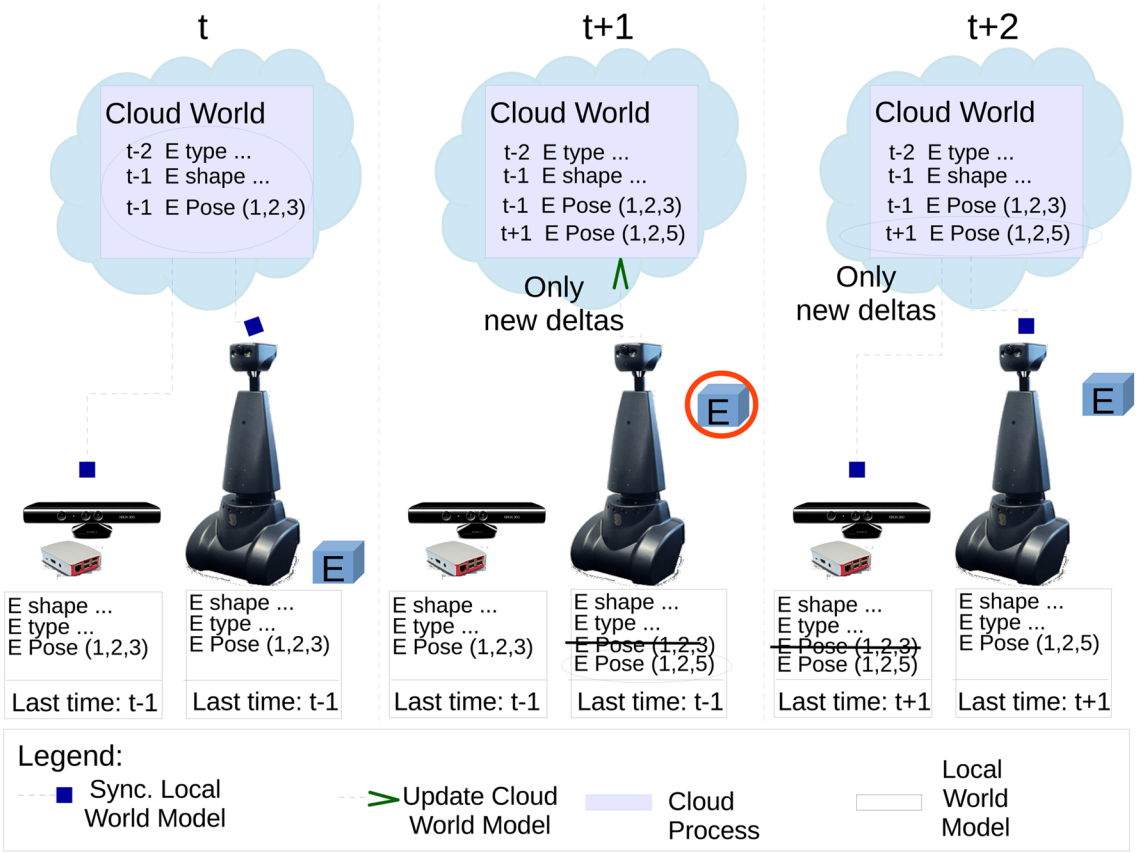
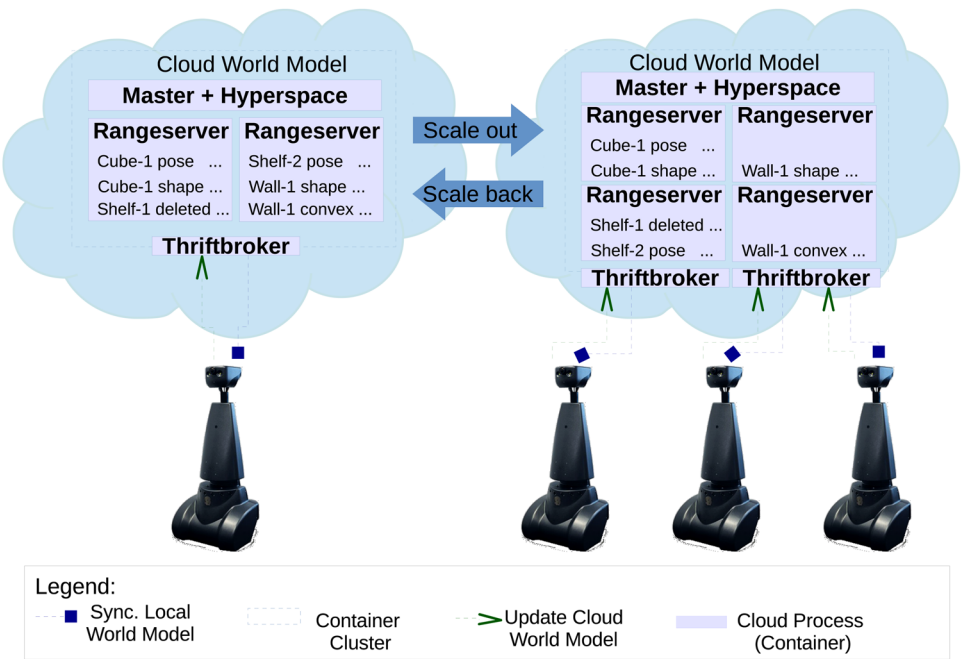


Fig. 3 Example of a delta update

Fig. 4 Implementation of the cloud World model using hypertable



Source Robotics Software Framework (using C++ and Python) that allows the implementation of advanced distributed architectures, very similar to Multi-Agent systems [18]. Thanks to the use of topics, services, peer-to-peer communications and compression techniques, ROS makes the offloading process simple and scalable. Main processes (nodes in ROS convention) correspond to those modules of Fig. 5, while connecting links of these modules have been implemented using ROS topics and services.

- The different operations done in the sensor integration are the following:RGBD Masking: This process will only use the D channel of the stream. It is responsible for detecting entities using feature extraction (producing 3D Point Clouds), association (comparing with a rendered world model) and segmentation (creating new entities from the unassociated data) [5, 11]. Figure 6 shows an example. This module is one of the most computationally demanding. In our case, as we are dealing with surveillance cases, this module will discard those entities whose size is not big enough to be a person.
- Sensor fusion: This process receives the RGBD stream and the aforementioned entities with mask measures. The idea is to use the previously unused RGB channel and extract other properties from the entities. Therefore, the module must buffer the RGBD stream until its related masked measurement arrives. Figure 7, using the mask from Fig. 6, now obtains an RGBD measurement (which, for example, contains color information).
- Perception: This process receives the RGBD measurements (created by the Sensor Fusion module) and performs an analysis to discover the type of the entity. This will be the responsible module for detecting whether an entity is an intruder or not. As our aim is demonstrating the cloud working and feasibility, in the experiments presented in this paper the system only detects whether the entity is wearing or not an uniform. If the person was not wearing it, then it would not be an authorized personnel. On commercial surveillance systems, it is obvious that a knowledge base (with templates) must be used. For the previous example, Fig. 8 analyzes the RGBD measurement and concludes that the entity is an authorized person.

The presented platform is aimed to be as much flexible as possible. In this sense, a robot may not have enough processing power to do all the sensor integration on-board but sufficient to do part of this processing. Therefore, the robot should be able to choose which parts of the sensor integration are offloaded. To implement this feature, the system was implemented using modules. For instance, one instance may only have the Perception module (and thus

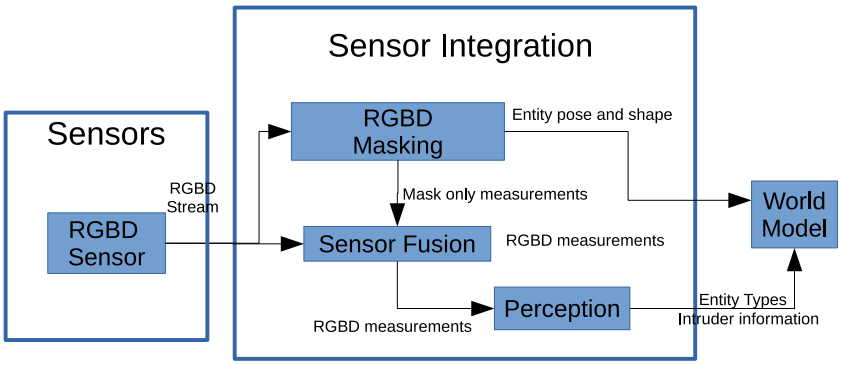


Fig. 5 Workflow of the Sensor Integration model

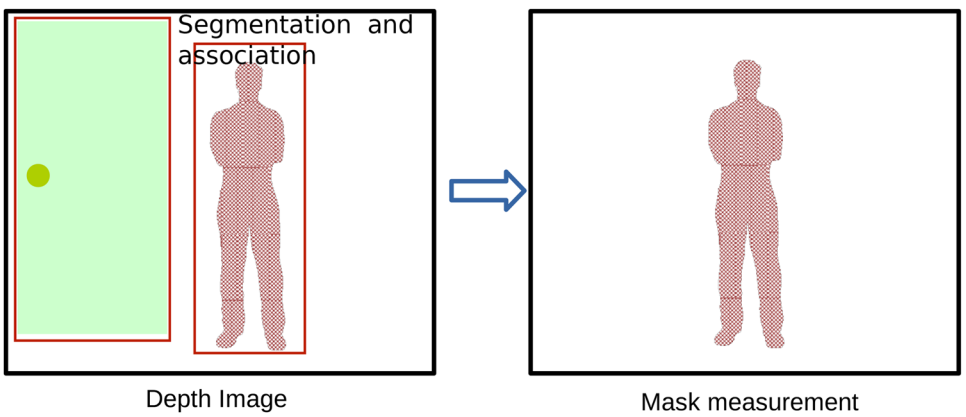


Fig. 6 Example of a mask measurement

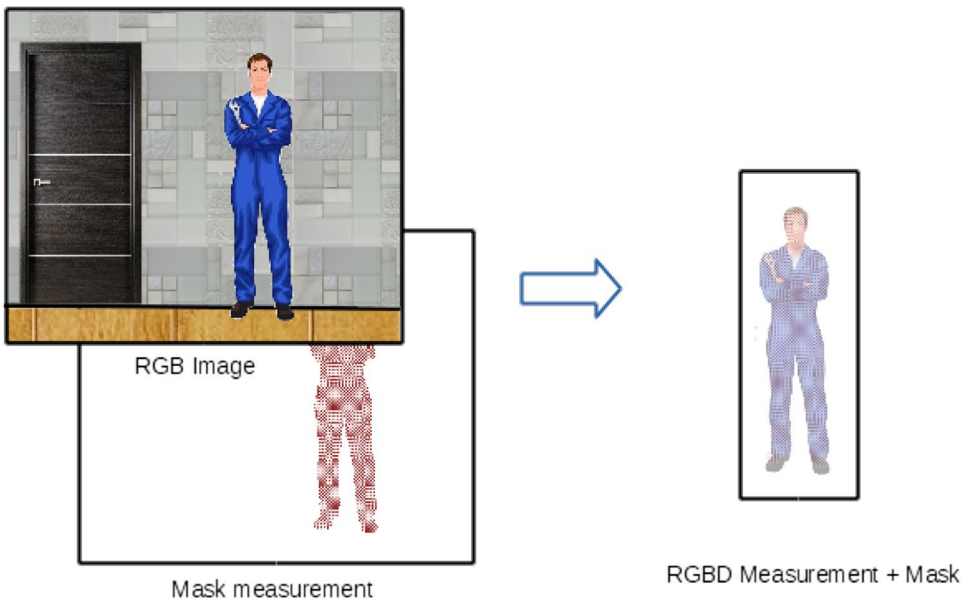


Fig. 7 Example of a RGBD measurement

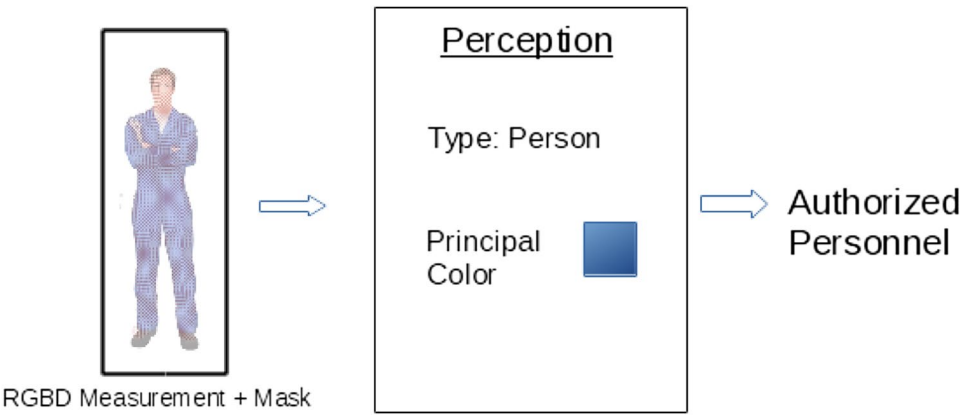


Fig. 8 Example of perception

only perform recognition operations), whereas another node may have all modules. As a consequence, several offloading options can be taken into consideration.

5 Preliminary results

These preliminary tests are aimed to show the efficacy and viability of not only having a shared world model, but also of using wireless technologies. The tests are divided into three categories:

1. Real experiment in a testing environment.
2. Queries in the world model.
3. Sensor integration offloading.

All the tests were done in a private Cloud with 5 nodes (1 front-end node and 4 computing nodes). Each node has a AMD FX-8320 octa-core CPU (with virtual extensions enabled) and 8 GB of RAM. They are all connected internally using Gigabit Ethernet bandwidth. For the cloud middleware, we decided to make use of Kubernetes + Docker (recently adopted by major cloud vendors, such as Google Container Engine or Amazon EC2 Container Engine) instead of traditional VM-based private clouds (such as Openstack or Eucalyptus). One of the main reasons behind this decision was to avoid the overheads derived from the virtualization of hardware and operating system layers [33].

5.1 Proof-of-concept experiment

A few preliminary experiments were devised to check whether the whole system works properly. For this purpose, only a few set of objects were modeled for an indoor environment; mainly rough shapes of the furniture and two people (one intruder and one authorized person). In one of the offices, two camera devices were installed to detect intruders

in the room. Basic experiments were based on the following facts:

1. Authorized personnel enter the room, move around and leave.
2. An intruder enters the room and moves around the room.

First of all, this experiment aims to show that both intruder detection (perception module) and person tracking (RGBD masking module) work successfully. Second and the most importantly, this experiment will demonstrate that the collaboration between devices is actually working. Let us suppose that an entity E goes from camera A's Field of View (FOV) to camera B's FOV. If the collaboration is effective, it should be detected that the entity that suddenly enters in camera B's area is actually E and not a new entity E'.

Prior to performing the experiment, the room was previously modeled (see Fig. 9). To avoid detection of unwanted items (such as the elements in a table), some overmodeling was done. A video of the experiment can be found in [32]. In terms of the desired objectives, the following can be stated (Fig. 10):

– Both person tracking and intruder detection algorithms worked correctly with different people and outfits.
– Thanks to the shared world model, collaborative tracking of people was done successfully. When the intruder left camera A's FOV and entered camera B's domain, it was possible to associate the entity correctly.

While it is still a first experiment, the viability of a WiFi 802.11ac, cloud-based, collaborative intruder detection system using an object-oriented world model has been thoroughly shown. However, the number of entities and their modeling accuracy should be raised in future works to guarantee the robustness of the detection algorithms for broader areas. The rest of subsections are aimed to demonstrate until which extent a wifi ac cloud-based surveillance system is able to respond when it is overloaded.

5.2 World model querying

First, I/O operations in the shared world model are performed with different data loads and network technologies. Our main interest does not reside in the capabilities of the Cloud Database. Instead, the wireless technology is the key factor, as it would demonstrate the validity of the wireless-based surveillance system.

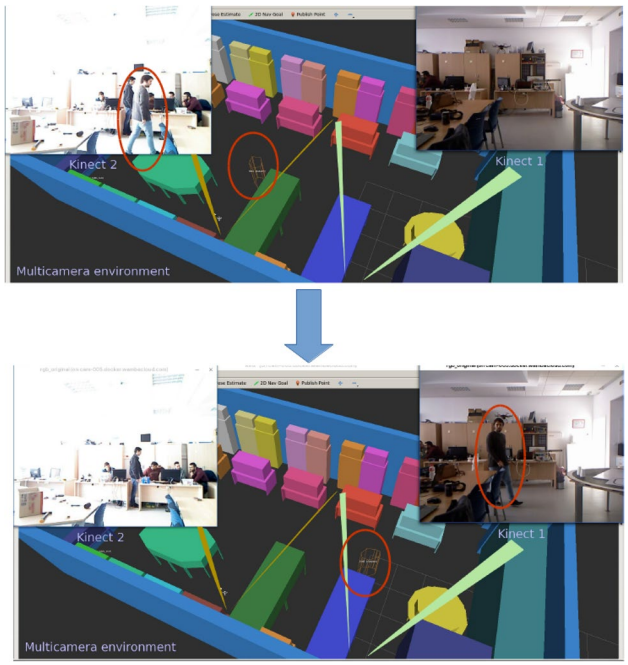


Fig. 10 Transition of the entity E (*red circle*) from one area to another. For the sake of simplicity, only the interested entity is shown (color figure online)

For this set of tests, watch robots will be used, assuming that they have sufficient CPU power to produce delta update information at either 1, 5, 10 and 20 Hz rates. To do so, the sensor integration part is skipped here and simulated data are used instead.

5.2.1 *Communication Technology Test*

In this first test, data transfer time for 802.11ac and Fast Ethernet will be compared for different data loads (entity delta updates).

Object-oriented, 3D world models present the advantage that objects contain symbolic information since they are instances of elements in an object database [5]. Other object specific information such as trajectories, properties and affordances can also be stored in this database, which

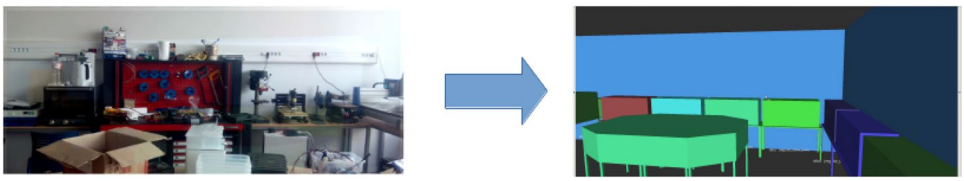


Fig. 9 Modeling of the testing environment

can be used to infer the nature of the detected objects. In our case, delta updates mainly consists of sending a timestamp, the tag of the object and its six posture coordinates (3 for position and 3 for orientation). Using single precision floating format, this supposes a few dozen of bytes (much less than the packet overhead of any message in current network protocols).

Sample results can be seen in Table 1. In this test, Fast Ethernet performs an average of two times faster than 802.11ac. Note that delta updates of a high amount of entities per second are highly unlikely for real surveillance environments. For instance, let us imagine that there were five intruders moving at speeds of around 3 m/s. If we wanted to approximate with an error of $\pm$ 0.5 m, we need to send $5 * (3/0.5) = 30$ delta updates per second. According to Table 1, this would take around 9 ms using WiFi, which is much inferior than the times required to offload the sensor integration (see Sect. 5.3). In conclusion, it can be guaranteed that delta update transfer times are assumable by wireless technologies.

Table 1 Average delta transfer time

Deltas	Technology	
	WiFi AC	Ethernet
1	0.0045	0.0014
16	0.0062	0.0029
32	0.0091	0.0048
64	0.021	0.0078
128	0.023	0.015
256	0.058	0.029
512	0.155	0.056
1024	0.167	0.156
2048	0.348	0.228

Two techonologies are compared: WiFi AC and Fast Ethernet. The frequency used is 1 Hz

5.2.2 Stability test

A watch robot will produce different data loads in the shared world model at different frequencies. This is closely related to previously explained communication test, as the data transfer speed plays a major role. If the communication technology is not fast enough, the robot would have to store in an intermediate buffer all the information to be written. In principle, this would not be a problem, unless the amount of buffered information becomes too big and the robot's memory runs out. In this case, the software would end up crashing, unless some buffered information is discarded. In this sense, a desirable robot configuration (that is, frequency and amount of data) would be one in which both tasks (information production and data transfer) remain stable. Because of this, in this test we will compare the stable configurations available when using 802.11ac and Fast Ethernet.Table 2a and b show the obtained results for 802.11ac and Fast Ethernet. Results are very similar for both networks. The robot can cope with only two extra workloads (that of 256 delta updates at 10 Hz and 128 delta updates at 20 Hz) in the case of Fast Ethernet. This makes sense as the previous test showed that Fast Ethernet performs around twice as fast as WiFi AC. However, it must be noted that 802.11ac is stable for workloads sufficient enough for most surveillance cases. It would be highly unlikely that a single camera or watch robot can produce, for instance, an average of 512 person warnings at 5 Hz. Therefore, it has been demonstrated that wireless technologies are usable for cases where a high update frequency is required.

Table 2 Stability of the system for different frequencies and data loads

(a) Stability for WiFi AC

WiFi AC				
	Update frequency			
Deltas	1	5	10	20
1	Y	Y	Y	Y
16	Y	Y	Y	Y
32	Y	Y	Y	Y
64	Y	Y	Y	Y
128	Y	Y	Y	N
256	Y	Y	N	N
512	Y	Y	N	N
1024	Y	N	N	N
2048	Y	N	N	N

(b) Stability for Fast Ethernet

Fast ethernet				
	Update frequency			
Deltas	1	5	10	20
1	Y	Y	Y	Y
16	Y	Y	Y	Y
32	Y	Y	Y	Y
64	Y	Y	Y	Y
128	Y	Y	Y	Y
256	Y	Y	Y	N
512	Y	Y	N	N
1024	Y	N	N	N
2048	Y	N	N	N

"Y" stands for stable and "N" stands for not stable

5.2.3 *Interference test*

The added value of our proposal is the collaboration between multiple robots by sharing information about entities. Because of this, it is highly likely that multiple embedded systems share the same Access Point (AP), and therefore collisions and interferences can occur. If the impact in the transfer time were high, then the stability of the system could be compromised (as shown in the previous experiment). In this test, the number of robots will be increased, and by measuring transfer times, the impact in the communications will be analyzed.

Table 3 shows sample transfer times when increasing the number of nodes using a fixed data load (128 delta updates). To detach as much as possible these results from possible performance issues in Hypertable, each robot has its own database in the cloud for two frequencies: 5 and 10 Hz.

Two conclusions can be extracted from this experiment. The first one is that adding more robots seems to add performance penalties in the communications (mainly for higher frequencies). As a result, the distribution of robots in multiple APs should be considered. In addition to this, the higher the frequency, the higher the number of transfers, and consequently higher penalties appear. As a conclusion, there is a tradeoff-between performance and number of robots. For instance, in the case of Table 2a, it can be seen that a data load of 128 delta updates at 10Hz was at the frontier between stable and unstable. As the configuration may become potentially unstable when more robots are added, a practical system should evaluate and choose the frequency of the robots as a function of the number of them.

5.3 Sensor integration offloading

The efficacy of computation offloading for embedded systems with low resources or legacy hardware must be assessed. To do so, we compare the two sensor cases of Fig. 2, that is, case 2, where a simple embedded system with a Kinect sensor sends its RGBD stream to the cloud, against case 1, in which the sensor integration module is computed on the embedded system, thus sending only the world delta updates inferred by the device. The embedded system used for testing is an Orange Pi PC, which is a $15 SoC with a 4-core processor (H3 Cortex-A7) and 1 GB of RAM.

The performance between the offloading case 1 opposed to executing everything on-board (case 2) are compared in Table 4. First of all, if we compare the results from Sect. 5.2.1, it can be seen that the delta update transfer times can be considered negligible compared to the sensor integration part.

On one hand, timing can be clearly divided into two main components for the offloading case: the transfer time penalties of sending the complete RGBD data stream (see Fig. 5), and the Frame Processing Time, which includes all the processing of the sensor integration module plus the necessary world delta updates (Fig. 3). The mean times per frame for a test composed of 2048 RGBD frames are shown on the last column of Table 4. The total mean time points out that the offloading case can reach up to more than two updates per second.

On the other hand, timing for case 2 can be divided into an internal transfer time penalty due to the inter-process communication between the Kinect driver and the sensor integration, and the local Frame Processing Times executed in the Orange Pi device. The mean times per frame for the same test are revealed on the middle column of Table 4. In this case, the embedded case can only achieve less than one update per second in the mean.

The relative high transfer time penalties in the Orange Pi arises, not only for the inter-process communication, but also due to the high computation loads that the system must support because of the complex processing taking place. In fact, this mean penalty is almost the same as the communication cost of sending the RGBD frame to the cloud. The main result is that a Cloud-Based Automated Surveillance System using Wireless Technologies is not only currently feasible, but it even overtakes the performance of simple standalone embedded devices.

This demonstrates that not only some simple embedded devices are unable to perform complex computations

Table 3 Communication time (in seconds) when increasing the number of robots and frequency. The data load is 128 entity deltas.

	Update frequency	
Nodes	EFEFEF5 Hz	EFEFEF10 Hz
1	0.026	0.028
2	0.028	0.048
4	0.038	0.067
6	0.057	0.173

Table 4 RGBD Processing time (in s) with and without cloud offloading

	On-board (Case 2)	Cloud offloading (Case 1)
Frame processing time	0.887	0.133
Transfer time	0.223	0.322
Total	1.11	0.455

on their own, but also their internal transaction times are of similar order to the transmission network times. Taking these facts, it is clear that thanks to using Cloud Offloading the capabilities of embedded devices can be extended as theoretical studies are exposing [31].

6 Future work

This paper sets up a promising proof-of-concept platform of a cloud-based intruder detection system. For future iterations of the prototype, there is a set of challenges that needs to be analyzed and addressed.

First of all, preliminary 802.11ac performance results were far from its theoretical 867 Mbps. It could be seen that Fast Ethernet obtained around double the transfer speed, and therefore the latter allowed more stable configurations (see Table 2). More experiments are needed to evaluate the viability of using wireless links in this kind of applications. A more ambitious system may be devoted to monitor thousands of people or hundreds of high speed moving entities, which would compromise the scalability for both wired LAN and wireless networks. In the case of wireless networks, further improvements in *802.11ac* MAC layer such as TDMA protocols, would be suitable for reducing variability [24] and thus guaranteeing Quality of Service. Moreover, Contention Free Period in the infrastructure mode with fixed size packets could guarantee a minimum bandwidth reservation, which may be suitable for high frequencies such as 20Hz. In addition to this, scheduling mechanisms such as those described in [38] could be useful.

Second, in the experiment detailed in Sect. 5.1, the cameras' field of view were completely disjointed. Special focus must be done when two cameras are pointing at the same entity, as some conflicts could occur. While having the same field of view may not be useful for fixed camera devices, this could be especially important in the case of watch robots.

One practical feature is that of fault tolerance, system reliability and recovery mechanisms. In such a complex system, many troubles can take place: failed delta updates, camera or robot failures, misleading delta timing, etc. Consequently, a satisfactory error handling, the ability of maintaining a sufficient level of functionality and a graceful return to an older correct state should be implemented. There are other modules that must be added or improved to achieve an operational cloud surveillance system at a Smart City level. However, these can be smoothly integrated as new distributed modules. An example of a new module to be developed is that of a central planner. This would reside in the cloud and would manage the whole surveillance system with several tasks: alarming the rest of the robots (and humans) of an intrusion, deciding an actuation strategy, etc. In addition, face recognition algorithms would improve substantially not only the intruder detection, but also person tracking.

Another open issue is how this system would behave if it were implemented in a more standard cloud system such as Amazon Web Services (AWS) or Microsoft Azure. We guess that an hybrid system would be required, with some tasks running at the edge of the network (for instance, in a private cloud like the one described in this paper) and others being moved to a pay-as-you-go commercial cloud platform. In particular, the high-level tasks associated with complex surveillance systems for Smart Cities would require a pure cloud implementation.

7 Conclusions

In this paper, a cloud-based, collaborative, world-model based surveillance prototype is presented. To the author's knowledge, this is the first implementation of a volumetric, 3D, object-oriented, cloud-based world model (including semantic information). Furthermore, the proposal of using simple embedded devices and mobile "watch" robots that can offload computation to the cloud is also novel in the field of surveillance. From the experiments performed with our prototype, several conclusions can be extracted: To begin with, the use of an object-oriented world model allowed the easy collaboration between multiple devices. This allows for all of them to have a complete view of a room (as seen in the proof-of-concept experiment) or, eventually, of a complete building or neighborhood in Smart Cities. The cloud eases the implementation of a central intelligence that will allow the coordination and realization of more complex tasks and surveillance strategies. In addition to this, the ability of the cloud to scale up and back compute resources does really help to deal with innumerable scenarios.

Second, the use of Cloud Offloading makes the use of small embedded devices possible. Thanks to the vast computing power that the cloud offers, it is possible to have very simple devices with just a camera and a wireless connection. On the other side, more powerful devices can be smoothly integrated so that they perform all the complex calculations on their own. In this sense, with our flexible offloading approach a wide variety of heterogeneous setups is more than possible. A step further could be taken and, apart of having a centralized Cloud System for offloading, alternatives such as P2P Cloud could be studied and combined [28]. 802.11ac could perfectly work for several configurations, allowing even the use of mobile "watch" robots, although protocols and software layers still need more development so as to reach the theoretical bandwidth and to reduce and bound latencies.

Acknowledgements The work shown in this paper has been supported by the Spanish grant (supported by the Ministerio de Economía y Competitividad and the European Regional Development Fund) COFNET (Event-based Cognitive Visual and Auditory Sensory Fusion, TEC2016-77785-P) and by Andalusian Regional Excellence Research Project grant (with support from the European Regional Development Fund) MINERVA (P12-TIC-1300).

References

1. Why Hypertable? | Hypertable-Big Data. Big Performance. URL http://hypertable.com/why_hypertable/
2. Ahmed, T., Pathan, A.S., Ahmed, S.: Adaptive algorithms for automated intruder detection in surveillance networks. In: 2014 International Conference on Advances in Computing, Communications and Informatics ICACCI, pp. 2775–2780 (2014). doi:10.1109/ICACCI.2014.6968617
3. Alamri, A., Hossain, M.S., Almogren, A., Hassan, M.M., Alnafjan, K., Zakariah, M., Seyam, L., Alghamdi, A.: QoS-adaptive service configuration framework for cloud-assisted video surveillance systems. Multimedia Tools and Applications pp. 1–16 (2015). doi:10.1007/s11042-015-3074-7. http://0-link.springer.com.fama.us.es/article/10.1007/s11042-015-3074-7
4. Angin, P., Bhargava, B., Helal, S.: A Mobile-Cloud Collaborative Traffic Lights Detector for Blind Navigation. In: 2010 Eleventh International Conference on Mobile Data Management (MDM), pp. 396–401 (2010). doi:10.1109/MDM.2010.71
5. Appeldoom, R.: A volumetric object-oriented world model applied in robot navigation. Master Thesis, Eindhoven University of Technology, Eindhoven (2014)
6. Kim, B., Bhaskar, K.P.: Special section on emerging multimedia technology for smart surveillance system with iot environment. J. Supercomput. **73**(3), 923–925 (2017). doi:10.1007/s11227-016-1939-9
7. Ben Hamida, A., Koubaa, M., Ben Amar, C., Nicolas, H.: Toward scalable application-oriented video surveillance systems. Sci. Inf. Conf. (SAI) **2014**, 384–388 (2014). doi:10.1109/SAI.2014.6918215
8. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: A Distributed Storage System for Structured Data. ACM Trans. Comput. Syst. 26(2), 4:1–4:26 (2008). doi:10.1145/1365815.1365816
9. Charfi, E., Chaari, L., Kamoun, L.: PHY/MAC enhancements and QoS mechanisms for very high throughput WLANs: a survey. IEEE Commun. Surveys Tutor. **15**(4), 1714–1735 (2013). doi:10.1109/SURV.2013.013013.00084
10. Dogmus, Z., Papantoniou, A., Kilinc, M., Yildirim, S., Erdem, E., Patoglu, V.: Rehabilitation robotics ontology on the cloud. In: 2013 IEEE International Conference on Rehabilitation Robotics (ICORR), pp. 1–6 (2013). doi:10.1109/ICORR.2013.6650415
11. Elfring, J., van den Dries, S., van de Molengraft, M.J.G., Steinbuch, M.: Semantic world modeling using probabilistic multiple hypothesis anchoring. Robotics and Autonomous Systems **61**(2), 95–105 (2013). doi:10.1016/j.robot.2012.11.005, http://www.sciencedirect.com/science/article/pii/S0921889012002163
12. Ghose, A., Chakravarty, K., Agrawal, A.K., Ahmed, N.: Unobtrusive Indoor Surveillance of Patients at Home Using Multiple Kinect Sensors. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, SenSys '13, pp. 40:1–40:2. ACM, New York, NY, USA (2013). doi:10.1145/2517351.2517412
13. Guizzo, E.: Robots with their heads in the clouds. IEEE Spectrum **48**(3), 16–18 (2011). doi:10.1109/MSPEC.2011.5719709
14. Hamida, A.B., Koubaa, M., Nicolas, H., Amar, C.B.: Video surveillance system based on a scalable application-oriented architecture. Multimedia Tools and Applications pp. 1–27 (2015). doi:10.1007/s11042-015-2987-5, http://0-link.springer.com.fama.us.es/article/10.1007/s11042-015-2987-5
15. Hassan, M., Hossain, M., Al-Qurishi, M.: Cloud-based mobile IPTV terminal for video surveillance. In: 2014 16th International Conference on Advanced Communication Technology (ICACT), pp. 876–880 (2014). doi:10.1109/ICACT.2014.6779086
16. Park, H.D.: Scalable architecture for an automated surveillance system using edge computing. J. Supercomput. **73**(3), 926 (2017). doi:10.1007/s11227-016-1750-7
17. Hossain, M.A.: Framework for a cloud-based multimedia surveillance system. International Journal of Distributed Sensor Networks **10**(5), 135,257 (2014). doi:10.1155/2014/135257
18. Iigo-Blasco, P., Diaz-del Rio, F., Romero-Ternero, M.C., Cagigas-Muiz, D., Vicente-Diaz, S.: Robotics software frameworks for multi-agent robotic systems development. Robot. Auton. Syst. **60**(6), 803–821 (2012). doi:10.1016/j.robot.2012.02.004
19. Khetrapal, A., Ganesh, V.: Hbase and hypertable for large scale distributed storage systems. Dept. of Computer Science, Purdue University (2006). Available at: urlhttp://cloud.pubs.dbs.uni-leipzig.de/node/46
20. Limna, T., Tandayya, P.: A flexible and scalable component-based system architecture for video surveillance as a service, running on infrastructure as a service. Multimedia Tools and Applications pp. 1–27 (2014). doi:10.1007/s11042-014-2373-8, http://0-link.springer.com.fama.us.es/article/10.1007/s11042-014-2373-8
21. Liu, J., Nishimura, S., Araki, T.: Wally: A Scalable Distributed Automated Video Surveillance System with Rich Search Functionalities. In: Proceedings of the 22Nd ACM International Conference on Multimedia, MM '14, pp. 729–730. ACM, New York, NY, USA (2014). doi:10.1145/2647868.2654872
22. Lunenburg, J., van den Dries, S., Bento Ferreira, L., van de Molengraft, M.J.G.: Tech United Eindhoven @Home 2015 Team Description Paper. Eindhoven University of Technology, Eindhoven, Tech. rep. (2015)
23. Alsmirat, M.A., Jararweh, Y.: Internet of surveillance: a cloud supported large-scale wireless surveillance system. J. Supercomput. **73**(3), 973 (2017). doi:10.1007/s11227-016-1857-x
24. Martins, G.: Reducing Communication Delay Variability for a Group of Robots. Ph.D. thesis, University of Denver, Denver, CO, USA (2013)
25. Meinel, L., Findeisen, M., Hes, M., Apitzsch, A., Hirtz, G.: Automated real-time surveillance for ambient assisted living using an omnidirectional camera. In: 2014 IEEE International Conference on Consumer Electronics (ICCE), pp. 396–399 (2014). doi:10.1109/ICCE.2014.6776056
26. Neal, D., Rahman, S.M.: Video surveillance in the cloud-computing? In: 2012 7th International Conference on Electrical and Computer Engineering, pp. 58–61 (2012). doi:10.1109/ICECE.2012.6471484
27. Oh, J.M., Moon, N., Hong, S.: Trajectory based database management for intelligent surveillance system with heterogeneous sensors. Multimedia Tools and Applications pp. 1–16 (2015). DOI 10.1007/s11042-015-2725-z. http://link.springer.com/article/10.1007/s11042-015-2725-z
28. Ozalp Babaoglu, Moreno Marzolla: Escape From the Data Center: The Promise of Peer-to-Peer Cloud Computing. IEEE Spectrum Magazine (2014)
29. Prati, A., Vezzani, R., Fornaciari, M., Cucchiara, R.: Intelligent video surveillance as a service. In: Atrey, P.K., Kankanhalli, M.S., Cavallaro A. (eds.) Intelligent multimedia surveillance, pp. 1–16. Springer Berlin Heidelberg (2013). doi:10.1007/978-3-642-41512-8_1

30. Riazuelo, L., Civera, J., Montiel, J.M.M.: C2tam: a cloud framework for cooperative tracking and mapping. Robot. Auton. Syst. **62**(4), 401–413 (2014). doi:10.1016/j.robot.2013.11.007
31. del Rio, F.D., Salmeron-Garcia, J., Sevillano, J.L.: Extending amdahl's law for the cloud computing era. Computer **49**(2), 14–22 (2016). doi:10.1109/MC.2016.49
32. RTC Group: Cloud Based Surveillance System (2015). URL https://www.youtube.com/playlist?list=PLgUj9dv84AxAVFttquWg1VPaza5no5b2K
33. Seo, K.T., Hwang, H.S., Moon, I.Y., Kwon, O.Y., Kim, B.J.: Performance comparison analysis of linux container and virtual machine for building cloud. Adv. Sci. Technol. Lett. **66**(105–111), 2 (2014)
34. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. IEEE Internet of Things Journal pp. 637–646 (2016). doi:10.1109/JIOT.2016.2579198. http://ieeexplore.ieee.org/document/7488250/
35. Shim, J., Lim, Y., Park, J.: Architectural Design of Cloud Gateway in Smart Surveillance System. In: Proceedings of the 2013 Research in Adaptive and Convergent Systems, RACS '13, pp. 261–266. ACM, New York, NY, USA (2013). doi:10.1145/2513228.2513320
36. Song, B., Tian, Y., Zhou, B.: Design and Evaluation of Remote Video Surveillance System on Private Cloud. In: 2014 International Symposium on Biometrics and Security Technologies (ISBAST), pp. 256–262 (2014). doi:10.1109/ISBAST.2014.7013131
37. Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J.M.M., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., van de Molengraft, R.: RoboEarth. IEEE Robot. Autom. Mag. **18**(2), 69–82 (2011). doi:10.1109/MRA.2011.941632
38. Zhang, T., Chowdhery, A., Bahl, P.V., Jamieson, K., Banerjee, S.: The Design and Implementation of a Wireless Video Surveillance System. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom '15, pp. 426–438. ACM, New York, NY, USA (2015). doi:10.1145/2789168.2790123