Study of Communication Issues in Dynamically Scalable Cloud-Based Vision Systems for Mobile Robots

Javier Salmerón-García, Pablo Iñigo-Blasco, Fernando Díaz-del-Río and Daniel Cagigas-Muñiz

Abstract Thanks to the advent of technologies like Cloud Computing, the idea of computation offloading of robotic tasks is more than feasible. Therefore, it is possible to use legacy embedded systems for computationally heavy tasks like navigation or artificial vision, hence extending its lifespan. In this chapter we apply Cloud Computing for building a Cloud-Based 3D Point Cloud extractor for stereo images. The objective is to have a dynamically scalable solution (one of Cloud Computing's most important features) and applicable to near real-time scenarios. This last feature brings several challenges that must be addressed: meeting of deadlines, stability, limitation of communication technologies. All those elements will be thoroughly analyzed in this chapter, providing experimental results that prove the efficacy of the solution. At the end of the chapter, a successful use case of the platform is explained: navigation assistance.

Keywords Cloud computing · Computation offloading · Robotics · Dynamic scalability

1 Introduction

Nowadays, new computationally expensive tasks are expected to be performed with relative fluency by robotic platforms. A well-known example is that of artificial vision and higher level tasks arisen from it, such as object detection, recognition and tracking; surveillance, gesture recognition, etc. However, these tasks are so computationally heavy that they may take several seconds in current embedded robot computers. Hence the advantages of using computation offloading (i.e. moving this computing task to an external computer system) are becoming evident in terms of time to finish the task, mobile robot energy saving, amongst others.

J. Salmerón-García (✉) · P. Iñigo-Blasco · F. Díaz-del-Río · D. Cagigas-Muñiz
Escuela Técnica Superior de Ingeniería Informática, University of Seville,
Av. Reina Mercedes S/n, 41012 Sevilla, Spain
e-mail: jsalmeron2@us.es

An interesting candidate for the aforementioned external computer system is that of a Cloud infrastructure, thanks to its inherent characteristics [7]: high reliability, larger storage capacity, stable electric power, reutilization of hardware, dynamic scalability and better resource utilization. In particular, the dynamic scalability property is very useful in robotics, as it allows the adaptation of the computing power at runtime (that is, scaling out and back, depending on the needs), and therefore it permits the robot to rapidly adapt in a changing environment. Moreover, another advantage of the Cloud is the instant incorporation of more computation demanding algorithms as they are being implemented.

Apart from computationally heavy tasks, the cloud is being used as a centralized powerful infrastructure for multi-robot cooperative systems that usually works at intermediate levels. This area is intensively studied as new cooperative algorithms are being developed. Examples of these solutions are multi-robot SLAM (simultaneous localization and mapping), map merging (acquired by several robots), networked information repository for robots [23], etc.

As a result, an important number of research papers and projects addressing the use of cloud infrastructures in robotics have been published during the last few years (see Sect. 3). Besides, the term Cloud Robotics has emerged to include this area, which promises a fast development of complex distributed robotics tasks in the forthcoming years.

This chapter addresses the "computation offloading" of an intermediate robot level task: 3-D point cloud (3DPC) extraction from stereo image pairs. In order to do so, a Cloud-based 3DPC extraction platform will be developed. This platform has innumerable applications, such as AI, artificial vision and navigation. The latter is especially interesting, as 3DPCs are one of the most used representation for several navigation tasks, including those of motion planning and obstacle avoidance. Because of that, at the end of the chapter (Sect. 6.5) a navigation use case of our platform will be briefly explained.

In this respect, current embedded computers are able to extract a 3D point cloud and to build a map of the surrounding obstacles in a natural and dynamically changing environment in less than a second, providing that low resolution frames are used. Nevertheless, when an accurate vision is needed, frame resolution must be increased. In addition to this, should the robot be in a fast changing environment, then higher frame rates would be necessary. As a consequence, the limitations of robot embedded hardware will likely arise, and thus sending the stereo image pairs to the cloud can compensate the inherent trade-offs of network communications (explained in Sect. 5.2 in more detail).

In order to exploit the cloud capabilities, the implemented platform must be able to scale out and back, so the robot gets the results faster when more computation power is required. Hence, a dynamically parallel algorithm has been implemented. In this context, the quotient between computation and communication times mainly indicates if the parallelization is to be successful. The developed platform is expected to be applied to near real-time systems as well, which is not without several challenges in terms of meeting deadlines. In this respect, the ratio between local-to-cloud transfer

time (especially in the case of large amounts of data) and computation time in a single node must be taken into account as well, as it indicates the usefulness of cloud offloading.

Section 3 summarizes several related works. Before presenting the developed platform, a thorough analysis of which robotics tasks (especially those that need some soft real time requirements) are candidate for computation offloading is done in Sect. 4. An overall analysis of the implemented solution is depicted in Sect. 5 (together with a time analysis). Experimental results are shown in Sect. 6 to quantify the benefits of the cloud approach for different scenarios. In this last section, we summarize an example application case of the presented platform: a navigation assistant for mobile robots [19]. Finally, conclusions are summarized in Sect. 7.

2 Background

This book chapter covers several areas. The first (and most important) is that of Cloud Computing. In this sense, books like [20] can be helpful for understanding its inherent characteristics. More specifically, this chapter focuses on the idea of computation offloading of High Performance Computing applications. Therefore, so some basic concepts this kind of applications, together with basic concepts on parallelism applied to cloud computing, are crucial to understand this chapter. These concepts are clearly explained in [4].

In addition to this, the developed platform is used for stereo vision, and more specifically in 3D Point Cloud extraction. The readers can find several works in the literature regarding this specific topic, for instance [22]. However, it is not necessary to know how a 3D Point Cloud is obtained from stereo frame pairs (that is, debayering, rectification, amongst others) to understand the contents of this chapter.

Moreover, The presented software solution was developed using the ROS Platform. Basic information about this software, together with beginner tutorials, can be found in their wiki (http://wiki.ros.org/). In [12] there is a thorough outline of current Robotics Software Frameworks (RSF).

Finally, communication issues are covered in this chapter. Therefore, readers can read [21] to get basic knowledge about basic networking concepts and technologies.

3 Related Work

In the last few years works and projects that accomplish high level vision tasks without real time requirements are more and more common [2, 9, 23]. Most of them use the cloud robotics paradigm to offload the robot from high level tasks like those related with visual processing or multirobot cooperation. In our opinion this is a tendency that will burst in the next decade, due to the previous cloud computing advantages.

However, a small group of papers proposes offloading the processing of several parts of the sensor feedback information that are near to real-time. For those operations, a fast and reliable response is needed. In [3] a high resolution SIFT-based object detection is speeded up by transmitting on-board preprocessed image information instead of raw image data to external servers. Here, properties of the cloud computing paradigm are not fully exploited, because the configuration of these external servers is specific to this work.

The idea of Computation Offloading is studied in [16]. These authors present an estimation of the computation and communication times needed for the tasks of recognition and object tracking in order to minimize the total execution time (approaching the real-time constraints). Their analysis permits making offloading decisions for object recognition for different bandwidths, background complexities, and database sizes. In this sense, the method for identifying the optimal balance between a cloud system overhead and performance presented in [8] can be useful. Executing SLAM in the cloud is also studied in [18], where they develop a cloud mapping framework (C2TAM). They combine both computation offloading and collaborative work, as the framework allows fusing the information obtained from several robots. They work with a 640×480 pixel RGBD camera and an average data flow of 1 MB/s, below 3 MB/s, which is the usual wireless bandwidth and hence the mapping is successfully done (moreover, they work with keyframes, reducing the amount of images to send).

In [24] an object-tracking scenario for a 14-DOF industrial dual-arm robot is presented. Standard UDP transport protocol for transmitting large-volume images over an Ethernet network is used. Thanks to the very low sending and cloud image processing times that are achieved, a stabilizing control law can be implemented. Due to the inherent time-varying Ethernet protocol delays, actuation signals incorporate an ingenious hold action.

Finally, the work [1] also asks whether the performance of distributed offloading tasks can be compared with those executed on-board. While the experiment performed here is simple (a visual line follower that guides the robot using a single low resolution camera that points to the floor), this demonstrator gives an idea of the possible scenario for many cloud-based robots of the upcoming future.

Contributions of the chapter: Compared with the described literature, the work presented in this chapter is the first that tries implement a stereo vision platform with focus on both dynamic scalability and near real-time applications. Moreover, a general offloading architecture (applicable to any case) is proposed, used to implement the presented platform. In addition to this, a thorough explanation of the the role of communication technologies, the problems of multi-robot and WiFi AC adds more novelty to our work.

4 Cloud Offloading for Robotics Applications

Figure 1 shows a block diagram of a Cloud-based computation offloading robotics applications. The robot's controller will collect any necessary environment information (using both internal and external sensors) to send the most convenient action to the actuators. Usually internal sensors (e.g. odometric) and simple sensor (e.g. sonars) are easy to be processed on-board, so they provide a fast and reactive feedback to the robot. On the contrary, the robot can include some others more complex or high level sensors (e.g. cameras), whose processing algorithms are more demanding both in computing time and energy.

Because of this, the Cloud-based approach aims that the robot's controller will be freed from heavy computations by offloading. In order to do so, it must send to the Cloud all the sensor information. While the size of high level sensoring is usually big, the rest of sensors suppose a few additional bits; hence all the information can be sent to the Cloud. This will even allow the Cloud to do more involved sensor fusion algorithms without demanding real time constraints (like SLAM). While the Cloud is performing all those high demanding computations, the robot can dedicate its computing power for other (real-time) tasks. Once the Cloud has the processed information ready and tailored to each robot, the robot will receive it and make use of it for whatever operation the robot may require (e.g. vision, AI, trajectory modification, etc.). Cloud offloading provides additional benefits due to the inherent centralization that the Cloud supposes for a distributed robotic team. For example, team collaborative tasks can be more easily and fluently handled in the Cloud as it can manage complete information from all the robots.

Even though its advantages are evident, there are several communication bounds and development issues when offloading robotics tasks. The software architecture (and its components) of a complex robotic system must cater for a variety of characteristics, which distinguish it from other system. The most relevant characteristics of are [12]: Concurrent and distributed architecture, Modularity (several components of high cohesion but low coupling), Robustness and fault tolerance; and real time
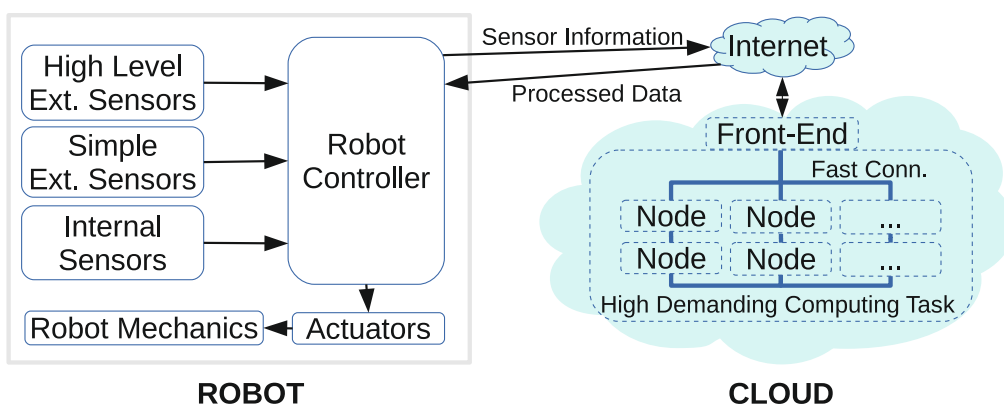

Fig. 1 Block Diagram of a Cloud-based computation offloading system

efficiency. The first two characteristics are primarily benefited by cloud offloading, while the third introduces new challenges (network robustness and fault tolerance appear as a new aspect to considerate). Nevertheless, due that the platform described in this chapter must cope with timing efficiency, communication delays are analyzed in the rest of this section.

As explained in Fig. 1 the robot has to send sensoring information packets to the cloud and wait until it receives the Cloud answer. The communication delays suppose an obligatory inferior bound in the loop controller period. Let BW be the network bandwidth rate and D, the total amount of transmitted and received data. Therefore D/BW must be inferior to the controller deadline. This minimum bound does not suppose for current network technologies a limit, except for the very reactive tasks. For example a WiFi AC networks can deliver until almost 1 Gbps [5]. Even for a demanding control loop of 50 Hz (higher than most mobile robot control loops), this bound would not be exceeded if the transmitted data were less than 0.02 Gbits, because the loop period is 0.02 s.

If images are to be transmitted, an amount of 20 Mbits of data represents 8 raw B/W images of 640×480 pixels (or 32 images for a lower resolution of 320×240 pixels). This suppose that the robot is sending 4 (16 for the lower resolution) stereo frame pairs at each period (without any compression). This is not the common case for a current robot, which is equipped usually with only one stereo camera. Moreover, currently the steady incremental ratio of WiFi networks is more than 40 % per year, which means that only in two years the bandwidth is predicted to duplicate. Hence it can be assured that theoretical bandwidth does not impose a limit in computation offloading. Nevertheless, this may not be the case for latency variability, as our results in Sect. 6.4 demonstrates.

Going further, a quantitative comparison of the times involved in local versus remote computing points out new outcomes. Let IPS the rate of instructions per second that the robot computer can execute [11]. Let us assume that the cloud can speedup an application S times more than the robot, that is, it gets an IPS of $S \cdot IPS$. A high S is expected because of several reasons. Firstly, the cloud is expected to have far more computational resources than a local (usually low power consuming) computer. Likewise, there are big amounts of data parallelism to be exploited when using many sensoring information (image processing, object, voice or face recognition, etc.). Finally these tasks are usually very repetitive. For instance, in image recognition, a pattern has to be compared with thousands of stored patterns. Hence, it can be supposed that S is very big for most sensoring applications. Therefore, for N_I computer instructions local and remote execution times are:

$$t_{local} = \frac{N_I}{IPS}; t_{remote} = \frac{N_I}{(IPS \cdot S)} + \frac{D}{BW};$$

And we can obtain this formula for timing comparison:

$$t_{local} > t_{remote} \text{ if } \frac{N_I}{D} > \frac{IPS}{BW}$$

which indicates whether computation offloading is faster than local computation, and gives us a prospect of which applications are prone to be offloaded. For example, let us compute an estimation of the two members of previous inequality for the Erratic Robot, which CPU runs at a frequency f $= 1.4$ GHz and has a CPI (Clocks per Instruction) around 2.0 [14]. Hence, if a frame pair is computed by this robot in $t_{exec} = 0.96$ s (see Sect. 6), the number of instructions that are executed [11] results:

$$N_I = \frac{(t_{exec} \cdot f)}{CPI} = 6.72 \cdot 10^8 instructions$$

Besides, transmitted data of this experiment (see Sect. 6) consists mainly in a color 1024×768 frame pair, that is: $D = 1024 \cdot 768 \cdot 3 \cdot 2 \cdot 8 = 3.77 \cdot 10^7 bits$. Hence:

$$\frac{N_I}{D} = 17.8 instr/bit$$

Let us remark that the first term of the inequality is application dependent, which means that high intensive computing tasks will be benefited by cloud computing.

On the contrary, the second member is mainly technology dependent. Being $IPS = f \cdot CPI$ [11], the Erratic CPU $IPS = 7.0 \cdot 10^8 instr/s$, but others platforms can achieve a higher IPS. Moreover, a 400 Mbps data rate transmission can be easily reached with WiFi IEEE 802.11ac. With these values, offloading is not bounded by time latency, as the second term has a very much lower value (1.75 for actual case) than the first one. As a conclusion, networking bandwidth is crucial for a successful offloading.

Let us finally make some predictions about the tendency of these two terms. If the last decade trend continues, uniprocessor IPS will have an annual growth rate much inferior to that of network technology [11]. This means that cloud offloading is promised to take even more advantage for the next years. With respect to the software development, it seems that the only possibility to speedup embedded CPU *IPS* is by means of more parallelism (and by more efficient tools to develop it). But this, indeed, would be beneficial for the advancement on cloud programming.

To sum up, it can be concluded that those applications with ratio N_I/D bigger than a few units, are currently candidates to be remotely executed. Those where this ratio would be inferior may be successfully offloaded in a few years. This includes many tasks from the top, middle, and even lowest, level of a common layered robot architec-ture (see Sect. 1). Moreover, independently of this ratio value, there are applications where the size of required (stored) data is huge (see Sect. 1). For them, maintaining local massive storage (in terms of power, failure immunity, backup, weight, etc.) is a hard problem and it is obvious that external offloading is the best solution.

5 3D Point Cloud (3DPC) Extraction Platform

Using the architecture shown in Fig. 1, the offloading of stereo vision tasks has been implemented. As stereo cameras are the high level external sensors, the robot will send a stereo video stream ("Sensor Information" in Fig. 1). The Cloud will extract all the necessary 3D information, sending that processed data (see Fig. 1) back to the robot in the form of 3D Point Clouds (3DPC). The resulting architecture can be seen in Fig. 2. These 3DPCs can be used by the robot to execute, for instance, a navigation algorithm (as explained in Sect. 6.5) or a SLAM algorithm (together with the internal sensing).

As mentioned in Sect. 1, with respect to the precision of the extracted information, the bigger the image resolution is, the more accurate the reconstruction of the surrounding objects will be (extensively demonstrated in the literature [10, 15]). The reason for choosing stereo cameras instead of other simpler sensors (such as those with infrared or ultrasonic technologies) is the completeness of the information they can offer, as well as they can serve for other high level visual tasks (like object detection and recognition, gesture recognition, etc.).

5.1 Software Implementation

In order to convert the image stream sent by the robot to a set of point clouds, the best option is the Point Cloud Library (PCL, http://www.pointclouds.org) combined with the OpenCV Library (www.opencv.org). These large-scale, open source projects for 2D/3D point cloud processing and computer vision, are used by a ROS (Robotics Operating System) library called stereo_image_proc.

This package offers a node that converts two stereo frames to a 3D Point Cloud. In order to do so, the node has an inner pipeline (using ROS nodelets) with several stages. Firstly, a monochrome version of the image is produced (debayer stage).
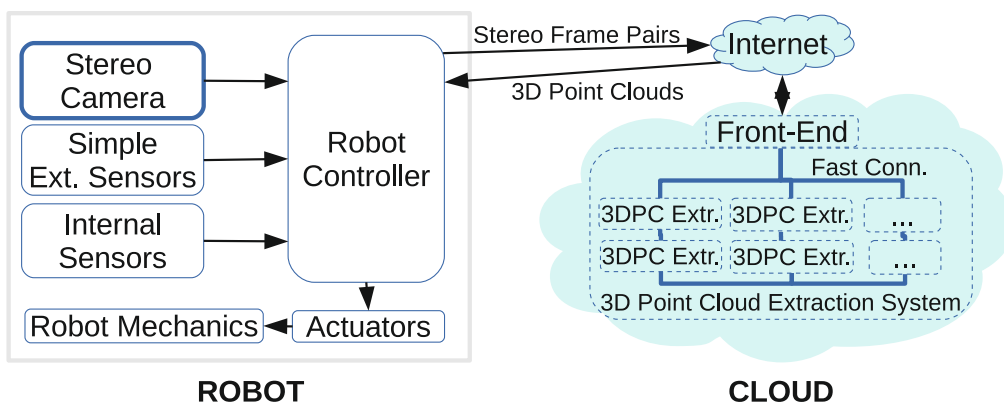

Fig. 2 Block diagram of the 3DPC extraction platform

Secondly, using the stereo camera intrinsic matrices, a rectified version of the image is produced (rectify stage). With the rectified frames, the image matching occurs, obtaining a disparity map (disparity stage). Finally, with this information, the fourth and last step is the 3D point cloud construction (3DPC stage). Due to the very different processing times of the four steps, the minimum time for processing a frame pair will be the maximum of all step times (usually the disparity stage, which lasts most of the whole processing time).

As it can be seen, the aforementioned process cannot be parallelized, as the steps need to be done in order. However, one of the objectives outlined in Sect. 1 is to have a dynamically scalable platform. In order to do so we have exploited the frame pair-level parallelism. Figure 3 depicts the parallel solution. The 3DPC extraction pipeline (stereo_image_proc) is replicated in several virtual instances in the cloud. Therefore, each stereo frame pair will be sent to a different virtual machine in a round-robin fashion. This solution requires an intermediate front-end node, responsible of scattering the stereo stream between the available 3DPC extraction nodes. This way, should the need faster 3DPC extraction times, then more virtual instances could be spun up. However, some extra considerations must be taken into account in order to exploit the parallelism successfully. These considerations are thoroughly explained in Sect. 6.3.

Nevertheless, if we want our system to be dynamically scalable, then it must be able to adapt itself at runtime. Therefore, the buffer node must be able to know how many virtual instances are alive at any moment. This feature is implemented thanks
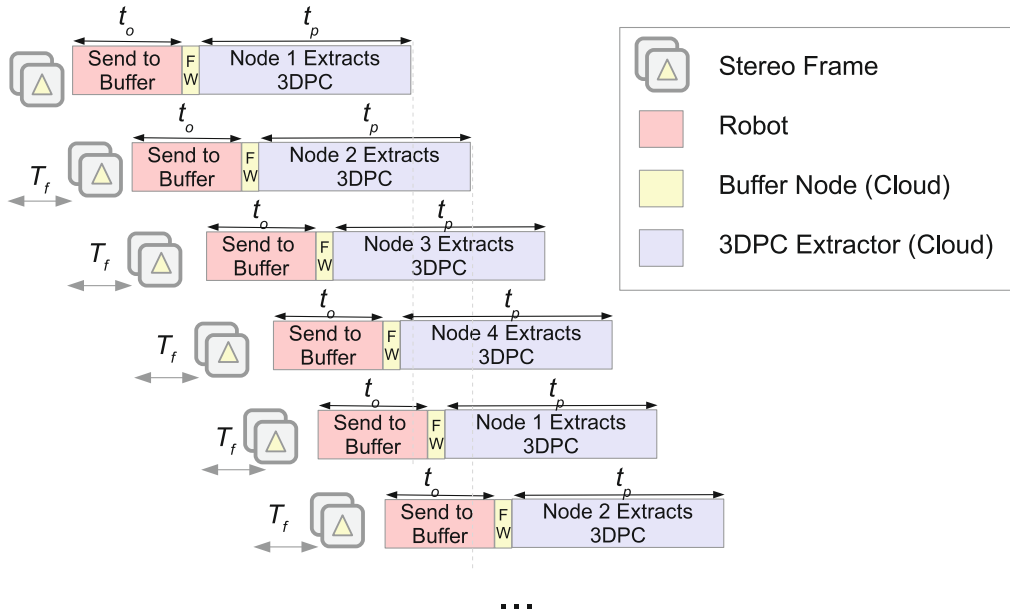

Fig. 3 Stereo frame pipeline process. Four nodes process (in a pipeline fashion) the frame pairs that the front-end node delivers in a round-robin form. T_f is the robot's stereo camera frequency, t_o is the time required to send the frame and t_p is the time required to obtain the 3DPC (see Sect. 5.2 for more information on the involved times). The time required to forward the frame from the buffer node to the 3DPC extractors, thanks to the Gigabit Ethernet connection, is negligible
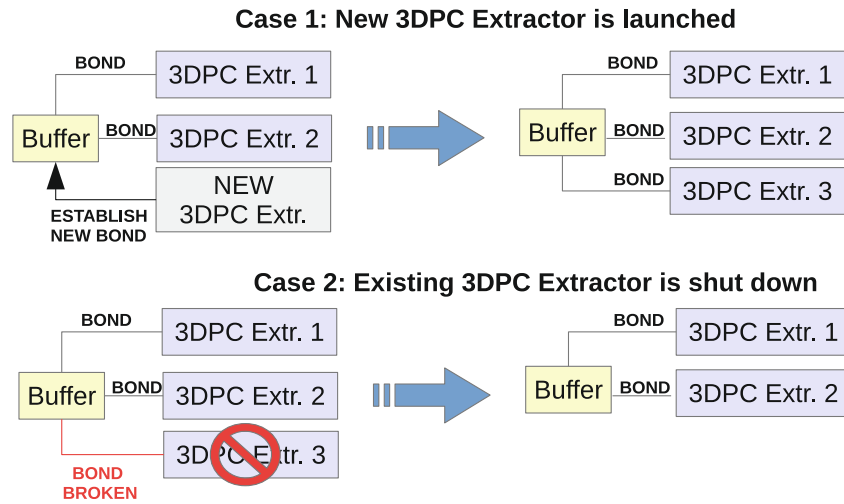
Case 1: New 3DPC Extractor is launched

Case 2: Existing 3DPC Extractor is shut down

Fig. 4 Dynamic adaptation of the platform when the number of 3DPC extractors changes

to ROS bond library. This library helps to establish a link between the intermediate buffer node and a 3DPC extractor. So, if one of the nodes of the link disappears, the other would be automatically notified. Figure 4 shows the different cases:

- A new 3DPC Extractor is added at runtime: as soon as a 3DPC Extractor node is spun up, it will automatically contact the buffer node and establish a bond between them. The buffer node, with this new link added, will add this new node to the round-robin list.
- One existing 3DPC Extractor is shut down: if this occurs, then the bond would be broken, and the buffer node would be immediately informed. Therefore, the round-robin list would be updated.

When dealing with dynamically scalable solutions, another question arises: When should the platform launch more 3DPC extractors or shut down virtual instances? For vision processing applications a simple Quality-of-Service (QoS) magnitude can serve to determine these actions (with a previous agreement between the robot and the platform). For instance: if the robot has agreed a 3DPC reception frequency of 5 Hz and the cloud is under-providing, then it can scale out to satisfy its demands. The same could be applied for over-providing. The user could change this QoS agreement at any time, and the platform would have to apply it accordingly.

For more technical details of the platform, the code is available with GPL license in GitHub.[1] In addition to this, there are cloud images ready for deployment using the newest cloud technologies: Amazon EC2 virtual machines [2] and Docker containers.[3]

---

[1] https://github.com/javsalgar/cloud_3dpc_extractor.

[2] The EC2 AMI is ami-893b11b9.

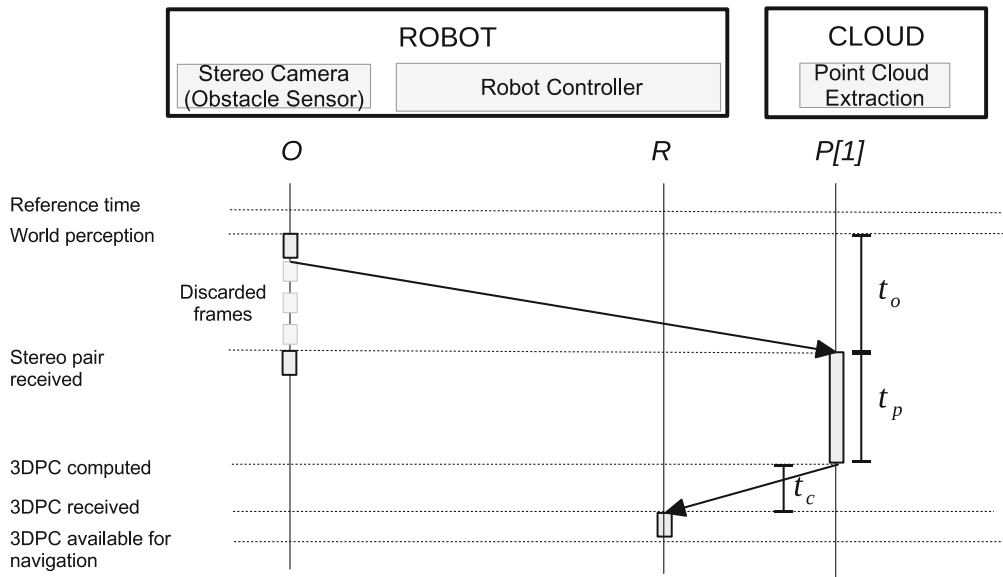[3] https://registry.hub.docker.com/u/javsalgar/cloud_3dpc_extractor/.

Fig. 5 Time diagram of the point cloud extraction for one virtual node. More virtual nodes suppose that more processes P will be running in parallel with different frames, so a little number of frames would be discarded

5.2 Time Analysis

Processes are running in different machines that are interconnected via standard network protocols, due that the system is running over the Robotic Software Framework ROS [17]. Figure 5 shows a simplified timing diagram for one virtual node (the front-end node is not shown because its delay times are negligible with respect to the other involved times). The two physical systems are shown in the upper part of the figure: the robot and the cloud, each one containing the different logical nodes of the system. As seen in Fig. 2, the robot comprises the stereo camera O and the robot controller R (responsible of tasks such as motor actuation subsystem, motion planner, amongst others). Here, R only contains a reception process that validates the point clouds and do the timing calculation. On the other side, the cloud is running the point cloud extractor P, which can be cloned in several virtual nodes.

Pairs of frames are continuously sent from camera node O to the cloud at a specified frequency. The transmission time from O to P is t_o. Each virtual node receives frames in a round robin fashion (in the figure only one node $P[1]$ is represented for simplicity). $P[1]$ extracts the 3DPC, being t_p the time invested.

This new extracted 3DPC is sent back to the robot R. If a new stereo pair joins the stereo_image_proc inner pipeline (explained in Sect. 5) and enters a stage that is still busy processing a previous frame, this new stereo pair will be discarded. Therefore, when processing times of P (t_p in Fig. 5) are elevated (more than the period of O), some frames are discarded (shown like clear rectangles in Fig. 5) until the point cloud extractors are idle again.

One of the crucial points in the timing analysis is the determination of the number of nodes that the cloud computer dedicates to the image processing (t_p in Fig. 5), in order to assure that the mean time to process a pair of frames is less than the transfer time (t_o in Fig. 5). To get rid of this issue, and due that a scalable cloud computer is available, this number is overestimated, so $t_p/p < t_o$ is always guaranteed (where p is the number of active nodes).

A common issue in distributed systems is the synchronization of the different processes and platforms. For the present experiments a simple solution has been carried out: a ping-pong messaging loop is executed between the cloud and the robot. In spite of non-deterministic TCP protocols, an offset under 0.03 s is always achieved, which is enough for our purposes as total computing latencies (see Sect. 6) are always above 0.2 s. Of course a better synchronization will be reached by using TDMA methods or by the incorporation of an external sync device to each platform.

6 Experimental Results

In this section, a set of experiments is described to do an intensive performance testing for different stereo streams, cloud states and connection technologies (between the robot and the cloud). Our cloud-based solution has been deployed in a private small cluster of 5 physical nodes (1 front-end node and 4 computing nodes). Each node has a AMD Phenom 965×4 CPU (with virtual extensions enabled) and 8 GB of RAM. They are all connected using Gigabit Ethernet bandwidth and Openstack Havana is the cloud middleware installed (other well known solutions such as Hadoop were not suitable as we are working with real time systems).

6.1 Scalability of the Platform

The first of the tests is a demonstration that the scalability of our solution is working properly. Thus we use high resolution images (1920×1080 pixels) that result into large 3DPC processing times. In order to isolate this experiment from other delaying factors, the test is carried out with offline video images, and the robot is emulated using an Intel Core i7 4750-HQ laptop with 16 GB of RAM. Moreover, the fastest available TCP network (Gigabit) is used to reduce transmission delay overheads. A variable number of frames is sent to the cloud, which processes them and sends a 3DPC back to the emulated robot.

As Gigabit Ethernet is a possible scenario for static robots, this experiment serves also as a reference of the number of virtual nodes needed to extract 3DPC for high resolution images.

Needless to say that, for low resolution images, 3DPC computation is sufficiently fast, so elevated frequencies are obtained for any p (number of virtual computing cloud nodes). The performance test shown in Table 1 measures the time required for

Table 1 Total times to process and receive n point clouds using p 3DPC extractors

Execution time (s)							
p/n	32	64	128	256	512	1024	2048
1	53.97	96.76	173	331	632	1213	2494
2	32.5	48.47	91.71	178	318	629	1218
4	25.38	33.84	55.09	106	186	437	904
6	27.07	36.66	51.48	87.02	171.3	351	635

Resolution of the stereo pairs is 1920×1080

the emulated robot to receive n point clouds processed in p nodes for HD 1080i video stream frames. A significant speedup (ratio between total time for 1 node and for p nodes) is obtained, approaching a sustained average frequency near to 4 frame pairs per second (reached when a high number of frames are processed). In this case, cloud elasticity makes it possible for the robot to change between different computing resources depending on the frequency required by the robot.

6.2 Communication Technology Performance Measures

Once the scalability of the cloud computing solution has been demonstrated, a second experiment is devised to analyze the performance impact of different communication technologies. As stated before, Gigabit Ethernet is a possible scenario for static robots, but the case of mobile robots (where the use of wireless technologies is practically mandatory) must be taken into account as well.

With this in mind, we have tested two wireless technologies: IEEE 802.11n WiFi and IEEE 802.11ac WiFi. The latter, though being still quite recent, can theoretically achieve bandwidths of 768 Mbps (which is close to what Gigabit Ethernet can offer). In this experiment, we have used the on-board computer of the Videre Erratic robot. the stereo camera and the image transmission has been carried out by a real mobile robot (in this case Videre Erratic by LLC). The cloud is configured to have $p = 6$ 3DPC Extractors.

Table 2 compares the performance of the system (in terms of 3DPC reception frequency) using different technologies and frame resolutions. Two facts can be deduced from these results. First of all, for the case of extracting 3DPC for small resolution frames, no performance boost has been found. This is due to two factors: the robot's hardware hardware is powerful enough (for simpler robots, cloud offloading of this process may be beneficial), and insufficient bandwidth of the networks used (better results could have been found for 10 Gbps Ethernet or Infiniband).

However, the robot starts performing worse (due to its hardware limitations) when the quality of the frames is increased. Hence we are able to obtain speed-ups when offloading this demanding computations.

Table 2 Performance measures for different communication technologies

Average Frequency of 3DPC reception (Hz)				
	320×240	640×480	1024×768	1920×1080
Gigabit	16.3	6.65	2.22	0.84
WiFi 11n	4.04	2.04	0.29	0.14
WiFi 11ac	4.98	3.02	0.76	0.24
Erratic alone	7.15	2.61	1.01	0.02

Erratic alone means that the Erratic robot is working alone, that is, working as a local stereo vision system

Note that there are performance differences between the Erratic robot and the laptop used in Sect. 6.1. To begin with, the laptop's hardware (both RAM and CPU) is 4 times better than that of Erratic's. Moreover, there are extra factors that affect the overall performance (even though the robot's controller has less to compute because of the offloading), such as frame buffering and sending, peer to peer connection management, 3DPC reception, amongst others.

This experiment (together with the one explained in Sect. 6.4) shows the current limitations of wireless technologies due to the big amount of data to transfer. In order to address this (as explained in Sect. 4) the computation versus communication trade-off must be carefully analyzed for each application case (as done in Sect. 6.5).

6.3 Time Delay Measures

Very delayed data is usually useless for most information processing applications, especially those with near real-time requirements. Taking into account the timing explained in Sect. 5.2, in this third experiment the average latencies to receive the 3DPC of each individual frame are obtained. Each latency is defined here as the time passed since the source stereo frame was actually obtained to the 3DPC reception.

Table 3 shows the latencies obtained (using 1024768 resolution frames) for different communication technologies. The last row shows the same times for Erratic robot computing all the process on its own (no network is used). Once again, these times can serve as a reference to show the viability of cloud computing.

Table 3 Average delay measures for Gigabit Ethernet in the case of 1024×768 resolution frames

Delay times (s)				
	t_o	t_p	t_c	Total
Gigabit	0.0704	0.389	0.317	0.7764
WiFi 11n	0.676		2.377	3.442
WiFi 11ac	0.4740		1.61	2.473
Erratic alone	0.0670	0.966	0.166	1.199

Erratic alone means that the Erratic robot is working alone, that is, working as a local stereo vision system

In order to calculate the delay, the average times taken to perform each of the stages explained in Sect. 5.2 (t_o, t_p and t_c) are measured using time stamps at the beginning of every process. The average latencies are calculated by adding the mean runtime of all these stages. In the case of using the cloud, the difference between technologies can be found in the transfer times t_o and t_c, whereas t_p remains unaffected (Fig. 2) clarifies this statement).

As it can be seen, for lower resolutions the robot can outperform the Cloud if wireless technologies are used. However, when increasing the stereo frame resolution, there is a point where the limitations of the embedded hardware start to arise. Therefore, it is worth considering Cloud offloading when bigger resolutions are required.

6.4 Interference Analysis with WiFi AC

The performance of the Cloud itself is not crucial, as we have the premise of "infinite resources". However, as wireless technology is the best choice for mobile robots, an in-depth analysis of interference when increasing the number of robots must be done. It is highly likely that not only one robot, but several are using the cloud at the same time. Hence it is extremely important to study the possible communication quality degradation.

The aim of this experiment is to analyze how WiFi 11ac manages the interferences, and to prove that it is the most suitable technology for mobile robots operation. We will focus only in the transmission of stereo frame pairs (resolution of 320×240) to the Cloud, what renders enough information about interference problems. We are interested in two elements:

- The average transfer time needed to send a stereo frame pair to the Cloud. As we are working with a real-time system, the meeting of certain deadlines is vital. For example, if the robot is transmitting stereo frames at $5\,\mathrm{Hz}$, transfer times lower than $1/5\,\mathrm{Hz} = 0.2\,\mathrm{s}$ are desired in order to meet deadlines.
- The message success rate. When more robots are added, there is the risk that some of the packets that form the message (containing a stereo frame pair) collide and get corrupted. Even though transport-level protocols like TCP allow packet resending, the following scenario could occur: while the Cloud waits for the missing packet (which corresponds to a stereo frame message with timestamp t) to be resent, the same robot had already begun sending packets of a new stereo frame message (that is, a frame with timestamp $t + 1$). Should a packet from a frame with timestamp $t + 1$ arrive, then all the packets from messages with a timestamp lower than $t + 1$ would be automatically discarded (because of its obsolescence). This necessary implies a lower message success ratio.

Table 4 compares the average latency and message success ratio when the number of robots and the message frequency increase. To begin with, note that the packet success rate works exactly as expected. When more robots are added, the number of

Table 4 Performance comparison when adding more robots in the case of 320 × 240 when no 3DPC extraction is done and only delays in stereo frame transmissions are considered

	# Robots	Mean transfer time (s)	Average message success (%)
5 Hz	1	0.117	100.00
	2	0.124	100.00
	4	0.157	94.99
	6	0.147	87.88
10 Hz	1	0.063	100.00
	2	0.086	99.86
	4	0.082	94.99
	6	0.084	87.79

The wireless technology is that of 802.11ac

packet collisions increases, and therefore more messages are lost. Thus, there is a trade-off between number of robots and system stability (e.g., missing environment information can result in a robot crash, in the context of robot navigation). The average transfer time deteriorates until a point where the percentage of message success is low enough. This phenomenon is understood because the mean transfer times are calculated only with those packets that have arrived successfully. That is, those "lucky" packets last little time to complete. Hence, it is not that messages are faster now, but that more packets do never arrive to the Cloud (and therefore their transfer time cannot be properly measured). Therefore, we can assure that there is another trade-off between message success ratio and average transfer time.

With respect to the meeting of deadlines, Fig. 6 shows the Empirical Cumulative Distribution Frequency (ECDF) of delays for the experiment above shown. As it can be seen, adding more robots make it more difficult to meet deadlines (vertical gray dashed line in the figures) because of network interference. This is especially evident in the case of 10 Hz. Therefore, because of the trade-offs previously explained, we can conclude that current wireless technologies are (at the moment) not enough developed for very critical real-time applications when more than one robot in the same wireless cell. Should this be the scenario, then it would be necessary to allow less strict deadlines. The high variability of total latency times that occurs in our experiments can be mitigated by a predictive timing correction of actuation signals [13]. There will be necessary further improvements in 802.11ac MAC layer like TDMA protocols to reduce this latency variance (as mentioned in Sect. 5.2). The use of the Contention Free Period with fixed size packets is an alternative to TDMA protocols. This could guarantee a minimum bandwidth reservation, and therefore we could address the issues explained in this experiment.

6.5 Application Case: Navigation Assistant

This last test summarizes the results for a real task for our cloud vision platform: a navigation assistant for mobile robots. While a teleoperator is driving a mobile robot, the information processed by the 3DPC Extraction Platform helps him/her
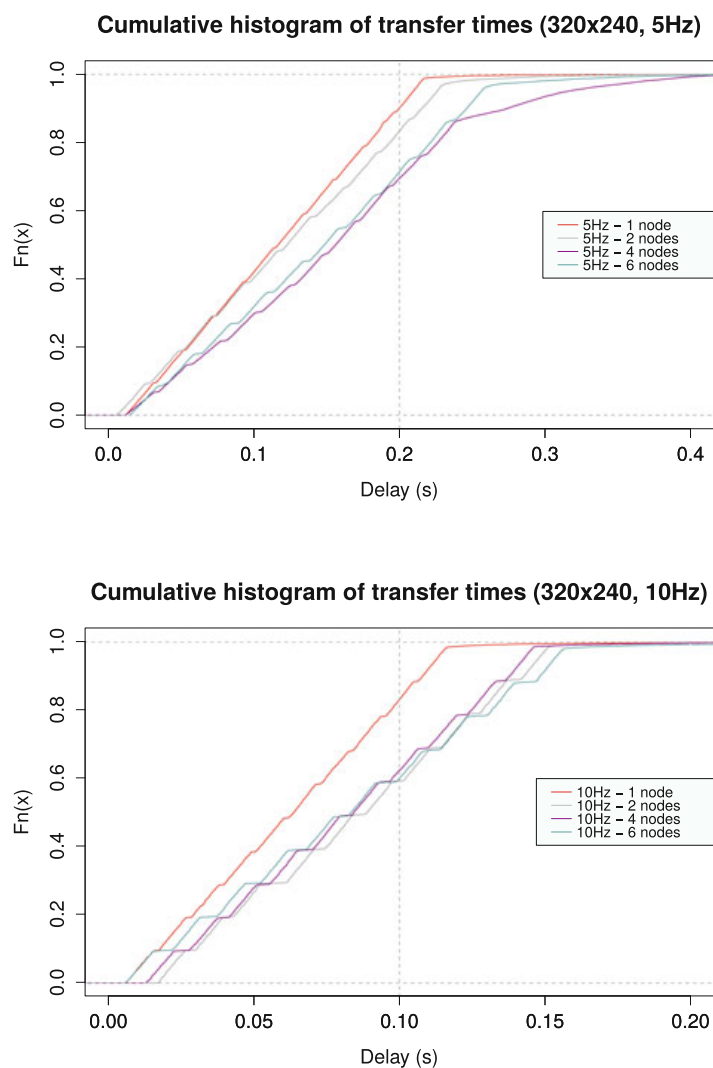
Cumulative histogram of transfer times (320x240, 5Hz)

Cumulative histogram of transfer times (320x240, 10Hz)

Fig. 6 Empirical cumulative distribution frequency of delays with 5 and 10 Hz

to avoid collisions. Numerous questions arise, as in any real experiment: are really high quality stereo frames necessary to assist in the navigation?, are cloud solution more effective than the on-board one?, do packet latency variability suppose a problem when navigating? If on-board navigation were successful enough for 320×240 stereo frames (which have an adequate 3DPC frequency rate, see Table 2), then Cloud offloading would not be necessary at all. In order to answer these questions, a testing circuit was prepared (see Fig. 7). The Erratic robot was equipped with a stereo camera built from two PSEye cameras and the circuit was completed several times. The ratio of collisions by maneuvers was used as a success magnitude.

The results obtained in Sects. 6.3 show that most of the delays come from t_c, that is, the time required to transfer the 3DPC back to the robot. Thus, we got rid of this communication overhead by moving the navigation assistant to the Cloud as well. Figure 8 shows the diagram of the resulting architecture.

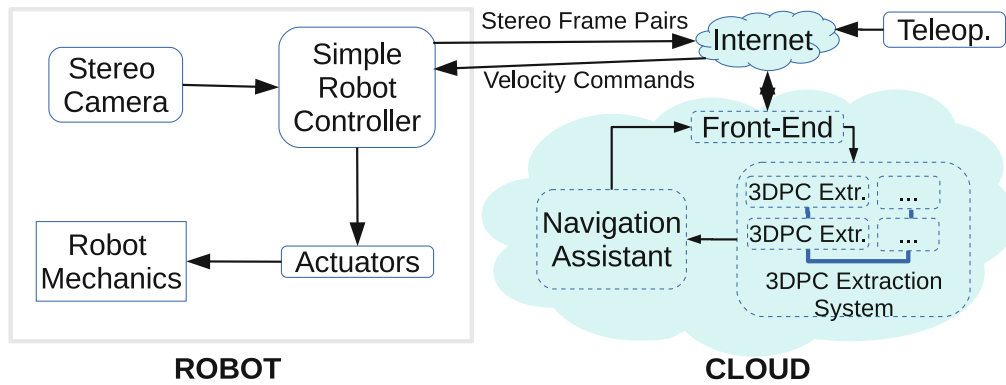**Fig. 7** Testing circuit used in the experiments

Fig. 8 Overview of the platform applied to the navigation use case

Thanks to this change in the offloading model, we obtained the following results, which solve most of previous questions. First of all, collisions were very frequent (about 50 %) when using 320×240 images (for any computing option). Secondly, the number of collisions were considerably reduced (less than 10 %) with higher resolutions (640×480 pixels) and using the Cloud. As a conclusion, for the stereo vision algorithm used here, low resolution images are not enough to detect the obstacle information properly, and hence using higher resolution images is justified. Moreover, images with more than 320×240 are more difficult for the Erratic robot to process on-board. A demonstration video can be found in [6] and all the details of the experiments and the navigation assistant can be seen in [19].

7 Conclusions and Lessons Learned

The implemented platform (and its experimental results) shows that the cloud-based offloading of heavy visual processing tasks is possible. Several conclusions can be extracted from the experience.

Firstly, the main bottleneck of cloud offloading is due to communication overheads. It is extremely important to mitigate this effect by choosing the correct network technology. Moreover, the non-real time middleware and the inherent non-deterministic of the TCP protocol (available in most Robotics Software Frameworks) introduce a high variability in timing latency. Thus, this drawback should be mitigated by using some kind of predictive correction terms in the loop controller and more deterministic middleware and networks. However, the results obtained by WiFi 11ac are promising, and in future years it may be able to provide bandwidths close to its theoretical 768 Mbps, which may reduce the collision problem that currently appears even for a reduced number of robots (as it has been outlined in Sect. 6.4).

Secondly, there is an inherent trade-off between computation offloading and communication overhead times. Therefore, the platform should be used finding the best balance between those two. In that sense, depending on the use case, it may be worth considering offloading not only the 3DPC extraction, but also other robotics tasks (just like the case shown in Sect. 6.5).

In addition to all this, it has also been demonstrated that if a Cloud Solution is not scalable, it is highly unlikely that good performance results can be achieved, and therefore impossible to meet real-time requirements. As it has been shown in Sect. 6.1, this is an indispensable element to exploit the Cloud's true potential.

As a final conclusion, it can be assured that, despite there are challenges that need to be addressed, the main question has been answered: using the Cloud for offloading can imply better performance results in a robot than using on-board computation (at least for a typical robot, whose hardware is much limited).

References

1. Agostinho, L., Olivi, L., Feliciano, G., Paolieri, F., Rodrigues, D., Cardozo, E., Guimaraes, E.: A cloud computing environment for supporting networked robotics applications. In: 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), pp. 1110–1116 (2011). doi:10.1109/DASC.2011.181
2. Arumugam, R., Enti, V., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F.F., Kumar, A., Meng, K.D., Kit, G.W.: DAvinCi: a cloud computing framework for service robots. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 3084–3089 (2010). doi:10.1109/ROBOT.2010.5509469
3. Bistry, H., Zhang, J.: A cloud computing approach to complex robot vision tasks using smart camera systems. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3195–3200 (2010). doi:10.1109/IROS.2010.5653660
4. Buyya, R., Vecchiola, C., Selvi, S.T.: Mastering Cloud Computing Foundations and Applications Programming. Morgan Kaufmann/Elsevier, Amsterdam (2013)

5. Charfi, E., Chaari, L., Kamoun, L.: PHY/MAC enhancements and QoS mechanisms for very high throughput WLANs: a survey. IEEE Commun. Surv. Tutor. **15**(4), 1714–1735 (2013). doi:10.1109/SURV.2013.013013.00084

6. Cloud-based robot navigation assistant using stereo image processing. http://www.rtc.us.es/cloud-based-robot-navigation-assistant-using-stereo-image-processing/ (2014). Accessed 12 Nov 2014

7. Furht, B., Escalante, A. (eds.): Handbook of Cloud Computing. Springer, New York (2010)

8. Gouveia, B.D., Portugal, D., Silva, D.C., Marques, L.: Computation sharing in distributed robotic systems: a case study on SLAM. IEEE Trans. Autom. Sci. Eng. (T-ASE) **12**(2), 410–422 (2015)

9. Guizzo, E.: Robots with their heads in the clouds. IEEE Spectr. **48**(3), 16–18 (2011). doi:10.1109/MSPEC.2011.5719709

10. Handa, A., Newcombe, R.A., Angeli, A., Davison, A.J.: Real-time camera tracking: when is high frame-rate best? In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) Computer Vision ECCV 2012. Lecture Notes in Computer Science, vol. 7578, pp. 222–235. Springer, Berlin (2012)

11. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach. Morgan Kaufmann, Waltham (2012)

12. Iñigo-Blasco, P., Diaz-del Rio, F., Romero-Ternero, M.C., Cagigas-Muñiz, 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. http://www.sciencedirect.com/science/article/pii/S0921889012000322

13. Iñigo-Blasco, P., Diaz-del Rio, F., Vicente-Diaz, S., Cagigas-Muñiz, D.: The shared control dynamic window approach for non-holonomic semi-autonomous robots. In: Proceedings of 41st International Symposium on Robotics. ISR/Robotik 2014. Munich (2014)

14. John, B.P.: Effectiveness of SPEC CPU2006 and Multimedia Applications on Intel's Single. Dual and Quad Core Processors. ProQuest (2009)

15. Kytö, M., Nuutinen, M., Pirkko, O.: Method for measuring stereo camera depth accuracy based on stereoscopic vision. In: Proceedings of SPIE/IS&T Electronic Imaging 2011 (2011)

16. Nimmagadda, Y., Kumar, K., Lu, Y.H., Lee, C.S.G.: Real-time moving object recognition and tracking using computation offloading. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2449–2455 (2010). doi:10.1109/IROBlS.2010.5650303

17. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.: ROS: an open-source robot operating system (2009)

18. 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

19. Salmeron-Garcia, J., Iñigo Blasco, P., Diaz-del Rio, F., Cagigas-Muniz, D.: A trade-off analysis of a cloud-based robot navigation assistant using stereo image processing. IEEE Trans. Autom. Sci. Eng. (T-ASE) **12**(2), 444–454 (2015)

20. Srinivasan, S.: Cloud Computing Basics. Springer Briefs in Electrical and Computer Engineering. Springer, New York (2014)

21. Stallings, W.: Data and Computer Communications. Always Learning, 10th edn. Pearson, Boston (2014)

22. Szeliski, R.: Computer Vision Algorithms and Applications. Texts in Computer Science. Springer, London (2011)

23. 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

24. Wu, H., Lou, L., Chen, C.C., Hirche, S., Kuhnlenz, K.: Cloud-based networked visual servo control. IEEE Trans. Ind. Electron. **60**(2), 554–566 (2013). doi:10.1109/TIE.2012.2186775