UNIVERSIDAD DE SEVILLA

DOCTORAL THESIS

Estudio y evaluación de plataformas de distribución de cómputo intensivo sobre sistemas externos para sistemas empotrados.

Study and evaluation of intensive distributed computing platforms on external systems for embedded systems

Author:
Javier J. Salmerón García

Supervisor:
Dr. Fernando Díaz del Río

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Robotics and Technology of Computers Lab
Departamento de Arquitectura y Tecnología de Computadores

December 2, 2015

Declaration of Authorship

I, Javier J. Salmerón García, declare that this thesis titled, 'Study and evaluation of intensive distributed computing platforms on external systems for embedded systems' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSIDAD DE SEVILLA

Abstract

Escuela Técnica Superior de Ingeniería Informática
Departamento de Arquitectura y Tecnología de Computadores

Doctor of Philosophy

Study and evaluation of intensive distributed computing platforms on external systems for embedded systems

by Javier J. Salmerón García

Nowadays, the capabilities of current embedded systems are constantly increasing, having a wide range of applications. However, there are a plethora of intensive computing tasks that, because of their hardware limitations, are unable to perform successfully. Moreover, there are innumerable tasks with strict deadlines to meet (e.g. Real Time Systems). Because of that, the use of external platforms with sufficient computing power is becoming widespread, especially thanks to the advent of Cloud Computing in recent years. Its use for knowledge sharing and information storage has been demonstrated innumerable times in the literature. However, its core properties, such as dynamic scalability, energy efficiency, infinite resources... amongst others, also make it the perfect candidate for computation off-loading. In this sense, this thesis demonstrates this fact in applying Cloud Computing in the area of Robotics (Cloud Robotics). This is done by building a 3D Point Cloud Extraction Platform, where robots can offload the complex stereo vision task of obtaining a 3D Point Cloud (3DPC) from Stereo Frames. In addition to this, the platform was applied to a typical robotics application: a Navigation Assistant. Using this case, the core challenges of computation offloading were thoroughly analyzed: the role of communication technologies (with special focus on 802.11ac), the role of offloading models, how to overcome the problem of communication delays by using predictive time corrections, until what extent offloading is a better choice compared to processing on board... etc. Furthermore, real navigation tests were performed, showing that better navigation results are obtained when using computation offloading. This experience was a starting point for the final research of this thesis: an extension of Amdahl's Law for Cloud Computing. This will provide a better understanding of Computation Offloading's inherent factors, especially focused on time and energy speedups. In addition to this, it helps to make some predictions regarding the future of Cloud Computing and computation offloading.

Acknowledgements

The project described in this document has been, without room for doubt, one of the biggest challenges in my life. I consider it my first real research, job that I find fascinating and to which I would love to focus my professional career (together with teaching, another passion that I have).

After overcoming all the difficulties, I have been able to finish it in the given time. I can honestly say that, if it had not been for all the given support, I would definetly not have succeeded in this task. Hence I would like to dedicate some words to all the people that helped me to make this thesis possible.

To begin with, I really want to express my deepest gratitude to my supervisor Fernando Díaz del Río. Not only for all the help and support throughout the whole project, but also for having the patience of attending me every time a problem arose, knocking at his office door without any previous appointment. He trusted helped me so much that I simply cannot imagine a better supervisor. In that sense, I couldn't have been luckier. In addition to this, I really want to help the assistance and knowledge of Jose Luis Sevillano, Daniel Cagigas and Pablo Iñigo.

I really think that, if this thesis was possible, is thanks to being in a research group that is like a second family. My gratitude goes to Elena Cerezuela, Manuel Domínguez, Rocío García, Miguel Ángel Rodríguez, Alejandro Linares, Gabriel Jiménez, Anton Civit, José Luis Guisado, Manuel Rivas, Ángel Jiménez, Francisco Gómez, Lourdes Miró, Saturnino Vicente, Ignacio García, Antonio Ríos, Juan Pedro Domínguez, Ricardo Tapiador and the rest of colleagues of the Robotics and Technology of Computers Lab. I cannot forget all the great moments and support during all these years, especially in our "debate" meals, research trips, classes, and innumerable experiencies. A special mention goes to Daniel Cascado for all his interesting comments, which were really helpful for my thesis.

Moreover, this work has been supported by the Spanish grant (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02/01) and by Andalusian Regional Excellence Research Project grant (with support from the European Regional Development Fund) MINERVA (P12-TIC-1300). I also want to thank the University of Seville for my IV Plan Propio grant, which helped me to have full dedication in this thesis. My deepest gratitude for this support.

During this period, I also had the opportunity visit foreign institutions and attend international conferences. There I had the chance to meet extraordinary researchers that did help to become the researcher I am today. Among these people, I will never forget Helen Karatza (Aristotle University of Thessaloniki), Rene van de Molengraft and Sjoerd van den Dries (TU/e Eindhoven).

Moreover, I am extremely lucky to have my wonderful parents, Mariano and Lourdes, and my greatest friends Daniel, Yesu, Gonzalo, Israel, Alejandro, Luis, Rafael, Jesús, Ángel, Iria, Goretti, Alberto, Antonio, Javier... amongst other extraordinary people. Moreover, I want to thank my family for all their support and love.

One of the biggest gratitudes goes to my beloved Elena Rodriguez, with whom I have been living seven years of love, happiness, support and passion, becoming the reason why I feel every day a little more alive and full. For what it has been, and for all that it will be, thank you very much. Obviously, I cannot forget our dear Ini, our first member of our future family together.

To all of them, once again, thank you.

Contents

To Fede.

Part I

Thesis

Chapter 1

Introduction

Traditionally, every information processing task was done by independent machines or centralized systems. In the case of embedded systems, such as mobile robots, this was true as well. However, while this approach worked well at the beginning, there was a moment where more and more computing intensive tasks were expected to be done by these systems. As a consequence, the hardware limitations eventually arised, and hence innumerable issues had to be addressed: meeting of deadlines, energy consumption, battery duration, amongst others.

Having identified the issues, two choices were possible:

- Upgrade the hardware: if the system's current hardware is unable to do the task, a reasonable option would be upgrading it by buying more powerful hardware. Nevertheless, when it comes to embedded systems, the upgrading process is not that straightforward as with traditional personal computers (PCs). To make matters worse, sometimes it is simply impossible to upgrade the embedded hardware.Therefore, the only choice is to dispose the whole embedded system and buy a new one, with all the capital expenditure involved.

- Put limits to the computation the system can perform: in other words, this implies narrowing the variety of doable tasks. If we use the context of computer vision as an example, if a robot cannot process high resolution images, then the resolution would have to be reduced. As a consequence, this can have a direct impact in the quality of the developed application.

A main question arises from this: couldn't there be a third, middle-of-the-road alternative where capabilities are upgraded without dramatic capital expenses? This is extremely important as there are a plethora of complex applications that could be developed exploiting the most from existing (even legacy) resources (and thus saving money and energy): robot navigation, scientific tasks, video surveillance, stereo vision, AI... etc.

Because of that, this thesis aims to respond this question using one of current top trending technologies: Cloud Computing.

The definition of Cloud Computing has yet to be properly established, but the main concept was already stated by John McCarthy in 1961[1]:

> If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility....The computer utility could become the basis of a new and important industry

In this sense, a platform known as the Cloud emerged in order to satisfy this demand. The two commonest properties are the following:

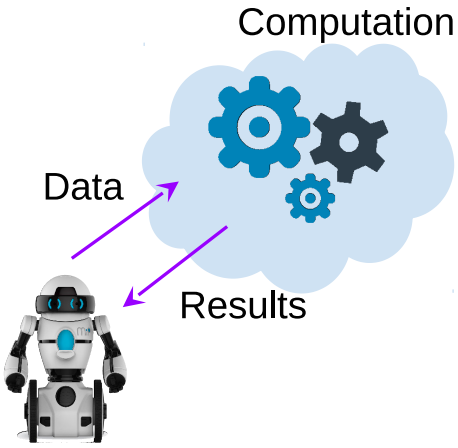


FIGURE 1.1: Concept of computation offloading

- On-demand virtual resource provision: The cloud is able to provide both computing and storage resources. This is normally done in the form of virtual machines. Should an user require computing power to perform a certain task, then it would only have to spin up as much virtual 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. However, undoubtedly this will 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 dinamically adapt to the user needs. A clear example is that of a web service. If, suddenly, the number of requests dramatically increase, the service might not be able to process them in a reasonable time. In order to address this, the Cloud could increase the number of virtual resources devoted to this task. Moreover, if the number of requests decreases, then the cloud could reduce the amount of virtual resources. Once again, this relates to the "utility" concept behind Cloud Computing. However, this forces the developer to rethink their software architectures so the dynamically scalable property can be exploited.

Therefore, a third solution to overcome the aforementioned issues for embedded systems would be the following:

- Move the computation to the Cloud: The system would send the information to this external platform. The information would be processed and the results would be sent back. This way, the only computation upgrade the embedded system would need is a proper network connection (see figure 1.1).

In this thesis, in order to show the viability of this third choice, Cloud Computing is applied to robotic tasks. The research area is known as as Cloud Robotics (first coined by James Kuffner in 2010) [2]. This term refers to the use the both storage and computing power of the cloud for the successful accomplishment of robotic tasks. In this sense, there are several items of interest in this area[3]:

The first one is that of Computation Offloading. The idea is to free the robot from heavy computations and have an external platform (the Cloud in this case) do them and get the results back. This would solve the two aforementioned issues, as it allows the robots to perform more complex tasks, overcoming its hardware limitations

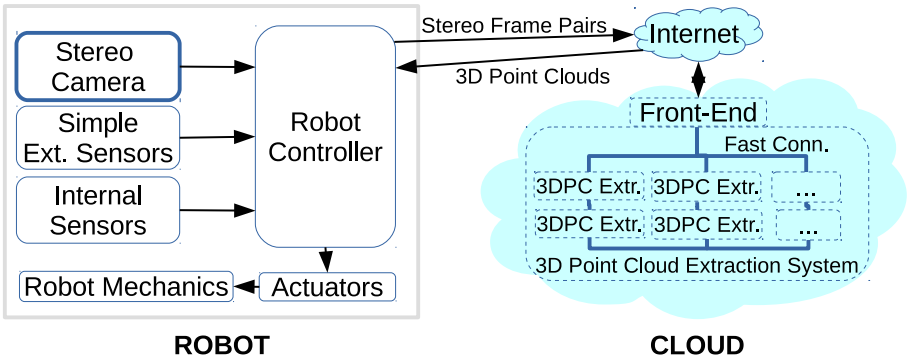


FIGURE 1.2: Diagram of the 3DPC Extraction Platform

(which is difficult to upgrade). Furthermore, offloading CPU-intensive tasks implies less power consumption in the robot. Following this idea, it will possible for engineers to build cheaper (extremely powerful hardware would not be required) and more energy-efficient robots. However, this does not come without inherent trade-offs and bottlenecks, namely the communication technology and software related issues.

A second item is that of Information Storage and Sharing. As stated before, the cloud offers storage resources, hence it is possible for the robots to store and retrieve data like maps, object information, pictures, amongst others. Furthermore, the use of a centralized platform like the Cloud allows the robots to build a common source of knowledge, and just like before, extending its capabilities. For instance, if a robot does not know how to handle a certain object, then it can query the Cloud for information. This concept has been defined as "an Internet for robots"[4].

For this thesis, the use of Computation Offloading in near real time scenarios has been analyzed. More precisely, the focus was on the offloading of stereo-vision tasks to the Cloud. In this sense, a dynamically scalable cloud-based 3D Point Cloud Extraction Platform was initially developed [5] . Afterwards, this platform was used for navigation assistance of mobile robots[6] (see figure 1.2). The experiments showed that, when higher resolutions were required, the limitations of the robot's embedded hardware started to arise. Therefore, the robot was able to navigate better using the cloud. Furthermore, the study of this case showed the trade-off between computation offloading and communication, and that choosing which tasks should be offloaded to the Cloud is far from trivial.

These experiments showed the viability of computation offloading in a near to real time scenario. Moreover, several aspects and challanges of computation offloading were thoroughly analyzed: offloading models, communication bottlenecks, communication delays... etc. However, it was necessary to take a step further, and make a deeper analysis of the concept of computation offloading. Therefore, fruit of this research, an extension of the Amdahl's Law for Cloud Computing was devised, providing a theoretical analysis for computation offloading in terms of performance and energy efficiency[7]. This helps to set a reference theoretical framework when dealing with computation offloading and Cloud Computing. It helps to get an idea whether an application should or shouldn't be offloaded. Moreover, it is an extremely useful tool for researchers to study future trends in computation offloading and hardware development.

This thesis is made of a series of papers [5]–[7], which are included in part II. In part I, there is a summary structured as follows: in chapter 2 the general objectives

of this thesis are described. In chapter 3 all the results obtained from this research are explained and analyzed. From these results, in chapter 4 we dive into their implications and future challenges. Finally, we outline the conclusions from this research in chapter 5.

Chapter 2

General Objectives

As stated in chapter 1, this thesis' main objective is to study the viability of Cloud Offloading for embedded systems. In order to obtain an answer, it is necessary to analyze various elements. These can be organized in the following categories:

Use Case Analysis [UCA]

In order to empirically prove that computation offloading in embedded systems is a feasible option, a real use case must be analyzed. In this sense, as stated in chapter 1, Robotics can be an extremely useful application scenario. In addition to this, cases involving extremely heavy computations and near to real time constraints could best showcase the Cloud's potential. Not to mention the innumerable research challenges that a case like this inherently brings. For all those reasons, the chosen case for this thesis is that of Navigation Assistance based on Stereo Vision. Therefore, the objectives in this category would be:

(UCA-1): Build a Cloud Based Stereo Vision platform where robots can offload stereo information processing.

(UCA-2): Build a Cloud-based Navigation Assistant that uses the stereo information to avoid obstacles.

(UCA-3): Perform tests on the whole system using simulated data, finding its potential benefits and pitfalls.

(UCA-4): Perform real navigation tests, comparing two cases: using computation offloading and performing all the computation on-board.

Study of Cloud Properties [SCP]

As mentioned in chapter 1, the cloud has unique properties that imply a big leap over other technologies such as cluster or grid computing. Because of that, this work must make use of all those properties when providing a Cloud offloading solution for embedded systems. Therefore, this category contains the following objectives:

(SCP-1): Analyze the performance impact of different virtual machine types.

(SCP-2): Study the existing trade-off when choosing different offloading models and configurations.

(SCP-3): Exploit Cloud's dynamic scalability property when building a solution and study its performance impact when using more and less resources (scaling out and back).

Compare Communication Technologies [CCT]

When it comes to offloading, especially when dealing with time constraints, there is unavoidable question that must be addressed: How can we overcome network latencies? Indeed, there is a risk that, even though there are computation speedups, the information does not arrive in time because of delays in the communication technologies. Therefore, in this thesis a thorough research on this area must be done in order to mitigate its effects:

(CCT-1): Compare the performance impact when using different kind of technologies.

(CCT-2): Perform deeper testing on wireless technologies, vital for mobile robots. Study the current state of 802.11ac technology, as is the newest and most promising.

(CCT-3): Propose a way to mitigate the effects of a delay in the information.

Devise a Theoretical Framework [DTF]

While performing empirical demonstrations is essential for answering this thesis' main question, not less valuable is providing a small theoretical framework for researchers and developers. This would be extremely useful for future application development, as well as new research areas. Hence, the objectives would be the following:

(DTF-1): Provide a general design pattern for offloading robotic applications.

(DTF-2): Provide basic theoretical formulae that helps deciding whether an application is worth offloading or not.

(DTF-3): Make predictions on the future possibilities of computation offloading.

Chapter 3

Summary of results

In this chapter all the obtained research results will be briefly described. These results will make reference to the objectives described in chapter 2. These global results can be divided into two categories:

Stereo Vision Offloading applied to navigation assistance

In this thesis, an application case is used to study the properties, issues and challenges of Cloud Offloading.In other words, an empirical demonstration of the effectiveness of Cloud offloading is sought.

Due to the high computational requirements and their importance in robotics, a case in stereo vision is proposed: several vision tasks use as main input a 3D Point Cloud (3DPC), which can be obtained from several input sources (stereo cameras, laser rangefinders, RGB-D cameras...). Extracting a 3DPC from stereo frame pairs (see figure 3.1) is computationally expensive. In addition to this, if more quality of the environment representation is desired, then higher resolution in the stereo camera is required. As a result, legacy embedded systems may be unable to obtain 3DPCs from stereo frames in a reasonable time. Therefore, developers would have no option but to purchase expensive, cutting-edge embedded systems, and still there would be no warranty that they would be able to process, for instance, HD 1080p stereo frames fast enough.

Because of this, a Cloud-based 3DPC Extraction Platform was implemented (UCA-1) based upon the aforementioned general architecture (DTF-1). This platform is based on the Robotics Operating System (ROS), and specially designed to be able to scale out and back depending on the computational needs at runtime (SCP-3). In order to do so, pipeline-based parallelism is exploited, as it can be seen in figure 3.2. Using a front-end node, the stereo stream is scattered to a set of 3DPC Extractor nodes in a round-robin fashion. This front-end node knows at runtime how many nodes are alive at any moment. Hence, if more computing power is required, more nodes would be dynamically added (thanks to a front-end node, as seen in figure 3.3). On the other side, if less computing power is required, then less 3DPC extractor nodes would be shut down.

The performance of the platform was tested using a private cloud and using simulated stereo footage (UCA-3). Several tests were performed, which can be divided into two main categories: scalability and communication performance.

In the first category the performance speedup was measured when increasing the number of virtual resources (SCP-1, SCP-3). The results (shown in table 3.1) show that, when the resolution of the image is sufficiently high (and therefore, the amount of required computation increases as well), the solution scales properly and considerable speedup is obtained.

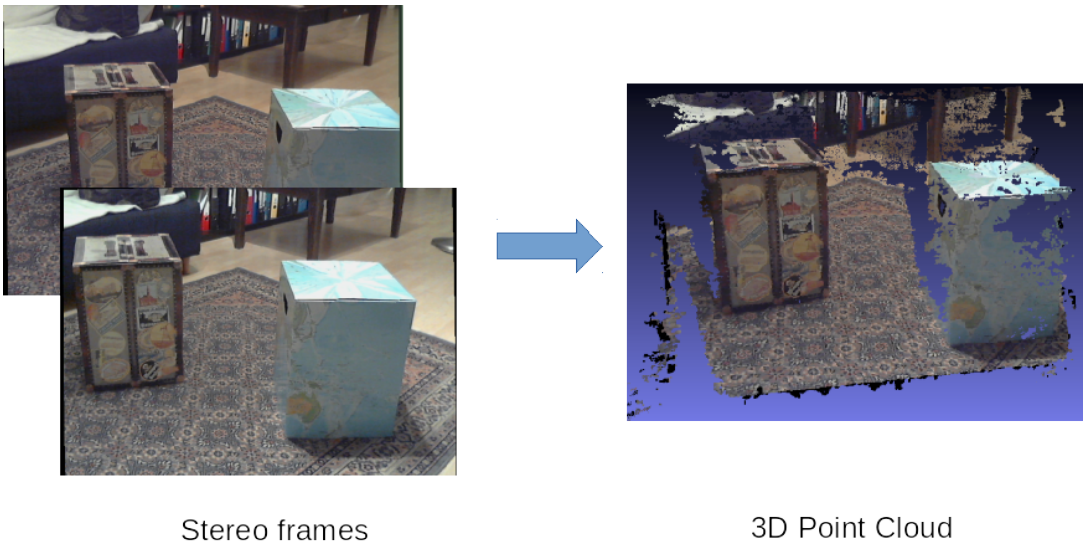


FIGURE 3.1: 3D Point Cloud Extraction from stereo frames. Images taken from https://erget.wordpress.com/2014/02/01/calibrating-a-stereo-camera-with-opencv/

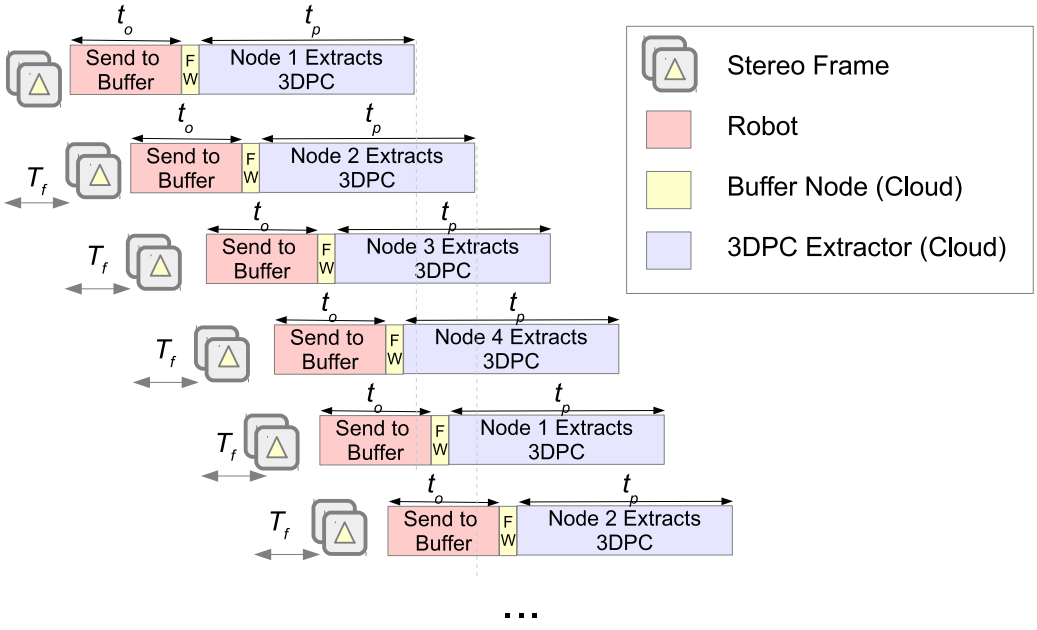


FIGURE 3.2: Stereo frame pipeline process. Four nodes process (in a pipeline fashion) the frame pairs that the front-end node delivers in a round-robin form.

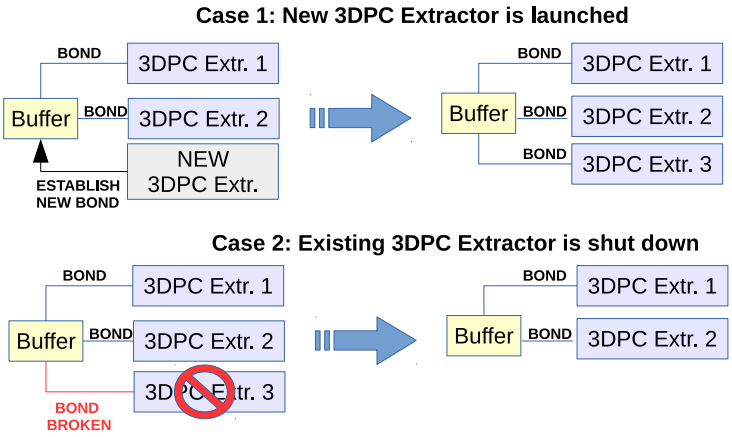


FIGURE 3.3: Dynamic adaptation of the platform when the number of 3DPC Extractors changes.

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

TABLE 3.1: Total times to process and receive n point clouds using p 3DPC extractors. Resolution of the stereo pairs is 1920x1080 and communication tehcnology used is that of Gigabit Ethernet.

Average Frequency of 3DPC reception (Hz)

	320x240	640x480	1024x768	1920x1080
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

TABLE 3.2: Performance measures for different communication technologies. 'Erratic alone' means that the Erratic robot is working alone, that is, working as a local stereo vision system.

	# 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

TABLE 3.3: Performance comparison when adding more robots in the case of 320x240 when no 3DPC extraction is done and only delays in stereo frame transmissions are considered. The wireless technology is that of 802.11ac. Note that the average time is for the messages that have arrived. Hence, a reduction in the number of successful messages

In the second category, there are several tests focused on measuring which communication technology performs better (CCT-1). In addition to this, we also compare it with the "all on-board" computation case, so it can be seen when the Cloud outperforms the embedded system. The embedded platform used for testing is the Erratic Robot by Videre LLC, with a Core 2 Duo CPU. Results show that, at the moment, Gigabit Ethernet provides the least communication penalties, with 802.11ac still far behind. This shows that current devices and software drivers are still unable to obtain the promised 867 Mbps bandwidth. However, when the resolution of the stereo footage increases, the limitations of the embedded hardware start to arise, being offloading a reasonable option, even with 802.11ac (see table 3.2).

In addition to this, an extra test was done for the case of 802.11ac (CCT-2). As the platform is expected to be used by multiple robots, it is necessary to analyze possible communication penalties when increasing the number of robots using the platform. The results show that, when increasing the number of robot nodes, communication penalties appear. There are two kind of penalties: delay in the message transfer time, and number of messages lost (see table 3.3 and figure 3.4). This shows an inherent tradeoff between the number of robots and wireless performance, and therefore solutions to mitigate this problem must be found (CCT-3). Some methods that can fin this problem are the following: changing the transport layer, changing the robotic middleware, implement TDMA methods[8]... amongst others.

Once the Stereo Vision Platform was developed and tested, a real application case was proposeed: a Cloud-based navigation assistant (UCA-2). It is meant to assist the

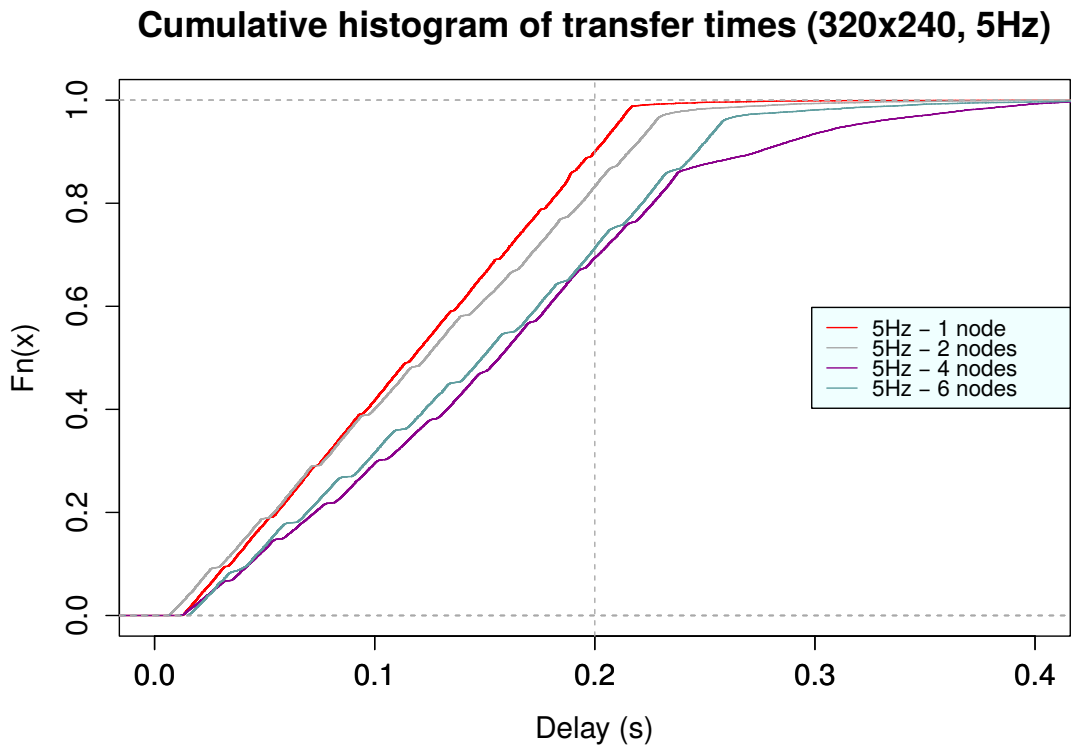


FIGURE 3.4: Empirical Cumulative Distribution Frequency of delays with 5 Hz and 320x240 stereo frames. Note that this diagram only considers the messages that have arrived.

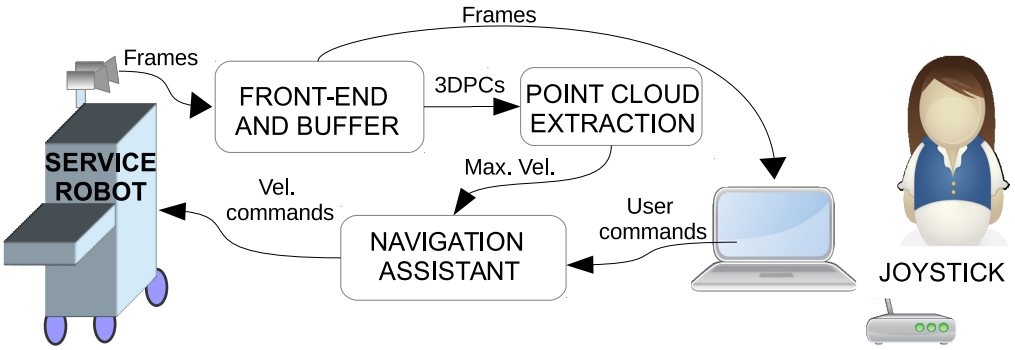


FIGURE 3.5: Block diagram of the Cloud-Based Navigation Assistant for the teleoperation of a mobile robot. The different types of data transfer are the following: Stereo frame pairs, 3D Point Clouds (3DPC), robot velocity commands ans user input commands. Also bond status messages are required by ROS bond library.

teleoperated navigation of a mobile robot by modifying the user's joysticks commands (see figure 3.5). This led to two different analyses.

To begin with, there was a timing analysis, obtaining that four main times are implied in the solution, as it can be seen in figure 3.6. The second analysis is vital for exploiting properly the Cloud properties: choosing the correct offloading model (SCP-2). As it is seen in figure 3.5, there are several modules that can be either executed on-board or offloaded to the Cloud. In this sense, when offloading to the Cloud, it is desirable to maximize the following properties:

- **Best scalability**: indicates if the involved processing can be easily scaled out or back.
- **Least communication bandwidth**: bandwidth is a precious resource in cloud computing and a source of timing delays.
- **Least virtual computing resources**: it indicates the amount of cloud resources used by an option,
- **Cheapest cloud pricing**: this must be considered when using public clouds, such as Amazon EC2.

However, there is a trade-off between those properties. In this work, four models were evaluated to see which one resulted in better performance (figure 3.7). This analysis is generic enough to be applied in other applications.

Fruit of these two analyses, new experiments were done. The first one is meant to compare the performance obtained with the different offloading models. The results (see table 3.4) showed that the best option is 1, even though option 3 seems a more common configuration for a real time system. The main reason for this is the fact that the navigation assistant is computationally expensive and needs a high amount of data (3DPCs). Because of this, when moving it to the Cloud, we dramatically decrease the consumed bandwidth, and thus obtain better performance results. The second experiment was done to emphasize another essential issue when dealing with near to real time scenarios: the delay in the information. As it can be seen in table 3.5, this is due to both communication and processing latencies. Therefore, implementing predictive timing corrections[9] is extremely necessary.

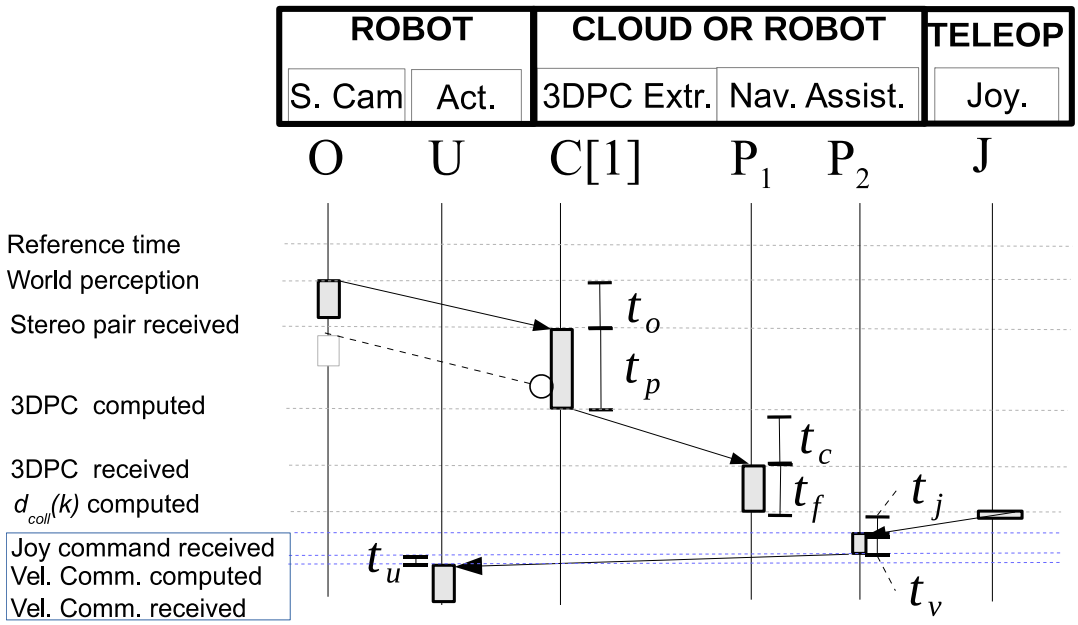


FIGURE 3.6: Time diagram of the system. Camera O captures a frame pair that sends it to the Point Clod extraction C. This extracts the 3D Point Cloud and sends it to the Navigation assistant, which fuses this information with the user command coming from the joystick J. Finally the actuators U receives the velocity commands. Different interval times increment the total latency of the system.

Frequency (Hz)

	320x240	640x480	1024x768	1920x1080
Option 1	12.55	5.72	3.73	1.06
Option 2	8.14	2.29	0.89	0.03
Option 3	7.48	2.16	0.96	0.25
Option 4	7.15	2.61	1.01	0.02

TABLE 3.4: Navigation assistant update frequencies for the different offloading models using 802.11ac WiFi.

Mean Delays (s)

	t_o	t_p	t_c	t_f	Total Comm.	Total
Option 1	0.199	0.382	0.154	0.026	0.353	0.761
Option 2	0.077	1.007	0.652	0.026	0.729	1.762
Option 3	0.511	0.382	1.223	0.064	1.734	2.180
Option 4	0.077	0.966	0.181	0.096	0.258	1.320

TABLE 3.5: Average delays of the system for 1024x768 frames and different offloading models using 802.11ac WiFi.

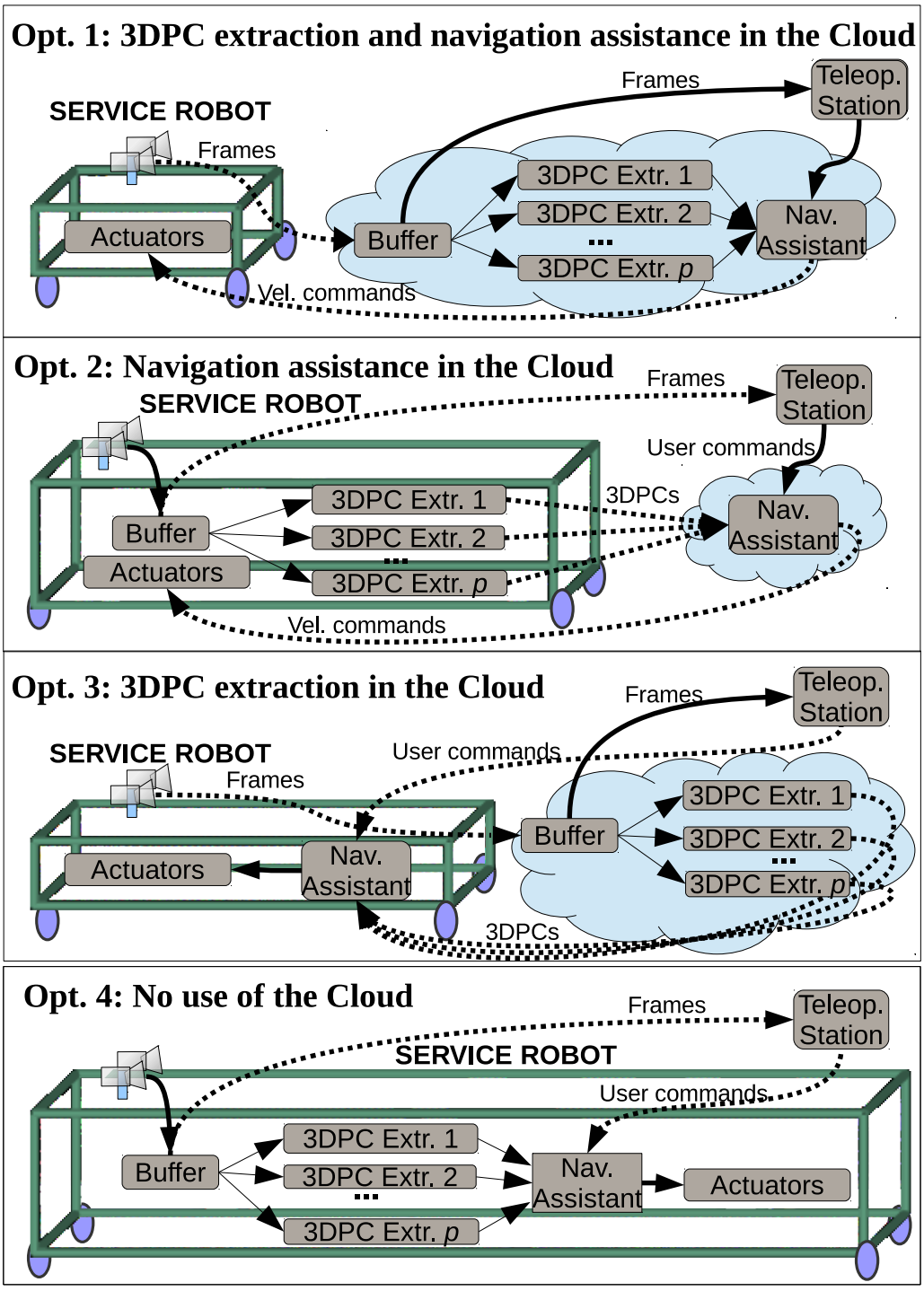


FIGURE 3.7: Possible configurations for Cloud offloading. Mandatory wireless communications (those which origin or destination is the robot) are represented by dotted arrows while communications that can be wired are depicted by continuous arrows.

Resolution	Option	# Collisions	Ratio
640x480	1	7	0.10
640x480	4	32	0.46
320x240	1	38	0.54
320x240	4	31	0.44

TABLE 3.6: Navigation results for test 2 using options 1,4 and two frame resolutions. The ratio is calculated over 70 maneuvers.

As a final test, a real navigation test is performed. This is crucial as it would empirically demonstrate not only the viability of cloud offloading, but also its potential benefits. Using a testing circuit, the Erratic robot is teleoperated using a stereo camera built from two PS Eye cameras. Along the circuit, the teleoperator tries to collide the robot to several obstacles, and the navigation system should stop the robot accordingly. Two main cases are tested: with offloading and without offloading. Moreover, different camera resolutions are tested. The results can be seen in table 3.6. Several conclusions can be extracted from it:

To begin with, there is little different between on-board and cloud offloading with low resolution images. Taking this into account, if navigating with 320x240 resolution stereo frames was enough, then no offloading would be necessary. However, as it can be seen, that quality of the stereo frames is not enough for the navigation. However, when the resolution of the stereo frames is increased, the quality of the navigation increases. This has been extensively demonstrated in the literature[10], [11].

The final and most important conclusion is that, as it was already shown with the simulated data, when the resolution of the stereo stream increases, the robot's hardware is unable to extract the 3DPCs and assist navigation in time. Therefore, the number of collisions are inevitably high. On the other side, thanks to the use of Cloud offloading, the collision rate was dramatically reduced (around 460%). Obviously, sometimes a delay in the information can lead to collisions. Nevertheless, it must be pointed out that 802.11ac has not reached yet to its full potential, so even better results can be expected for this experiment in the future.

Theoretical analysis of Cloud Offloading

Apart from empirically outlining Cloud Offloading current advantages and issues, in this thesis, a deeper approach in the concept of computation offloading is done. From a theoretical point of view, a general offloading architecture for robotic tasks is proposed (DTF-1), which can serve as a reference for developers when building applications that require the computing power that the Cloud provides (see figure 3.8). In addition to this, a first theoretical analysis regarding the effectiveness of Cloud offloading is done (DTF-2), which first result can be summarized as:

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

where:

- t_{local}: Execution time of a robotic task on-board.
- t_{remote}: Execution time of a robotic task when offloading to the Cloud.
- N_I: Number of instructions of the robotic task.

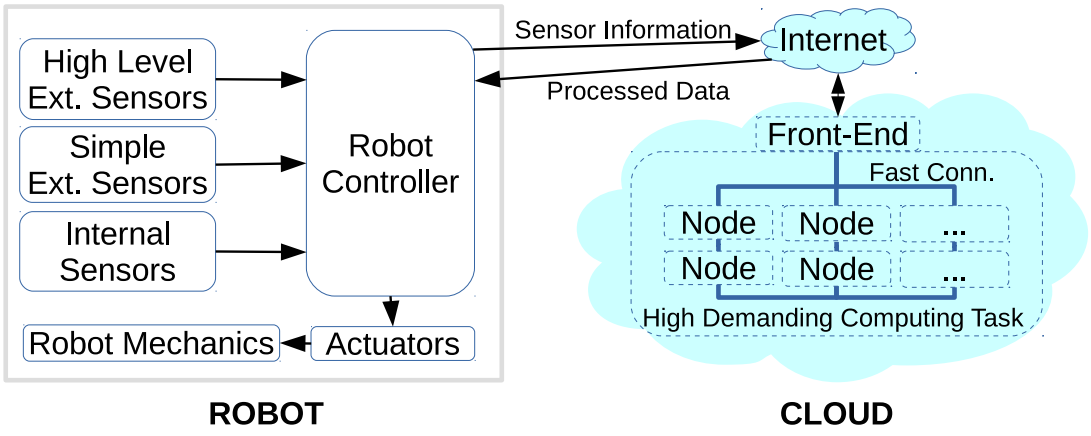


FIGURE 3.8: Block Diagram of a Cloud-based computation offloading system

- N_{Data}: Data to be sent when offloading to the Cloud.
- IPS: Instructions per Second that local embedded system can execute. It is assumed that each processor available in the Cloud, due to its vast processing power, can execute at least this value of instructions per second.
- BW: Network bandwidth.

As it can be seen, not every single task should be offloaded to the cloud. It must be taken into account the amount of available bandwidth, the data to be sent, the number of instructions and the speed of the embedded system. Looking at the inequation, two main elements can be defined:

- $D_I = \frac{N_I}{N_{Data}}$: This is the "computing density" of the application. This refers to a relationship between the number of instructions executed and the number of bits (both data and code) transferred. Ideally, the Cloud would obtain more benefits with high values of this fraction, that is to say, it computes a lot for every bit of data to be sent. Hence, the more instructions to execute and the less data to transfer, the more beneficial for Cloud Offloading. Obviously, these two variables that form the fraction are entirely task-dependent. For instance, a task that needs high resolution frame processing will presumably have bigger N_{Data} than a task that simply requires a joystick command, which only occupies few bytes. This same reasoning can be applied to the N_I variable. For instance, a stereo vision task that uses Graph Cuts with Occlusions, with $O(n^{11}))$ time complexity, will presumably have a bigger value of N_I than a stereo vision task that uses Dynamic Programming, which has $O(n^3)$ time complexity [12] (assuming that both use the same input data).
- $\mu = \frac{IPS}{BW} = (CPI \cdot T \cdot BW)^{-1}$: This relationship refers to the execution power of the local embedded system and the available network bandwidth. In this sense, Cloud Offloading would benefit if fast networks are used. On the other side, the less instructions per second can the embedded system execute, the more benefits will obtain from the Cloud. In contrast with $\frac{N_I}{N_{Data}}$, this fraction is almost completely technology-dependent. In other words, if a task has a certain value of N_I and D, this will have little effect in the value of IPS and BW, as the first one depends on the embedded hardware and the second in the network connection.

Taking a step further, an extension of Amdahl's Law for Cloud Computing is presented (DTF-2). Amdahl's law is expressed as follows:

$$S = \frac{1}{(1-F) + \frac{F}{S_f}}$$

where:

- F: Fraction of the program can that be parallelized.
- S: Global speedup of the program.
- S_f: Speedup obtained by parallelizing the fraction F of the program.

Firstly, an analysis is done for the case of computation speedup. In this sense, the following extension is proposed:

$$S_t = \frac{\frac{F}{N_{c,local}} + (1-F)}{\frac{\mu}{D_I} + (1-F)}$$

where, apart from the all the aforementioned variables, the number of available cores in the local embedded system $N_{c,local}$ is considered.The more cores the local system has, the more parallelism it can exploit. On the other side, it is assumed that the amount of cores available in the Cloud $N_{c,cloud}$ tends to infinity.

After analyses and simulations, several results are obtained, which help to make predictions on the future of Cloud offloading (DTF-3):

1. Total cloud execution time depends strongly on the amount of communication and computation times that can be overlapped. Therefore, middleware solutions for Cloud Offloading should focus in that matter.
2. $\frac{\mu}{D_I}$ plays a crucial role in speedup. In this sense, due to the expected development in network bandwidth, together with the stabilized value of $CPI \cdot T$, it can be assured that μ will progressively decrease. Hence the amount of achievable speedup will increase, making the offloading choice more desirable.
3. As long as $\frac{\mu}{D_I}$ decreases in the future, simplifying hardware may set a trend. This implication is vital for the development of the Internet of Things (IoT).

In addition to this, a second analysis is done for the case of energy efficiency, which is essential for embedded systems. In this sense, it is also essential to know whether to offload an application to the cloud saves energy in the local device. For this reason, an extension of Amdahl's law for energy has been devised:

$$S_{txE} = \frac{E_{local} \cdot t_{local}}{E_{cloud} \cdot t_{cloud}} = \frac{[(1-F) \cdot (N_{c,local} - 1) \cdot k_{idle} + 1] \cdot P}{\frac{\mu}{D_I} \cdot P} \cdot \frac{\frac{F}{N_{c,local}} + (1-F)}{\frac{\mu}{D_I} + (1-F)}$$

Just like the case of computational time speedup, some predictions can be made from the obtained results:

1. In general, cloud offloading would not be beneficial for applications with low D_I, for instance those that do not reuse input data like video or audio streaming.

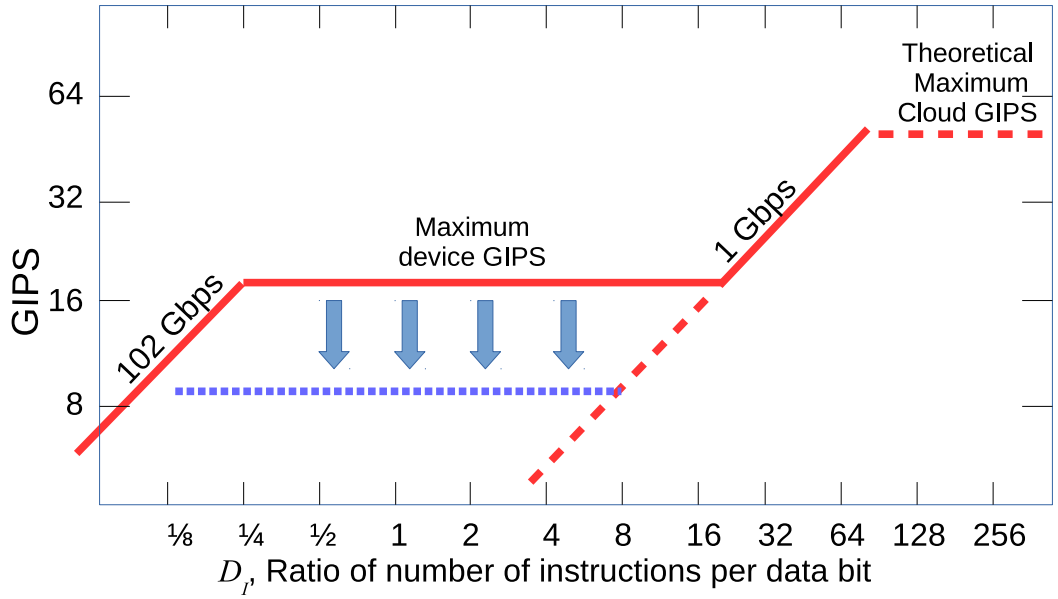


FIGURE 3.9: Two-Roofline Model for GIPS (Giga Instructions Per Second) vs. D I for the Snapdragon 610 S4 Pro. The diminution of maximum device GIPS is marked with down arrows.

2. If μ continues with its expected reduction, these bounds will decrease close to a proportional rate. Therefore, future evolution for current technology is promoting offloading computation for embedded devices.

3. - When $F \to 1$ or $N_{c,local} = 1$, the energy speedup is proportional to $S_{txE} = \left(\frac{D_I}{\mu}\right)^2$. For simple devices or very parallel applications, energy efficiency of cloud migration is expected to be reached way earlier than that of timing speedup (quadratic order).

In addition to this, this papers analyzes several example applications (DAXPY, matrix multiplication, and stereo vision), obtaining the following results:

1. For many applications, F and D_I grow with the order of the problem. As a consequence, as new applications require more accurate solutions, using the cloud becomes a more viable solution. Moreover, future research should focus on trying to reduce N_{Data}.

2. When $F = 1$, the maximum achievable performance can be seen using an extension of the Roofline Model for GIPS (see figure 3.9). This helps to conclude that some applications, which have poor CPI and real GIPS are far from the maximum, may obtain benefits by moving its computation to the Cloud (as local device's maximum theoretical GIPS cannot be achieved).

Even though the real world is much more complex than a theoretical model, these Amdahl's law extensions did help to encompass the main trends and have an idea of what the future holds for Cloud Offloading.

Chapter 4

Discussion

Undoubtedly, Cloud Computing has become one of the top trending computing technologies in this decade[13], even though it is still at a relatively early stage. Cloud Computing is more than virtual web servers and storage (basically, this is for what it is currently known for). The Cloud still has an enormous potential to offer in the following decades, not only for humans, but also for embedded systems. Because of this, the rising interest of applying the Cloud in virtually every area is more than justified.

Moreover, this is especially interesting in the area of robotics (Cloud Robotics), which is imagined like "offload compute-intensive tasks like image recognition and voice recognition and even download new skills instantly, Matrix-style" [14]. However, is it Matrix-style robotics far-fetched? Until what extent is feasible? As stated in section 1, this work aims to give answers to this question. But, has this thesis succeeded in bringing more light to the topic? In this chapter, this last question will be discussed.

To begin with, this work has not been the first one in trying to demonstrate the validity of Cloud Computing for robotics. In this sense, several works can be found in the literature since approximately 2010 (even though the dream of Cloud Computing dates from decades ago[1]). In [15] an overall analysis of the Cloud Robotics area can be seen. Its general architecture, applications and technical challenges are described. About the architecture, it defines three cloud models: peer-based model (the one used in this thesis), proxy-based model and clone-based model. Regarding the technical challenges, the following are enumerated: high communication failure rates, virtual resource optimization, security and truth establishment and optimal energy consumption.

One of the best known projects was the Roboearth project [4], which had as a main goal the creation of an "Internet for robots". Indeed, by having a cloud-based knowledge base where robots can share information, robots can not only perform collaborative tasks successfully, but also extend their capabilities by downloading new knowledge. Therefore, there are several areas where this cloud feature is being exploited[3], [16], such as human-robot interaction [17], healthcare [18] or world representation [19]. Moreover, this knowledge base helps the Cloud to become a global planner, as seen in [20] and [21]. These works, amongst others, show the great potential of knowledge sharing in different areas: healthcare, energy and transportation. Moreover, following the Roboearth approach, new global knowledge platforms are arising, like the RoboBrain project, which will learn not only from robots, but also from computer simulations and Internet resources[22].

In this sense, this is simply a natural extension of the interconnected world mankind already enjoys. Nowadays, the necessity and effectiveness of interconnected embedded systems is so taken for granted that no thesis would be necessary to prove it. However, this is not that clear in the area of computation offloading.

Indeed, its the main advantage is clearly seen: cheaper embedded devices could potentially perform complex tasks, thus saving money and energy. Nevertheless, as it

can be read in [14]:

> "But Laumond and others note that the cloud is not the solution to all of robotics' difficulties. In particular, controlling a robot's motion- which relies heavily on sensors and feedback - won't benefit much from the cloud."

The question is: is this statement axiomatic? will never Cloud offloading be beneficial when dealing with near to real time tasks? In this sense, together with this thesis, several works can be found striving to show for what cases this statement is not true:

For instance, in [23] cloud offloading is applied to traffic light status detection based on image processing. Instead of robots, this work makes use of mobile phones which send camera frames to a Cloud server, responsible of detecting the status of the traffic light. Even though the task has time constraints, the Cloud is able to respond successfully in less than a second, especially by trying to reduce the amount of transferred data thanks to image compression. This relates to the necessity of reducing N_{Data} that was mentioned in section 3.

Another example of image processing offloading can be seen in [24], where a cloud-based face detector is implemented. This system was implemented in a Pioneer P3-DX robot and wireless communication. Though their results are at an early stage (for instance, no performance figures are provided), offloading works sucessfully in this task.

A widely used robotics application is that of Self Localization and Mapping (SLAM). Therefore, several works try to prove the viability of offloading this task. In [25], GPUs are combined with the virtual resources of a national datacenter, with everything connected using Gigabit Ethernet. Thanks to this, high resolution stereo footage (1224x1024) at 15 frames per second was successfully processed. Moreover, This paper points virtualization as one of the main sources of delays. In [26] the main difference resides in the communication technology used: wireless for the robot and fabric for the Cloud. This solution is based on landmark databases, which can be helpful for exploiting parallelism. This work states that using Cloud offloading increases the throughput compared with their previous on-board solution. This papers points as future research the necessity of examining the limits of network technologies, which is exactly what this thesis strives to analyze (see section 3). In the case of [27], a step further is taken, as a complete Platform-as-a-Service (PaaS) solution is here developed. This platform, Rapyuta, is general enough to be applied to any robotic task. On the other side, the dynamic scalability property is not exploited, as it is implemented using a Clone-based model[15], hence the parallelism is exploited inside a virtual machine. As a consequence, it would be necessary to shut down and respawn a bigger virtual machines in order to get more resources. A very interesting contribution in [27] is the combination of not only computation offloading, but also knowledge sharing, as a map is collaboratively built by two robots using 802.11n technology. In comparison with this thesis, Rapyuta also outlines the limitations of current 802.11n (802.11ac was not tested in this work) wireless devices in terms of throughput, proposing optimizations in N_{Data} such as keyframing and compression techniques.

Apart from Rapyuta, other Cloud Robotics Platforms can be found in the literature. Continuing with the SLAM task, the C^2TAM platform offers SLAM as a service [28]. They combine both computation offloading and collaborative work, as the framework allows fusing the information obtained from several robots. Thanks to the use of keyframes, N_{Data} is reduced, obtaining a data flow of 1MB/s with 640x480 RGBD cameras, perfectly assumable by the wireless technologies used in the paper. Another

robotics offloading platform is the DaVinci Platform[29]. Just like Rapyuta, it is meant to be used with any robotic algorithm, as long as it uses the Map-Reduce approach. In order to validate the platform, the offloading of the FastSLAM algorithm was tested using a dataset. Just like this thesis, the scalability of the cloud was demonstrated, obtaining considerable speedup when increasing the number of nodes. Compared with this thesis, their performance figures are better because FastSLAM is embarrasingly parallel. However, they did not consider the effects of network latencies or dynamic scalability. This can be put down to the fact that they were not dealing with task with real-time constraints. Another similar platform is that of REALcloud [30] which, compared to Da-Vinci, has security and user authentication as a major asset. In addition to these platforms, [31] presents the Robot-Cloud framework, which is also based in a Hadoop Cluster. It contains high level protocols and services that allow a robot to choose between several robotic algorithms. Just like Da-Vinci, it performs several scalability tests but using less compute nodes. In this sense, can be seen that the speedup results degrade when N_{Data} increases (in the case of three compute nodes, the speedup drops from 3x to 2x when doubling N_{Data}).

Another example of robotic task with real-time constraints is that of object detection and tracking. In [32] this task is offloaded to the cloud using a decision based system. This is done by estimating both communication (802.11g is used) and computation times required to perform the task. This is related to the Amdahl's law extension presented in this thesis. In this sense, the paper also considers bandwidth (denoted as β) and server speed (denoted as η) as important factors to take into account when offloading. Therefore, future work should try to apply this extension to their communication and computation estimations. In [33] high resolution SIFT-based object detection is speeded up by transmitting on-board preprocessed image information instead of raw image data to external servers (once again, this is a way of reducing N_{Data}). The configuration of these external servers is specific to this work, so some properties of the cloud computing paradigm are not exploited. In addition to this, authors of [34] present an object-tracking scenario for a 14-DOF industrial dual-arm robot using a UDP transport protocol for transmitting large-volume image over an Ethernet network. Thanks to the very low sending and cloud image processing times that are achieved, a stabilizing control law can be implemented, with time-varying feedback time delay.

In addition to this, cloud offloading has been applied to grasp analysis and planning. Using a Hadoop Cluster in the Cloud, [35] offloads this task, obtaining successful results when increasing the number of nodes (they tested up to 500 processing nodes). Moreover, in order to increase performance, they propose using asynchronous parallelism, that is, to only wait for a subset of processing nodes to finish rather than waiting for all of them.

Apart from new results to compare, this thesis does add new contributions to the area. To begin with, when reading most of the aforementioned works, a question inevitable arises: why calling it Cloud and not Cluster, Grid, or simply Server? There are already a vast amount of research works dealing with distributed systems, and most of the offloading tasks perfectly could be done in a simple server (actually, this is what several of the aforementioned works do). Is this simply a nomenclature change or really the Cloud has something else that previous distributed systems had not? This thesis defends that the Cloud does bring an added value, and properties like infinite resources or dynamic scalability should always be taken into account. In this sense, the platform here proposed was developed with this idea in mind: to be able to adapt itself to different computing needs at runtime. Hence, the classical distributed applications design pattern may not exploit the cloud's full potential.

However, having accepted that dynamic scalability is one of Cloud's major assets, why would this be desirable in the case of embedded systems offloading? Why would a robot want the Cloud to scale back instead of having full processing power the whole execution? Reasonable question, if in the wrong era. Nowadays, having a vast computing system as the Cloud for just one robot sounds illogical. The idea would be to have tens, even hundreds of robots (or embedded systems) offloading tasks. Moreover, even though the Cloud is theoretically supposed to have infinite resources, the actual reality is quite different. Therefore, it is vital to transparently balance computing resources trying to maintain the best Quality of Service. As it can be seen, this differs little from what the Cloud is currently providing to enterprises and end-users, why would it be different for the case of robots? Hence, if we really aim for realistic scenarios, then talking about adaption and elasticity is simply mandatory.

Following this idea, when dealing with a multiple-robot setup, it is vital to define proper scheduling and resource allocation strategies, in order to optimize both CPU and bandwidth usage. In this sense, [36] proposes a scheduling mechanism for multi-sensor data retrieval. It is based on market-based management strategies and it allows to set Quality-of-Service (QoS) criteria. Results show that time of response, reliability of response, bandwidth usage and CPU load are improved when using their proposed mechanism. This strategy could be extended for dealing with dynamic scalability, so the system is able to decide when to scale out and back. In this sense, there are other real-time scheduling strategies that could be used to extend the Stereo Vision Platform implemented in this thesis, for instance: QoS-aware scheduling [37] or scheming considering temporal overlapping [38]. Therefore, future work of this thesis should focus on studying, implementing and extending these different heuristics.

Another contribution of the thesis is the application case: navigation assistance. Few of the aforementioned previous work has tried to study offloading in such a reactive and delicate task (where small errors do compromise the integrity of a robot). In [39] a theoretical cloud framework for navigation, but no implementation or real tests are provided. A more empirical approach is that of [30], where REALcloud is used by a robot to follow a line according to the images acquired from a single low resolution (320x240) camera that points to the floor. However, this thesis takes a step further and deals with a more complex approach, which has teleoperation and collision avoidance. After the experiments detailed in the thesis, we can assure that a robot has obtained better results using the Cloud than by processing on-board. However, it must be admitted that the developed solution still needs more robustness, as some collisions were unavoided. Nevertheless, this thesis did not intend to ditch on-board computing in favor of cloud offloading. This is not a matter of discarding, it is a matter of combinating. That is the main reason why this thesis also emphasizes the necessity of offloading configurations. Therefore, choosing carefully what to offload is the key to success in virtually any application case, even those with real time requirements. In this sense, in the area of mobile code offloading, there are several frameworks that try to analyze which parts of an application should be offloaded [40]. Most are based on code profiling, either automatic, like the COMET framework [41] or manual, like the ThinkAir platform [42]. However, this is not without issues and challenges [40] like inaccurate code profiling or integration complexity. In any case, extending these frameworks for robotic applications and, even more important, for real time constraints is a very interesting and challenging area of research. In addition to this, when it comes to offloading, the principles of multi-agent robotics systems (MARS) should applied [43]: modularity, concurrency, distributed architecture... amongst others. This would make the process of developing and offloading applications much easier thanks to these properties.

Closely related to this, we cannot detach computation offloading from another vital concept: communication. In this sense, this thesis offers a more in-depth analysis in both the key and state of current technologies, especially 802.11ac. If we really aim for successful offloading when having time constraints, delays in information must be thoroughly analyzed. Admittedly, the performance results shown for 802.11ac, even though its bandwidth is expected to be close to 1 Gbps [44], are still far from being "the Gigabit Ethernet equivalent of wireless networks". However, this thesis helped to turn the spotlight on future research works:

- How can the variability of transfer times be reduced? Are the TCP and UDP layers enough for task offloading? As the TCP transport layer is not prepared for real time systems, it might be time to study and develop enhancements (or other alternatives alternative layers) especially focused in this kind of tasks. In this sense, further improvements in *802.11ac* MAC layer like TDMA protocols, would reduce substantially this latency variance [8]. Moreover, Contention Free Period in the infrastructure mode with fixed size packets could guarantee a minimum bandwidth reservation, which may be suitable for many real time systems. If we really aim for successful offloading when having time constraints, some QoS mechanisms should be incorporated to allow differentiation between traffic types and guarantee allocation of resources to the offloaded applications with real time requirements. These QoS mechanisms can be classified into two main categories:"Traffic handling mechanisms" or "bandwidth management mechanisms" [44] . Traffic handling mechanisms are designed to classify, handle, and monitor the traffic across the network, while bandwidth management mechanisms manage the network resource and coordinate the network devices. Typically, mechanisms from the two categories can be used together: for instance, packet classification may be used to distinguish between different kinds of traffics and then an admission control mechanism takes a decision of accepting or not the incoming flow according to its QoS requirements and the available network resources [44] . With IEEE 802.11ac, enhanced distributed channel access (EDCA) is the primarily adopted access control mechanism for multimedia traffic transmission, so it seems that it could also be used when offloading applications with real time requirements. However, its inefficient backoff procedure may introduce unpredictable delays [45]. There are still many open research issues with regard to this important question.

- To what extent non real time middleware and variable data transfers affect a soft real time system? Not only current hardware must be enhanced, but also the available software and frameworks like ROS should have a focus on real time requirements.

- What happens if suddenly the communication is broken? A robot that relies solely in the Cloud lacks enough robustness. Hence, offloading solutions with fallback mechanisms in case of network fault should be studied and analyzed.

- Can the negative effect of delayed information be circumvented by using predictive time corrections[9]? If a predictive method is implemented, then robots may afford to use delayed information even with reactive applications.

Apart from all this, computation offloading cannot avoid the following questions: is this just a short-term trend? will embedded hardware develop with such a pace that

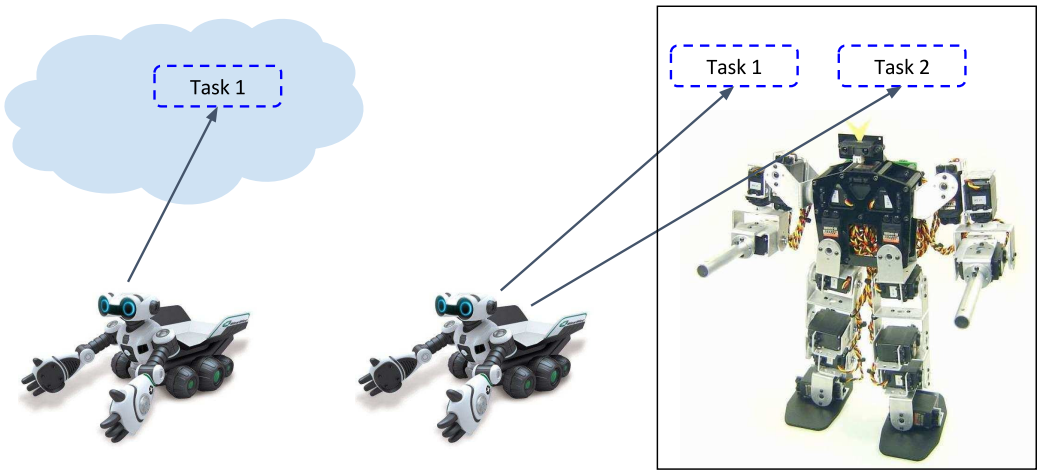


FIGURE 4.1: Concept of Peer to Peer Cloud applied to Robotics. The leftmost robot makes use of the computational resources in the Cloud but, instead, the second robot makes use of the resources of a third robot

renders Cloud Offloading useless? In this sense, [46] stated in that Cloud Offloading can be beneficial for some applications. As a very interesting idea, it must be taken into account that if the data is already stored in the Cloud, no data transfer would be necessary, only a pointer to it. That implies considerable energy savings. Just like [46], this thesis did not hide and strived to provide answers, not only in the case of energy but also in the case of performance. That is the reason why another contribution was a deeper study in the concept of offloading, having as a result the extension of Amdahl's Law for Cloud Computing. Thanks to this research, it could be seen that Cloud Offloading will have its place due to both current technology and software trends. Moreover, it helps developers and researchers to have an idea of whether a certain computation should be offloaded or not. In addition to this, this extension of Amdahl's law shows a tendency: in the future, computation offloading will be more and more beneficial. Therefore, another question arises: what is the role of other computing platforms like embedded Graphical Processing Units (GPUs)? If, down the line, offloading will provide better performance and energy results, then maybe embedded GPUs could be the technology with an expiry date (obviously, GPUs could be of great use in the Cloud, but probably not in the embedded system). Furthermore, Field Programmable Gate Arrays (FGPAs), which offers extraordinary performance results (thanks to its hundreds of available functional units) are also increasingly becoming more energy efficient. Hence, it can be assured that, in the future, two computing choices will prevail: very simple devices that use computation offloading (this is very related to the rise of Internet of Things [47] and Cloud Computing, known as Cloud of Things [48]) and reconfigurable computing devices.

Of course, in order to provide the Amdahl's Law extension and its inherent predictions, it was necessary to make some simplifications and assumptions. Not to mention the fact that, for complex applications, obtaining the parallel fraction F can be extremely challenging. However, is not less true this was the necessary first step in order to develop a deeper and more ambitious theoretical framework. In this sense, future research could study and model issues like virtualization, middleware and OS overheads, task scheduling, amount of overlap between transfers, heterogeinity of hardware resources between cloud and the embedded system... amongst others.

In addition to this, this thesis' results point where future research and development

should focus on, for instance trying to increase D_I and reduce N_{Data} of applications by developing new software and compressing techniques. Not only this is important for performance, but also for energy efficiency. As it was mentioned before, this effort to optimize these values can be seen in several works [23], [27], [28], [32], [33]. Moreover, in cases where the data to be sent are images (just like this thesis), there are other works that propose energy-efficient compression techniques which are worth applying in Cloud Offloading. For instance, [49] proposes a bandwidth aware compression method, obtaining a 20% energy reduction average. In addition to this, [50] proposes a method based in dynamic switching of compression level based on the available bandwidth, thus optimizing data transfer. Apart from this, it is important to study cross-layer transmission techniques to, once again, optimize communication and energy savings [51].

For all this reasons, it can be assured that this thesis, with its issues and limitations, not only provided experimental contributions to the scientific community, but also it is a first step in the development of a future Cloud Offloading theoretical framework.

As a final discussion item, there is another element that should be taken into account: practically all the mentioned works, including this thesis, treat the cloud as a centralized platform that robots make use of. Hence, another question arises: is this enough? Should we go a step further and make robots "part of the Cloud"? This is especially related to the concept of Peer-to-peer Cloud [52]. Let us imagine a set of heterogeneous robots performing several tasks. If one of the robots has plenty of computational resources, these could be offered to the rest of robots for offloading (see figure 4.1). Thanks to this, robots would have more flexibility when choosing where to offload their tasks. In this sense, [53] proposes the Robot-as-a-Service framework, where thanks to the use of web 2.0 technologies and Service Oriented Architecture (SOA), a robot would be Cloud Computing Unit, just as Software, Platform or Infrastructure. This idea is extended in several works. For instance, [54] integrates the use of SOA and Map-Reduce algorithms in a Hadoop Cluster. In [55] an algorithm for providing robotic resources in order to satisfy a certain QoS is proposed. In addition to this, in [56] the RoboWeb architecture is created. This architecutre exposes both robot's software and hardware and exposes them as services through the Web. Another related concept is the term "Robotic Cluster" introduced in [57]: "group of individual robots which are able to share their processing resources, therefore, the robots can solve difficult problems by using the processing units of other robots". In this sense, [58] applies this concept to the use case of SLAM, developing two archictectures (stateless and stateful) in order to cope with the limitations of bandwidth. The speedup results obtained are promising, hence the combination of robotic clusters, robots as a service and centralized approaches would extend Cloud Offloading capabilities. An example of this can be seen in [59], where an engine for collaboration between networked robots and public clouds is developed. This platform makes decisions of where each task should be executed (either in the robots or in the Cloud). At the moment only two policies have been implemented: minimum energy and minimum budget (as it is using a public cloud, using its resources cost money). However, it is a perfect starting point for future work in integrating both centralized cloud and cluster of robots. Therefore, as it can be seen, the definition of Cloud Computing keeps evolving through the years, and in this sense, the results of this thesis strived to take a step forward in this direction.

Chapter 5

Conclusions

- Cloud Offloading can be beneficial even in applications with near to real time applications. It is here demonstrated that the teleoperation of a mobile robot using a cloud-based navigation assistant (using stereo-vision obstacle detection) provided better quality than its all-on-board counterpart.
- Not all applications may benefit from Cloud Offloading. Several parameters must be taken into account: data size, bandwidth, local resources, potential energy savings, response time, characteristics of the application... amongst others. In this sense, the most important parameters are the density of instructions $D_I = \frac{N_I}{N_{Data}}$ and the technological factor μ, which depends on the execution power of the local embedded system and the available network bandwidth.
- Current 802.11ac wireless devices are still unable to provide speeds close to Gigabit Ethernet. However, they perform better than 802.11n devices. If future developments of WiFi AC obtain throughputs close its theoretical 867 Mbps, substantial speedups could be obtained.
- Benefits of offloading increase with D_I/μ linearly in performance but quadratically in energy.
- Researchers and software developers should target in increasing D_I and reducing N_{Data} by compression and software techniques.
- There is an inherent trade-off between computation and communication in Cloud Offloading. Therefore, when developing applications, all the different offloading configurations (that is, which modules must reside on board or be offloaded) must be analyzed, especially the implied transfers. Not every offloading configuration provides successful results.
- Trends indicate that μ will decrease in the future. As D_I and F (parallel fraction) grow with the order of the problem, Cloud Offloading will become more and more beneficial as years go by. Therefore, the full offload of computationally expensive applications will become preferable, even for tasks that nowadays provide better results when running on-board.

  Future work:

- When offloading tasks with time constraints, a big issue is the delay in the processed information. In order to mitigate this effect, predictive time corrections should be studied and developed.
- Current middleware and transport layer technologies introduce high variability in the information transfer. Research should focus on studying the performance given by alternative standard and non-standard technologies and platforms.

Chapter 6

Bibliography

[1] Simson Garfinkel and Harold Abelson. *Architects of the information society: 35 years of the Laboratory for Computer Science at MIT*. Cambridge, MA: MIT Press, 1999. ISBN: 978-0-585-05488-9 978-0-262-57131-9 978-0-262-07196-3.

[2] James Kuffner. "Cloud-Enabled Robots". In: *IEEE-RAS International Conference on Humanoid Robots*. Nashville, TN, 2010.

[3] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. "A Survey of Research on Cloud Robotics and Automation". In: *IEEE Transactions on Automation Science and Engineering* 12.2 (Apr. 2015), pp. 398–409. ISSN: 1545-5955. DOI: 10.1109/TASE.2014.2376492.

[4] M. Waibel, M. Beetz, J. Civera, et al. "RoboEarth". In: *IEEE Robotics Automation Magazine* 18.2 (June 2011), pp. 69–82. ISSN: 1070-9932. DOI: 10.1109/MRA.2011.941632.

[5] Javier Salmerón-García, Pablo Iñigo-Blasco, Fernando Díaz-del Río, and Daniel Cagigas-Muñiz. "Study of Communication Issues in Dynamically Scalable Cloud-Based Vision Systems for Mobile Robots". en. In: *Robots and Sensor Clouds*. Ed. by Anis Koubaa and Elhadi Shakshuki. Studies in Systems, Decision and Control 36. Springer International Publishing, 2016, pp. 33–52. ISBN: 978-3-319-22167-0 978-3-319-22168-7. URL: http://link.springer.com/chapter/10.1007/978-3-319-22168-7_2 (visited on 09/14/2015).

[6] J. Salmeron-Garcia, P. Inigo-Blasco, F. Diaz-del Rio, and D. Cagigas-Muniz. "A Tradeoff Analysis of a Cloud-Based Robot Navigation Assistant Using Stereo Image Processing". In: *IEEE Transactions on Automation Science and Engineering* PP.99 (2015), pp. 1–11. ISSN: 1545-5955. DOI: 10.1109/TASE.2015.2403593.

[7] F. Diaz Del Rio, Javier Salmerón-García, and J.L. Sevillano. "Extending Amdahl's Law For the Cloud Computing Era". In: *IEEE Computer Magazine* (), To appear.

[8] Goncalo Martins. "Reducing Communication Delay Variability for a Group of Robots". AAI3607664. PhD thesis. Denver, CO, USA: University of Denver, 2013.

[9] Pablo Iñigo-Blasco, Fernando Diaz-del Rio, Saturnino Vicente-Diaz, and Daniel Cagigas-Muñiz. "The Shared Control Dynamic Window Approach for Non - Holonomic Semi-Autonomous Robots". In: *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*. Munich, June 2014.

[10] Ankur Handa, Richard A. Newcombe, Adrien Angeli, and Andrew J. Davison. "Real-Time Camera Tracking: When is High Frame-Rate Best?" en. In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid. Lecture Notes in Computer Science 7578. Springer Berlin Heidelberg, Jan. 2012, pp. 222–235. ISBN: 978-3-642-33785-7, 978-3-642-33786-4. (Visited on 11/11/2014).

[11] Mikko Kytö; Mikko Nuutinen, and Oittinen Pirkko. "Method for measuring stereo camera depth accuracy based on stereoscopic vision". In: *Proc. of SPIE/IS&T Electronic Imaging 2011*. 2011.

[12] Mikhail Sizintsev and Richard P. Wildes. "Coarse-to-fine stereo vision with accurate 3D boundaries". In: *Image and Vision Computing* 28.3 (Mar. 2010), pp. 352–366. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2009.06.008. URL: http://www.sciencedirect.com/science/article/pii/S0262885609001358 (visited on 08/01/2014).

[13] *Hype Cycle for Application Architecture, 2013*. URL: https://www.gartner.com/doc/2569522/hype-cycle-application-architecture- (visited on 07/31/2014).

[14] E. Guizzo. "Robots with their heads in the clouds". In: *IEEE Spectrum* 48.3 (Mar. 2011), pp. 16 –18. ISSN: 0018-9235. DOI: 10.1109/MSPEC.2011.5719709.

[15] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. "Cloud robotics: architecture, challenges and applications". In: *IEEE Network* 26.3 (June 2012), pp. 21 –28. ISSN: 0890-8044. DOI: 10.1109/MNET.2012.6201212.

[16] P.C. Evans and M. Annunziata. *Industrial Internet: Pushing the Boundaries of Minds and Machines*. Tech Report. General Electric, 2012.

[17] Yan-You Chen, Jhing-Fa Wang, Po-Chuan Lin, et al. "Human-robot interaction based on cloud computing infrastructure for senior companion". In: *TENCON 2011 - 2011 IEEE Region 10 Conference*. Nov. 2011, pp. 1431 –1434. DOI: 10.1109/TENCON.2011.6129046.

[18] Z. Dogmus, A. Papantoniou, M. Kilinc, et al. "Rehabilitation robotics ontology on the cloud". In: *2013 IEEE International Conference on Rehabilitation Robotics (ICORR)*. June 2013, pp. 1–6. DOI: 10.1109/ICORR.2013.6650415.

[19] M. Tenorth, A.C. Perzylo, R. Lafrenz, and M. Beetz. "Representation and Exchange of Knowledge About Actions, Objects, and Environments in the RoboEarth Framework". In: *IEEE Transactions on Automation Science and Engineering* 10.3 (July 2013), pp. 643–651. ISSN: 1545-5955. DOI: 10.1109/TASE.2013.2244883.

[20] Rob Janssen, René van de Molengraft, Herman Bruyninckx, and Maarten Steinbuch. "Cloud based centralized task control for human domain multi-robot operations". en. In: *Intelligent Service Robotics* (Sept. 2015), pp. 1–15. ISSN: 1861-2776, 1861-2784. DOI: 10.1007/s11370-015-0185-y. URL: http://link.springer.com/article/10.1007/s11370-015-0185-y (visited on 10/22/2015).

[21] D. Berenson, P. Abbeel, and K. Goldberg. "A robot path planning framework that learns from experience". In: *2012 IEEE International Conference on Robotics and Automation (ICRA)*. May 2012, pp. 3671–3678. DOI: 10.1109/ICRA.2012.6224742.

[22] L. Vasiliu, I. Trochidis, C. Bussler, and A. Koumpis. "RoboBrain: A software architecture mapping the human brain". In: *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Nov. 2014, pp. 160–165. DOI: 10.1109/HUMANOIDS.2014.7041353.

[23] Pelin Angin, Bharat Bhargava, and Sumi Helal. "A Mobile-Cloud Collaborative Traffic Lights Detector for Blind Navigation". In: *2010 Eleventh International Conference on Mobile Data Management (MDM)*. May 2010, pp. 396 –401. DOI: 10.1109/MDM.2010.71.

[24] Shuqing Tian and Suk Gyu Lee. "An implementation of cloud robotic platform for real time face recognition". In: *2015 IEEE International Conference on Information and Automation*. Aug. 2015, pp. 1509–1514. DOI: 10.1109/ICInfA.2015.7279524.

[25] Kumar Ayush and Navin Kumar Agarwal. "Real time visual SLAM using cloud computing". In: *Computing, Communications and Networking Technologies (ICCCNT) , 2013 Fourth International Conference on*. Tiruchengode: IEEE, July 2013, pp. 1–7. ISBN: 978-1-4799-3926-8 978-1-4799-3925-1. DOI: 10.1109/ICCCNT.2013.6726744. URL: http://www.ieeeexplore.net/xpl/articleDetails.jsp?tp=&arnumber=6726744&searchWithin%3DSearch_Index_Terms%3A.QT.SLAM.QT. (visited on 03/18/2014).

[26] P. Benavidez, M. Muppidi, P. Rad, et al. "Cloud-based realtime robotic Visual SLAM". In: *Systems Conference (SysCon), 2015 9th Annual IEEE International*. Apr. 2015, pp. 773–777. DOI: 10.1109/SYSCON.2015.7116844.

[27] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel. "Rapyuta: A Cloud Robotics Platform". In: *IEEE Transactions on Automation Science and Engineering* 12.2 (Apr. 2015), pp. 481–493. ISSN: 1545-5955. DOI: 10.1109/TASE.2014.2329556.

[28] L. Riazuelo, Javier Civera, and J. M. M. Montiel. "C2TAM: A Cloud framework for cooperative tracking and mapping". In: *Robotics and Autonomous Systems* 62.4 (Apr. 2014), pp. 401–413. ISSN: 0921-8890. DOI: 10.1016/j.robot.2013.11.007. URL: http://www.sciencedirect.com/science/article/pii/S0921889013002248 (visited on 03/16/2014).

[29] R. Arumugam, V.R. Enti, Liu Bingbing, et al. "DAvinCi: A cloud computing framework for service robots". In: *2010 IEEE International Conference on Robotics and Automation (ICRA)*. May 2010, pp. 3084 –3089. DOI: 10.1109/ROBOT.2010.5509469.

[30] L. Agostinho, L. Olivi, G. Feliciano, et al. "A Cloud Computing Environment for Supporting Networked Robotics Applications". In: *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*. Dec. 2011, pp. 1110 –1116. DOI: 10.1109/DASC.2011.181.

[31] R. Doriya, P. Chakraborty, and G.C. Nandi. "Robot-Cloud: A framework to assist heterogeneous low cost robots". In: *2012 International Conference on Communication, Information Computing Technology (ICCICT)*. Oct. 2012, pp. 1–5. DOI: 10.1109/ICCICT.2012.6398208.

[32] Y. Nimmagadda, K. Kumar, Yung-Hsiang Lu, and C.S.G. Lee. "Real-time moving object recognition and tracking using computation offloading". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2010, pp. 2449–2455. DOI: 10.1109/IROS.2010.5650303.

[33] H. Bistry and Jianwei Zhang. "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)*. Oct. 2010, pp. 3195–3200. DOI: 10.1109/IROS.2010.5653660.

[34] Haiyan Wu, Lei Lou, Chih-Chung Chen, S. Hirche, and K. Kuhnlenz. "Cloud-Based Networked Visual Servo Control". In: *IEEE Transactions on Industrial Electronics* 60.2 (Feb. 2013), pp. 554–566. ISSN: 0278-0046. DOI: 10.1109/TIE.2012.2186775.

[35] B. Kehoe, D. Warrier, S. Patil, and K. Goldberg. "Cloud-Based Grasp Analysis and Planning for Toleranced Parts Using Parallelized Monte Carlo Sampling". In: *IEEE Transactions on Automation Science and Engineering* 12.2 (Apr. 2015), pp. 455–470. ISSN: 1545-5955. DOI: 10.1109/TASE.2014.2356451.

[36] Lujia Wang, Ming Liu, and M.Q.-H. Meng. "Real-Time Multisensor Data Retrieval for Cloud Robotic Systems". In: *IEEE Transactions on Automation Science and Engineering* 12.2 (Apr. 2015), pp. 507–518. ISSN: 1545-5955. DOI: 10.1109/TASE.2015.2408634.

[37] Georgios L. Stavrinides and Helen D. Karatza. "A Cost-Effective and QoS-Aware Approach to Scheduling Real-Time Workflow Applications in PaaS and SaaS Clouds". In: *2015 3rd International Conference on Future Internet of Things and Cloud (FiCloud)*. Aug. 2015, pp. 231–239. DOI: 10.1109/FiCloud.2015.93.

[38] K. Kumar, Jing Feng, Y. Nimmagadda, and Yung-Hsiang Lu. "Resource Allocation for Real-Time Tasks Using Cloud Computing". In: *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*. July 2011, pp. 1–7. DOI: 10.1109/ICCCN.2011.6006077.

[39] Wu Jiang, Huang Xiaona, Fan Xunli, Zhao Haichuan, and She YuJing. "Study on the Cloud Computing Supported Platform for Navigation and Positioning System". In: *2011 International Conference on Internet Technology and Applications (iTAP)*. Aug. 2011, pp. 1 –4. DOI: 10.1109/ITAP.2011.6006341.

[40] H. Flores, Pan Hui, S. Tarkoma, et al. "Mobile code offloading: from concept to practice and beyond". In: *IEEE Communications Magazine* 53.3 (Mar. 2015), pp. 80–88. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7060486.

[41] Mark S. Gordon, D. Anoushe Jamshidi, Scott Mahlke, Z. Morley Mao, and Xu Chen. "COMET: Code Offload by Migrating Execution Transparently". In: *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*. OSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 93–106. ISBN: 978-1-931971-96-6. URL: http://dl.acm.org/citation.cfm?id=2387880.2387890 (visited on 04/06/2015).

[42] S. Kosta, A. Aucinas, Pan Hui, R. Mortier, and Xinwen Zhang. "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading". In: *2012 Proceedings IEEE INFOCOM*. Mar. 2012, pp. 945–953. DOI: 10.1109/INFCOM.2012.6195845.

[43] Pablo Iñigo-Blasco, Fernando Diaz-del Rio, M Carmen Romero-Ternero, Daniel Cagigas-Muñiz, and Saturnino Vicente-Diaz. "Robotics software frameworks for multi-agent robotic systems development". In: *Robotics and Autonomous Systems* 60.6 (June 2012), pp. 803–821. ISSN: 0921-8890. DOI: 10.1016/j.robot.2012.02.004. URL: http://www.sciencedirect.com/science/article/pii/S0921889012000322 (visited on 03/19/2014).

[44] E. Charfi, L. Chaari, and L. Kamoun. "PHY/MAC Enhancements and QoS Mechanisms for Very High Throughput WLANs: A Survey". In: *IEEE Communications Surveys Tutorials* 15.4 (2013), pp. 1714–1735. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.013013.00084.

[45] C.-Y. Chang, H.-C. Yen, C.-C. Lin, and D.-J. Deng. "QoS/QoE Support for H.264 / AVC Video Stream in IEEE 802.11ac WLANs". In: *IEEE Systems Journal* PP.99 (2015), pp. 1–10. ISSN: 1932-8184. DOI: 10.1109/JSYST.2015.2431291.

[46] K. Kumar and Yung-Hsiang Lu. "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" In: *Computer* 43.4 (Apr. 2010), pp. 51–56. ISSN: 0018-9162. DOI: 10.1109/MC.2010.98.

[47] Mohammed Riyadh Abdmeziem, Djamel Tandjaoui, and Imed Romdhani. "Architecting the Internet of Things: State of the Art". en. In: *Robots and Sensor Clouds*. Ed. by Anis Koubaa and Elhadi Shakshuki. Studies in Systems, Decision and Control 36. DOI: 10.1007/978-3-319-22168-7_3. Springer International Publishing, 2016, pp. 55–75. ISBN: 978-3-319-22167-0 978-3-319-22168-7. URL: http://link.springer.com/chapter/10.1007/978-3-319-22168-7_3 (visited on 11/03/2015).

[48] Mohammad Aazam, Eui-Nam Huh, Marc St-Hilaire, Chung-Horng Lung, and Ioannis Lambadaris. "Cloud of Things: Integration of IoT with Cloud Computing". en. In: *Robots and Sensor Clouds*. Ed. by Anis Koubaa and Elhadi Shakshuki. Studies in Systems, Decision and Control 36. DOI: 10.1007/978-3-319-22168-7_4. Springer International Publishing, 2016, pp. 77–94. ISBN: 978-3-319-22167-0 978-3-319-22168-7. URL: http://link.springer.com/chapter/10.1007/978-3-319-22168-7_4 (visited on 11/03/2015).

[49] Y. Nimmagadda, K. Kumar, and Yung-Hsiang Lu. "Energy-efficient image compression in mobile devices for wireless transmission". In: *IEEE International Conference on Multimedia and Expo, 2009. ICME 2009*. June 2009, pp. 1278–1281. DOI: 10.1109/ICME.2009.5202735.

[50] M. Omote, K. Ootsu, T. Ohkawa, and T. Yokota. "Efficient Data Communication Using Dynamic Switching of Compression Method". In: *2013 First International Symposium on Computing and Networking (CANDAR)*. Dec. 2013, pp. 607–611. DOI: 10.1109/CANDAR.2013.109.

[51] Tao Ma, M. Hempel, Dongming Peng, and H. Sharif. "A Survey of Energy-Efficient Compression and Communication Techniques for Multimedia in Resource Constrained Systems". In: *IEEE Communications Surveys Tutorials* 15.3 (2013), pp. 963–972. ISSN: 1553-877X. DOI: 10.1109/SURV.2012.060912.00149.

[52] Ozalp Babaoglu and Moreno Marzolla. "Escape From the Data Center: The Promise of Peer-to-Peer Cloud Computing". In: *IEEE Spectrum Magazine* (Sept. 2014). URL: http://spectrum.ieee.org/computing/networks/escape-from-the-data-center-the-promise-of-peertopeer-cloud-computing (visited on 04/21/2015).

[53] Yinong Chen, Zhihui Du, and M. Garcí anda Acosta. "Robot as a Service in Cloud Computing". In: *2010 Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE)*. June 2010, pp. 151 –158. DOI: 10.1109/SOSE.2010.44.

[54] R. Doriya, P. Chakraborty, and G.C. Nandi. "Robotic Services in Cloud Computing Paradigm". In: *2012 International Symposium on Cloud and Services Computing (ISCOS)*. Dec. 2012, pp. 80–83. DOI: 10.1109/ISCOS.2012.24.

[55] Zhihui Du, Weiqiang Yang, Yinong Chen, et al. "Design of a Robot Cloud Center". In: *2011 10th International Symposium on Autonomous Decentralized Systems (ISADS)*. Mar. 2011, pp. 269–275. DOI: 10.1109/ISADS.2011.36.

[56] Anis Koubaa. "A Service-Oriented Architecture for Virtualizing Robots in Robot-as-a-Service Clouds". en. In: *Architecture of Computing Systems – ARCS 2014*. Ed. by Erik Maehle, Kay Römer, Wolfgang Karl, and Eduardo Tovar. Lecture Notes in Computer Science 8350. DOI: 10.1007/978-3-319-04891-8_17. Springer International Publishing, Feb. 2014, pp. 196–208. ISBN: 978-3-319-04890-1 978-3-319-04891-8. URL: http://link.springer.com/chapter/10.1007/978-3-319-04891-8_17 (visited on 11/03/2015).

[57] Ali Marjovi, Sarvenaz Choobdar, and Lino Marques. "Robotic clusters: Multi-robot systems as computer clusters: A topological map merging demonstration". In: *Robotics and Autonomous Systems* 60.9 (Sept. 2012), pp. 1191–1204. ISSN: 0921-8890. URL: http://www.sciencedirect.com/science/article/pii/S0921889012000632 (visited on 11/03/2015).

[58] Bruno D. Gouveia, David Portugal, Daniel C. Silva, and Lino Marques. "Computation Sharing in Distributed Robotic Systems: a Case Study on SLAM". In: *IEEE Transactions on Automation Science and Engineering (T-ASE)* 12.2 (Apr. 2015).

[59] P. Pandey, D. Pompili, and Jingang Yi. "Dynamic Collaboration Between Networked Robots and Clouds in Resource-Constrained Environments". In: *IEEE Transactions on Automation Science and Engineering* 12.2 (Apr. 2015), pp. 471–480. ISSN: 1545-5955. DOI: 10.1109/TASE.2015.2406115.

Part II

Set of papers

Appendix A

Mobile robot motion planning based on Cloud Computing stereo vision processing

Authors

- Javier J. Salmerón García
- Pablo Íñigo Blasco
- Fernando Díaz del Río
- Daniel Cagigas Muñiz

Publication

Title: Proceedings of 41st International Symposium on Robotics (ISR 2014).

Type: Conference Paper.

Conference Name: 41st International Symposium on Robotics (ISR 2014).

Place: Munich, Germany.

Date: June 2014.

Publisher: VDE.

Date: 2016.

Pages: 1-6.

ISBN: 978-3-8007-3601-0.

Mobile robot motion planning based on Cloud Computing stereo vision processing.

Javier, Salmerón-García, Robotics and Computer Technology Lab, Univ. of Seville, jsalmeron2@us.es, Spain
Pablo, Iñigo-Blasco, Robotics and Computer Technology Lab, Univ. of Seville, pabloinigo@us.es, Spain
Fernando, Díaz-del-Río, Robotics and Computer Technology Lab, Univ. of Seville, fdiaz@us.es, Spain
Daniel, Cagigas-Muñiz, Robotics and Computer Technology Lab, Univ. of Seville, dcagigas@us.es, Spain

Abstract

Nowadays, the limitations of robot embedded hardware (which cannot be upgraded easily) make difficult to perform computationally complex tasks such as those of high level artificial vision. However, instead of disposing these "outdated" embedded systems, Cloud technologies for computation offloading can be used. In this paper we present and analyze an example of computation offloading in the context of artifical vision: point cloud extraction for stereo images. A prototype prepared for exploiting the cloud's unique capabilities (such as elasticity) has been developed, and the inherent issues that appears are explained and addressed.

1 Introduction

New computationally expensive tasks are expected to be performed with relative fluency for 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, along with others.

An interesting candidate for the aforementioned "external computer system" is that of a Cloud infrastructure, thanks to its inherent characteristics: high reliability, larger storage capacity, stable electric power, reutilization of hardware, dynamic scalability and better resource utilization. In particular, the dynamic scalability property (adaptation of the computing power at runtime, that is, scaling out and back, depending on the needs) is very useful in robotics, because it will allow the incorporation of more computation demanding algorithms as they are being implemented. 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[1], [2]. 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 (see section 2).

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 [3], etc.

In this paper we address the "computation offloading" of an intermediate robot level task: 3-D point cloud extraction from stereo image pairs. Point clouds (3DPC) are one of the most used representation for several navigation tasks, including those of motion planning and obstacle avoidance. 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 task demand a fast reacting navigation, then higher frame rates would be necessary. In this context, 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 section 3 in more detail).

In order to exploit the cloud capabilities, the implemented solution 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. In addition to this, in the case of large amounts of data, the ratio between local-to-cloud transfer time and computation time in a single node must be taken into account as well, as it indicates the usefulness of cloud off-loading.

Despite that the use of images for motion planning can impose a bound in the stereo vision processing rate (due to image transfer overheads), it must be noticed that higher robotics levels (like object detection and recognition, gesture recognition, etc.) do require those images. Therefore these images will likely be transmitted to the cloud in any case (e.g. to build a global map or to be later

ISBN 978-3-8007-3601-0

shared with other robots), and these transfers would not be a waste of time for the whole system.
Section 2 summarizes several related works, and an overall analysis of the parallel solution is depicted in section 3 (together with a time analysis). Experimental results are shown in section 4 to quantify the benefits of the cloud approach for different scenarios, and finally conclusions are summarized in 5.

2 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 [1]–[3]. 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 [4] 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 deeply studied in [5]. 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 [6] 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 [7] 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 incoming future.

3 Architecture Overview

Fig 1 shows a block diagram of a usual vision guided robotic system. In the current work the components that are considered are those related to the stereo processing. Currently we are working on the proper accommodation of the visual information to a shared control based teleoperation of a mobile robot (according to [8]). Visual processing is carried out as following. An on-board stereo camera is the source of information of the environment for the robot. 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 (see section 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. Taking this into account, the heavy task of extracting and filtering a point cloud (3DPC) from the camera must be processed in a powerful computing system (as mentioned in section 1).
The camera sends frame pairs to a front-end virtual node (located in a private cloud). In this work, standard Internet protocols are being used. Despite their inherent non-deterministic nature, they have the advantages of the easiness of implementation, deployment and library availability. In fact, the robotics software framework currently used here works over standard TCP. These pairs are buffered and delivered to the cloud system. The cloud returns the 3DPC of the scene again to the Internet, which can be used by the robot to execute, for instance, a

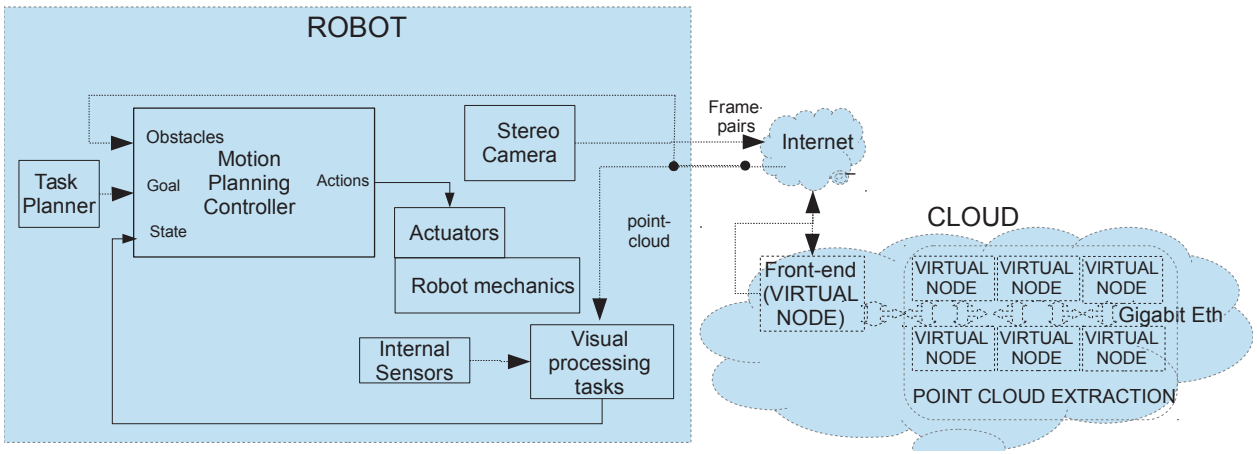


Fig 1: Overview of the robotic system

ISBN 978-3-8007-3601-0

navigation algorithm or SLAM algorithm (together with the internal sensing).

3.1 Software Implementation

In order to convert the image stream sent by the robot to a set of point clouds, we make use of the Point Cloud Library (PCL, http://www.pointclouds.org) and 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*.

More precisely, this ROS package offers a node which takes a pair of synchronized stereo frames and, after rectifying the images, produces a point cloud with all the 3D information. In order to do so, ROS implements an inner pipeline (using ROS nodelets) with several stages. Firstly, a monochrome version of the image is produced (debayer stage). 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). Thanks to this pipeline, certain degree of parallelism can be exploited if multiple cores were available. 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).

In order to decrement the minimum time below that of the longer step, and with the purpose of exploiting cloud computing properties, we have developed a more scalable solution for our cloud-based system. Apart from taking advantage of inner 3DPC extraction pipeline, a parallel solution for different frame pairs has been implemented, and thus more virtual instances can be dynamically launched if more computing power is required (for example when higher frequency rates are needed). With this important consideration in mind, the parallel solution developed is shown in Fig 2: *stereo_image_proc* processes are replicated in several virtual instances in the cloud. Each stereo frame pair will be sent to a different virtual machine in a round-robin fashion. The intermediate front-end buffering node will be responsible of determining how many *stereo_image_proc* nodes are alive at any moment, as well as scattering the stereo stream. In other words, apart from the inner pipeline that *stereo_image_proc* offers, our system adds an outer pipeline that increases the performance of the whole system. This is especially evident when bigger frame resolutions are used, and therefore the aforementioned inner pipeline is not enough to handle all the processing.

3.2 Time Analysis and Temporal correction

Processes are running in different machines that are communicated via standard network protocols, due that the system is running over the Robotic Software Framework ROS [9]. Fig 3 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. The on-board computer comprises the stereo camera *O* and the other robotics tasks *R* (as motor actuation subsystem, motion planner, along with 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 inverted . 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 section 3.1) 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 3) are elevated (more than the period of *O*), , some frames are discarded (shown like clear rectangles in Fig 3) 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 3), in order to assure that the mean time to process a pair of frames is less than the transfer time (t_o in Fig 3). 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).

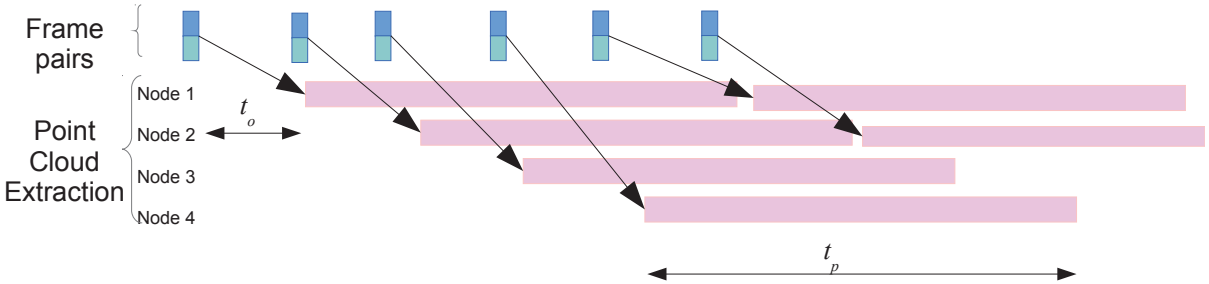


Fig 2: Stereo frame pipeline process. Four nodes process (in a pipeline fashion) the frame pairs that the front-end node delivers in a round-robin form.

ISBN 978-3-8007-3601-0

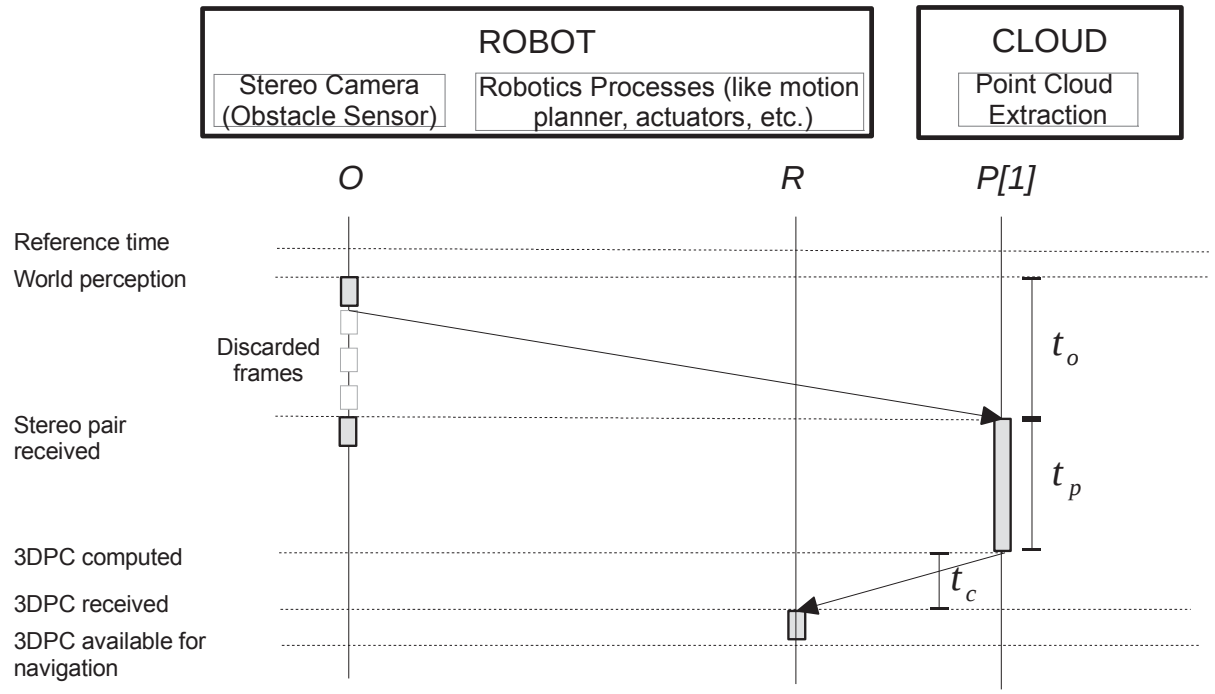


Fig 3: Time diagram of the point cloud extraction for one virtual node. More virtual nodes suppose that more processes P will be running in parallel with different frames, so a little number of frames would be discarded.

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 seconds is always achieved, which is enough for our purposes as total computing latencies (see section 4) are always above 0.2 seconds. Of course a better synchronization will be reached by using TDMA methods or by the incorporation of an external sync device to each platform.

4 Experimental Results

The aim of the experiment is to do an intensive performance test using different stereo streams, cloud states and connection technologies (between the robot and the cloud). The first cloud platform tested is a private small cluster of 5 physical nodes (1 front-end node and 4 computing nodes). Each node has a *AMD Phenom 965 x4* 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).

4.1 Scalability Measures.

The first of the tests is a demonstration that the scalability of the cloud is working properly. Thus we use high resolution images (1920x1080 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.

	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

Table 1: Total times to process and receive n point clouds using p virtual computing nodes. Resolution of the stereo pairs is 1920x1080

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 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 possible for the robot to change between different computing resources depending on the frequency required by the robot.

ISBN 978-3-8007-3601-0

4.2 Communication technology performance measures.

Once the scalability of the cloud computing solution has been demonstrated, a second experiment is done to compare how the different communication technologies affect to the achievable processing rates. 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 second one is quite recent and promises an expected bandwidth close to that of *Gigabit*. In this experiment, 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 computing nodes.

Table 2 shows the frequency of the system for different technologies and different frame resolutions. Two facts can be deduced from these results. First of all, for the case of extracting 3DPC for small resolution frames, a boost of performance when using an external platform is not encountered, with the communication technology used here. This is due to two factors: the Erratic hardware is fast enough (for other simpler robots, cloud offloading of this process may be beneficial), and the networks we have used do not have enough bandwidth (for Infiniband or 10 Gbps Ethernet results might be better).

Despite this, when the quality of the frames is increased, the robot hardware limitations arise. Therefore the Cloud can be used not only for a simple computation offloading, but also for speeding it up.

It must be pointed out that, when changing the hardware platform (from the laptop to the robot), the frequencies obtained for *Gigabit Ethernet* differ from the ones obtained in subsection 4.2. In order to understand this fact, we cannot forget that both the CPU are the RAM are 4 times better in the case of the laptop. Even though the robot has been freed from heavy computations, there are other factors that do affect in the whole performance of the system (peer to peer connection management, frame buffering and sending, 3DPC reception, along with others).

Unfortunately, the second fact discovered is that current wireless technologies are not enough to handle this process successfully due to the big amount of data transferred (not only the frames are transferred but also the 3DPCs). This shows that distribution of processes between the robot and the cloud must be carefully evaluated and data transfer must always be kept in mind. An example of a real application will serve to clarify this situation. A common navigation scenario for a mobile robot is the following: suppose that the robot uses the point cloud for executing a navigation algorithm, whose output is a velocity command (composed of a few dozens of bytes). Moving the navigation process to the cloud (and therefore avoiding transferring the 3DPC back to the robot) would imply that other internal robot sensor information (like wheel encoder readings) and/or processes may be computed in the cloud as well. Nevertheless, the whole system would likely work at higher frequencies than those obtained by receiving and processing the 3DPC on board, because the amount of data transfers (velocity commands instead of big 3DPC), would be reduced considerably.

Average frequency of 3DPC reception (Hz)

	320x240	640x480	1024x768	1920x1080
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

Table 2: Performance measures for different communication technologies.

To sum up, a good robotics software middleware allowing dynamic and easy process migration is shown to be a very valuable tool.

4.3 Time delay measures.

The other important data for real-time navigation is the average latency that a system lasts in processing the visual information. Taking into account the timing explained in section 3.2, in this third experiment the average latencies to receive the 3DPC of each individual frame is obtained. Each latency is defined here as the time passed since the source stereo frame was actually obtained to the 3DPC reception.

It seems logical that very delayed information is useless for most applications, especially those with near real-time requirements. For a real time navigation task, using obsolete obstacle information in a close loop control will produce oscillations in the robot navigation. Even if the latencies overpass certain bound (one second is usually an upper limit) navigation may exhibit an erratic behavior if the number of obstacles is large, or collisions may be almost unavoidable. As a consequence, the processed information is always expected to be as much recent as possible. With this fact in mind, table 3 shows the latencies obtained (using 1024×768 resolution frames) for different communication technologies. The last row shows same times for Erratic robot computing all the process on its own (no network is used). These times can serve again as a reference to show the viability of cloud computing.

In order to calculate the delay, the average times taken to perform each of the stages explained in section 3.2 (t_o, t_p and t_c) are measured using time stamps in 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 1 clarifies this statement). The results show again that we can increase the quality of the whole robotic system using the proper communication technologies.

A common problem in real-time distributed systems is the satisfaction of time deadlines. A high variability of to-

ISBN 978-3-8007-3601-0

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

Table 3: Average delay measures for Gigabit Ethernetin the case of 1024x768 resolution frames.

tal latency times occurs in our experiments. This is due to four main factors: 1) The ROS middleware used, 2) complexity of 3DPC filters and 3) TCP protocols 4) interferences, and packet rejection with back-off periods in MAC layer. It is well known that this may introduce oscillations in a control loop. Because of this we are currently working on predictive timing correction of actuation signals to mitigate this problem. Moreover we are sure that further improvements in *802.11ac* MAC layer like TDMA protocols, would reduce substantially this latency variance. An alternative to TDMA protocols is the use of the Contention Free Period in the infrastructure mode with fixed size packets. This could not reduce the variability as much as the use of TDMA, but it guarantees a minimum bandwidth reservation, which may be suitable for many real time systems.

5 Conclusions

This work shows that the cloud-based offloading of heavy visual processing tasks (like those used in motion planners) is possible. Two main evidences allow this: *a)* cloud scalability is pretty well achieved, that is, more hardware resources can be dynamically added if they were required by the visual tasks. On the contrary, the embedded hardware of a mobile robot is difficult to upgrade, whereas its energy saving requisites justify to have less performance than those of the cloud nodes. *b)* processing times run in parallel to the transmission ones, being the latter the bottleneck of the cloud offloading. In fact mean processing frequencies are almost proportional to bandwidth network. We expect that the extension and natural evolution of wireless networks would improve their bandwidth in the next few years.

On the other side, the most important problem is the high and variable latencies in our cloud solution, mainly due to: the non-real time middleware and the inherent non-deterministic of the TCP protocol that we are currently using. Future work should mitigate this drawbacks by using some kind of predictive correction terms in the loop controller and more deterministic middleware and networks.

6 Acknowledgements

This work has been supported by the Spanish grant (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02/01) and by Andalusian Regional Excellence Research Project grant (with support from the European Regional Development Fund) MINERVA (P12-TIC-1300). We wish to thank also Prof. D. Cascado for his interesting comments.

7 References

[1] E. Guizzo, “Robots with their heads in the clouds,” *IEEE Spectrum*, vol. 48, no. 3, pp. 16 –18, Mar. 2011.

[2] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, “DAvinCi: A cloud computing framework for service robots,” in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 3084 –3089.

[3] RoboEarth Group, “What is RoboEarth ?” [Online]. Available: http://roboearth.org/index.html%3Fp=1272.html. [Accessed: 05-Feb-2014].

[4] H. Bistry and J. Zhang, “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)*, 2010, pp. 3195–3200.

[5] Y. Nimmagadda, K. Kumar, Y.-H. Lu, and C. S. G. Lee, “Real-time moving object recognition and tracking using computation offloading,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 2449–2455.

[6] H. Wu, L. Lou, C.-C. Chen, S. Hirche, and K. Kuhnlenz, “Cloud-Based Networked Visual Servo Control,” *IEEE Transactions on Industrial Electronics*, vol. 60, no. 2, pp. 554–566, Feb. 2013.

[7] L. Agostinho, L. Olivi, G. Feliciano, F. Paolieri, D. Rodrigues, E. Cardozo, and E. Guimaraes, “A Cloud Computing Environment for Supporting Networked Robotics Applications,” in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*, 2011, pp. 1110 –1116.

[8] P. Iñigo-Blasco, F. Diaz-del-Rio, S. Vicente-Diaz, and D. Cagigas-Muñiz, “The Shared Control Dynamic Window Approach for Non-Holonomic Semi-Autonomous Robots,” presented at the 45th International Symposium on Robotics (ISR 2014) (Publication accepted), Munich, 2014.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, “ROS: an open-source Robot Operating System,” 2009. .

ISBN 978-3-8007-3601-0

Appendix B

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

Authors

- Javier J. Salmerón García
- Pablo Íñigo Blasco
- Fernando Díaz del Río
- Daniel Cagigas Muñiz

Publication

Title: Robots and Sensor Clouds.

Type: Book Chapter.

Editors: Anis Koubaa, Elhadi Shakshuki.

Series: Studies in Systems, Decision and Control.

Series Number: 36.

Publisher: Springer International Publishing.

Date: 2016.

Pages: 33-52.

ISBN: 978-3-319-22167-0 (hardcover), 978-3-319-22168-7 (e-book).

Studies in Systems, Decision and Control 36

Anis Koubaa
Elhadi Shakshuki *Editors*

Robots and Sensor Clouds

Springer

Studies in Systems, Decision and Control

Volume 36

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

Editors
Anis Koubaa
Prince Sultan University
Riyadh
Saudi Arabia

and

ISEP/CISTER Research Unit
Porto
Portugal

Elhadi Shakshuki
Jodrey School of Computer Science
Acadia University
Wolfville, NS
Canada

ISSN 2198-4182 ISSN 2198-4190 (electronic)
Studies in Systems, Decision and Control
ISBN 978-3-319-22167-0 ISBN 978-3-319-22168-7 (eBook)
DOI 10.1007/978-3-319-22168-7

Library of Congress Control Number: 2015947113

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Contents

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

A. Koubaa and E. Shakshuki (eds.), *Robots and Sensor Clouds*,
Studies in Systems, Decision and Control 36,
DOI 10.1007/978-3-319-22168-7_2

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

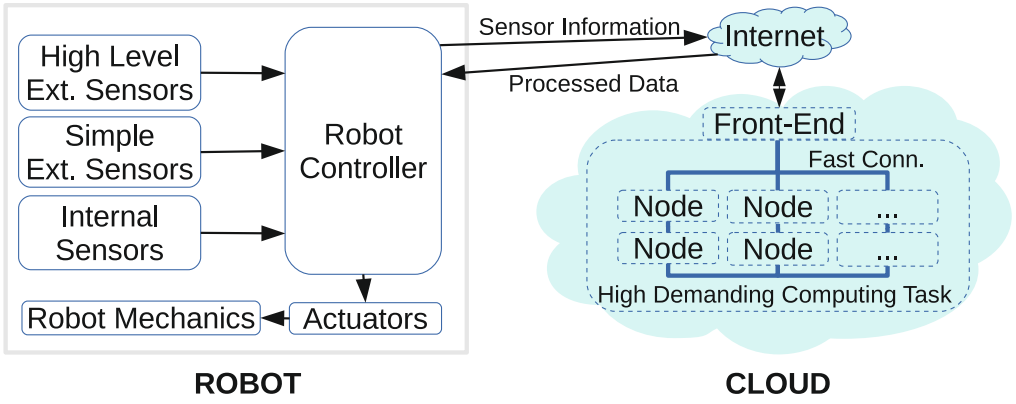


Fig. 1 Block Diagram of a Cloud-based computation offloading system

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 $IPS = f/CPI$ [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 architecture (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).

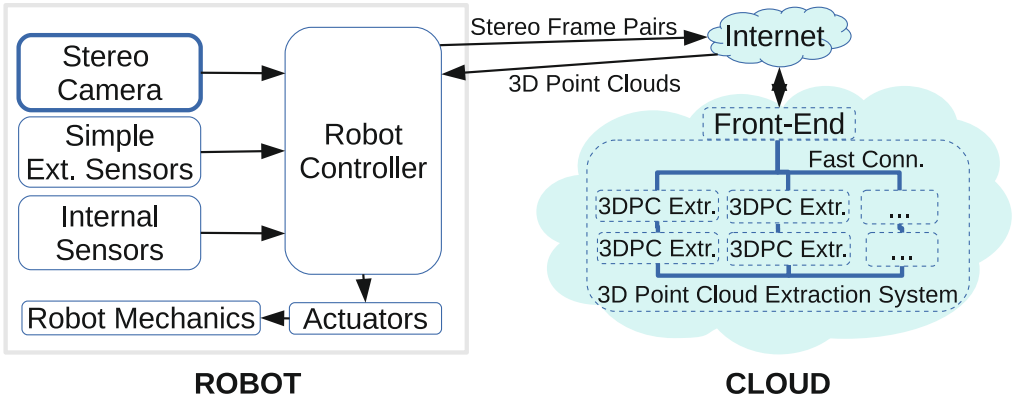


Fig. 2 Block diagram of the 3DPC extraction platform

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

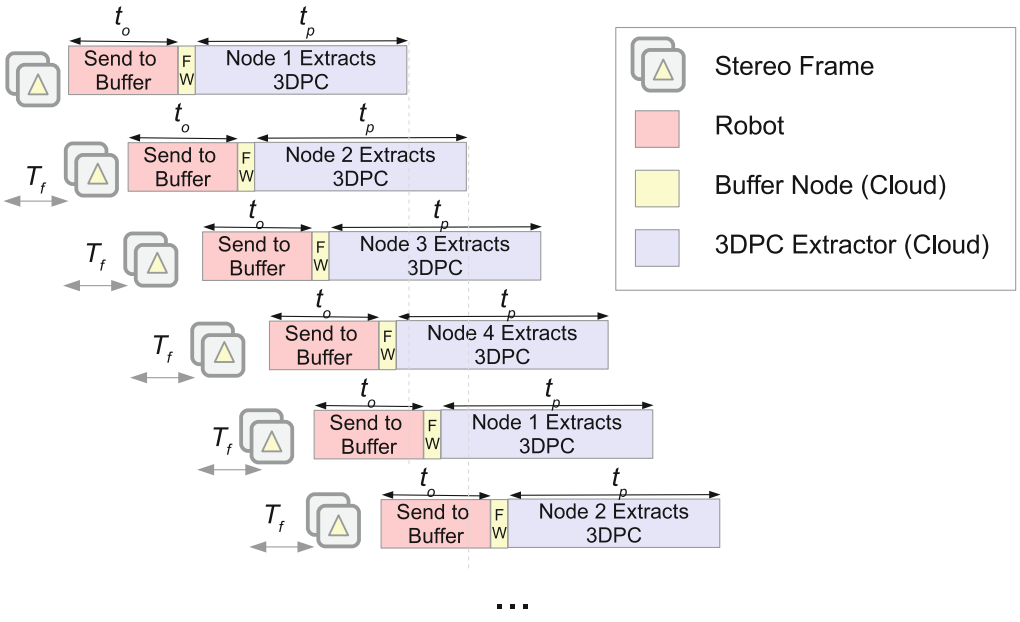


Fig. 3 Stereo frame pipeline process. Four nodes process (in a pipeline fashion) the frame pairs that the front-end node delivers in a round-robin form. T_f is the robot's stereo camera frequency, t_o is the time required to send the frame and t_p is the time required to obtain the 3DPC (see Sect. 5.2 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

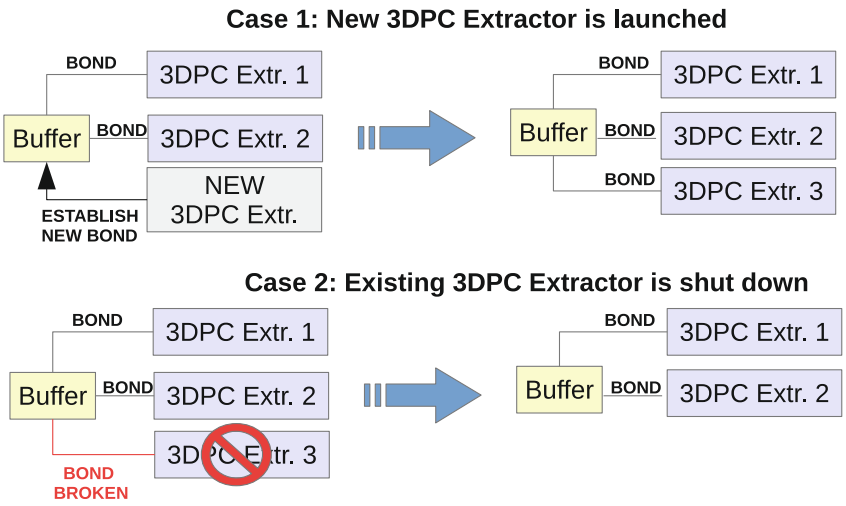


Fig. 4 Dynamic adaptation of the platform when the number of 3DPC extractors changes

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/.

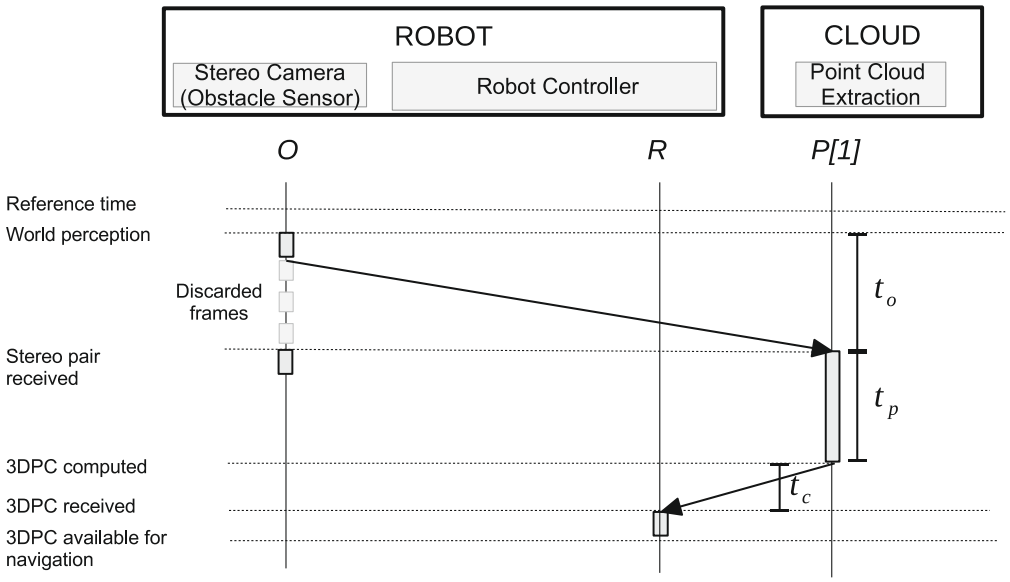


Fig. 5 Time diagram of the point cloud extraction for one virtual node. More virtual nodes suppose that more processes P will be running in parallel with different frames, so a little number of frames would be discarded

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 $\times$ 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 $\times$ 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 Hz, transfer times lower than $1/5\,\text{Hz} = 0.2\,\text{s}$ 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

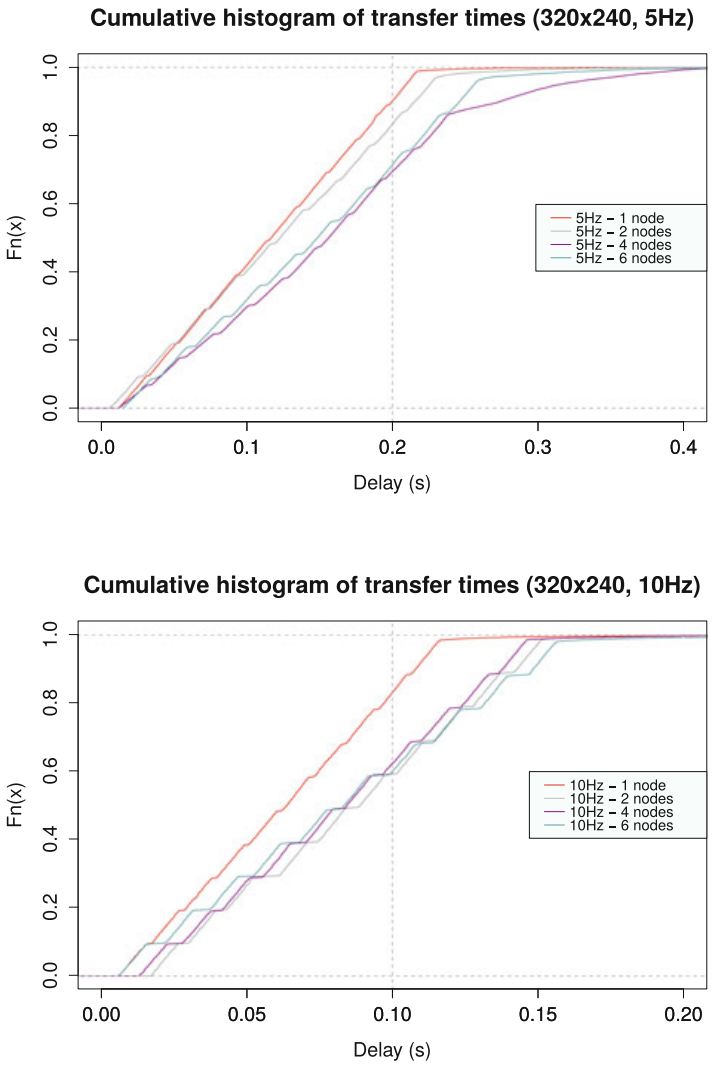


Fig. 6 Empirical cumulative distribution frequency of delays with 5 and 10 Hz

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

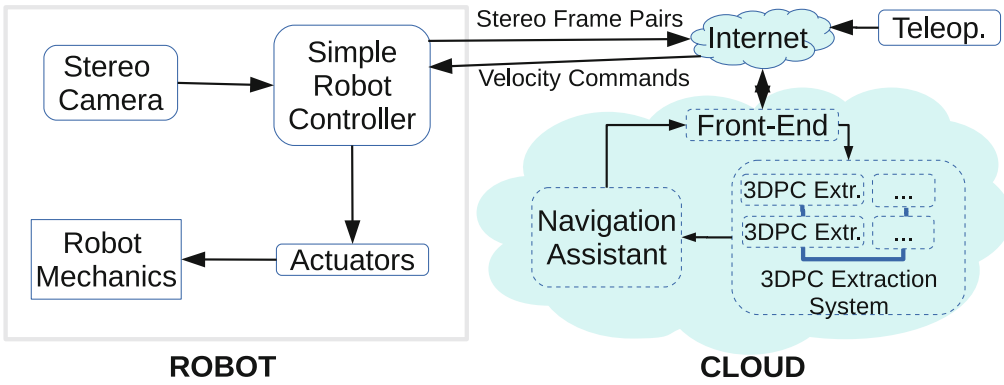


Fig. 8 Overview of the platform applied to the navigation use case

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).

Acknowledgments The work shown in this chapter has been supported by the Spanish grant (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02/01) and by Andalusian Regional Excellence Research Project grant (with support from the European Regional Development Fund) MINERVA (P12-TIC-1300). We wish to thank also Prof. D. Cascado for his interesting comments.

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

Appendix C

A Tradeoff Analysis of a Cloud-Based Robot Navigation Assistant Using Stereo Image Processing

Authors

- Javier J. Salmerón García
- Pablo Íñigo Blasco
- Fernando Díaz del Río
- Daniel Cagigas Muñiz

Publication

Title: IEEE Transactions on Automation Science and Engineering.

Type: Journal Article.

Issue: 99

Publisher: IEEE.

Date: 2015.

Pages: 1-11.

ISSN: 1545-5955.

IEEE TRANSACTIONS ON

AUTOMATION SCIENCE AND ENGINEERING

A PUBLICATION OF THE IEEE ROBOTICS AND AUTOMATION SOCIETY

APRIL 2015 VOLUME 12 NUMBER 2 ITASC7 (ISSN 1545-5955)

SPECIAL SECTION ON CLOUD ROBOTICS AND AUTOMATION

(Contents Continued on Page 393)

IEEE

(Contents Continued from Front Cover)

REGULAR ISSUE PAPERS

REGULAR ISSUE SHORT PAPERS

A Tradeoff Analysis of a Cloud-Based Robot Navigation Assistant Using Stereo Image Processing

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

***Abstract*—The use of Cloud Computing for computation offloading in the robotics area has become a field of interest today. The aim of this work is to demonstrate the viability of cloud offloading in a low level and intensive computing task: a vision-based navigation assistance of a service mobile robot. In order to do so, a prototype, running over a ROS-based mobile robot (Erratic by Videre Design LLC) is presented. The information extracted from on-board stereo cameras will be used by a private cloud platform consisting of five bare-metal nodes with AMD Phenom 965 $\times$ 4 CPU, with the cloud middleware Openstack Havana. The actual task is the shared control of the robot teleoperation, that is, the smooth filtering of the teleoperated commands with the detected obstacles to prevent collisions. All the possible offloading models for this case are presented and analyzed. Several performance results using different communication technologies and offloading models are explained as well. In addition to this, a real navigation case in a domestic circuit was done. The tests demonstrate that offloading computation to the Cloud improves the performance and navigation results with respect to the case where all processing is done by the robot.**

***Note to Practitioners*—Cloud computing for robotics is very promising for several reasons, like robot's energy saving, larger storage capacity, stable electric power, better resource utilization and the difficulty of upgrading the robots' embedded hardware. The presented application extracts 3D point clouds from the stereo image pairs of a camera situated on the robot. Using these 3D points, a shared control is implemented to help the remote teleoperation of a robot. That is, the commands sent by a joystick are attenuated when a possible collision is detected (by checking the future commanded trajectory against the 3D points). All of these computationally heavy tasks (difficult to perform by a mobile robot) are done in the cloud. The offloading models proposed in this paper are generic enough to be used in other applications. Obtained results show that further improvement in communication technologies will suppose a significant performance boost for offloading computation.**

***Index Terms*—Cloud offloading, cloud robotics, image point cloud, mobile robots, navigation assistance, shared control.**

Manuscript received November 12, 2014; accepted January 21, 2015. Date of publication March 02, 2015; date of current version April 03, 2015. This paper was recommended for publication by Associate Editor J. Civera and Editor S. Sarma upon evaluation of the reviewers' comments. This work was supported by the European Regional Development Fund under Spanish Grant BIOSENSE TEC2012-37868-C04-02/01.

J. Salmerón-Garca, P. Iñigo-Blasco, and F. Díaz-del-Río are with the Robotics and Computer Technology Laboratory, University of Seville, Seville 41012, Spain (e-mail: jsalmeron2@us.es; pabloinigo@us.es; fdiaz@us.es).

D. Cagigas-Muñoz is with the Department of Computer Architecture and Technology, University of Seville, Seville 41012, Spain (e-mail: dcagigas@us.es).

Digital Object Identifier 10.1109/TASE.2015.2403593

I. Introduction

CLOUD computing is an emerging technology for robotics, especially for mobile service robots. The needs come from the huge amount of information that a service robot has to process in order to interact and interpret the environment correctly. For this reason, during the last few years, an important number of research papers and projects are addressing the use of cloud infrastructures in robotics [1]–[4]. Research fields where clouds are of interest are those where computation is very intensive. But, where and how can high computing tasks be identified in mobile robots? According to [5], a robot usually has a layered architecture. Layers in the top level of the hierarchy can contain processes that, for example, perform cognitive tasks similar to humans. In the middle layers, tasks also involve complex processes like path planning, object handling, speech recognition, etc. Finally, in the lowest levels reactive and real-time control operations are performed (e.g., obstacle avoidance, guidance, beacon detection, signal communications processing, etc). The amount of computation is not necessarily proportional to the level. For instance, in [6], it is pointed out that an intelligent mobile robot in an office-like environment can be modeled by the Soar cognitive architecture with only a few milliseconds of computational cost. However, other middle and low-level tasks that feed higher layers in a robot architecture, can be very computing demanding. The main example is that of artificial vision and higher level tasks arising from it, such as object detection, recognition and tracking, surveillance, gesture recognition, etc. Another very promising field is that related to multi-robot cooperation at different levels, where new cooperative algorithms are being developed like multi-robot simultaneous localization and mapping (SLAM) [7], map merging (acquired by several robots), networked information repository for robots [8], amongst others.

The main properties that make cloud infrastructures compelling are the following: high reliability, larger storage capacity, stable electric power, reutilization of hardware, dynamic scalability and better resource utilization. In particular, the dynamic scalability property (adaptation of the computing power at runtime) is extremely useful in Robotics, because it allows the almost instant incorporation of new computation demanding algorithms as soon as they are implemented. Besides, the term "Cloud Robotics" has emerged to include this area, which promises fast development of complex distributed robotics tasks in the forthcoming years.

Consequently, the use of cloud computing is expected to be widespread in the next few years in the service and rehabilitation robotics field. Its use can offer them additional advantages,

such as: supplementing local information collected by the robot with that coming from Intelligent Environments, Ambient Assisted Living applications like monitoring user's health and daily activities, multirobot cooperation and so on. In this respect, it must be noticed that images captured by on-board cameras are to be required by higher robotics levels (like object detection and recognition, gesture recognition, etc., which must run in the cloud), or by a human teleoperator that must make a decision over the robotics system or simply teleoperate the robot itself. Therefore, these images will be transmitted to the cloud in any case, and these transfers would not be a burden for the whole system.

Navigation assistance and human-computer shared control, are common application cases for intelligent wheelchairs. Many wheelchairs incorporate high-ended sensors, like stereo cameras, laser rangers, and Ambient Intelligence aids [9] to address this question. However, despite great advancements in power wheelchair technology, research shows that wheelchair related accidents occur frequently, especially for users with considerable handicaps [10]. Daily wheelchair maneuvers could be challenging due to the users' pathologies, poor maneuvering skills, user's fatigue, and unknown or adverse environments. Therefore, a safe maneuverability of a wheelchair using on-board or external high-ended sensors has great importance in this kind of situations. Moreover, remote teleoperation of the wheelchair by an external caregiver could be sometimes necessary.

In addition to this, the results obtained in our Lab research in advanced wheelchairs control [11], [12] support the necessity of navigation assistance in certain situations. Because of the probable omnipresence of cloud infrastructures for service robots, in this work a prototype is implemented as a first step to analyze and demonstrate the viability of carrying out in a cloud the lower (but intensive computing) levels of a mobile robot navigation assistant. The presented prototype is running over a Robotic Operating System (ROS)-based mobile robot (Erratic by Videre Design LLC) due to its software deploying and debugging ease. However, this idea will be applied to other devices like intelligent wheelchairs or other mobile service robots, extending our previous research. More precisely, the task consists of the teleoperation by a local or remote (which has little influence because the inherent distributed architecture of the developed system) user, who is helped by using the sensing information extracted from an on-board (but processed in a cloud infrastructure) stereo camera. In order to exploit the cloud capabilities, a dynamic parallel algorithm has been implemented, so the solution is able to scale out and back. Hence, the robot gets the results faster when more computation power is required.

The Computation offloading of a robotics task includes several tradeoffs. The software architecture (and its components) of a complex robotic system must cater for a variety of characteristics, distinguish it from other systems. The most relevant are [13]: concurrent and distributed architecture, modularity (several components of high cohesion but low coupling), robustness and fault tolerance, and real-time efficiency. The first two characteristics primarily benefit from cloud offloading, while the third introduces new challenges (network robustness and fault tolerance appear as a new aspect to considerate). Nevertheless, the main issue that offloading must address for a navigation task is the fourth of them. In this respect, a short quantitative comparison of times involved in local versus remote computing follows. This demonstrates the theoretical real-time viability of cloud offloading and it points out new considerations.

The first mandatory bound is the time spent in data transmission to the cloud. If the bandwidth rate of communication technology is BW, and data size to be transmitted and received is D, it is concluded that D/BW must be inferior to the deadline of the task. Nowadays, there are wireless networks whose bandwidths [14] are approaching to 1 Gbps, with a steady incremental ratio of more than 40% per year, or two times every two years (see http://www.wi-fi.org/). For a typical navigation control loop, frequencies around 20 Hz (period $T = 0.05$ s) are usually enough. This gives us a limit of 0.05 Gbits, or equivalently, a 3 Mpixel raw B/W stereo image, per period. Therefore, offloading computing is feasible, which, indeed, has the aforementioned benefits.

A secondary aim is that cloud accelerates timing execution. Let us suppose that the robot computer can execute at a rate of IPS instructions per second (millions of IPS or $MIPS$ is a common magnitude in computer architecture [15]), and that the cloud can speedup an application S times more that this IPS. Therefore, local and remote execution times for N_I computer instructions could be expressed as

$$t_{\text{local}} = \frac{N_I}{IPS}; \quad t_{\text{remote}} = \frac{N_I}{IPS \cdot S} + \frac{D}{BW}.$$

For stereo vision applications, S is very big because the cloud is supposed to have far more computational resources than the local computer, and many operations are performed in parallel. Being that the case, timing comparison gets to a short formula, which indicates whether computation offloading is faster than local execution, that is, $t_{\text{local}} > t_{\text{remote}}$, if

$$\frac{N_I}{D} > \frac{IPS}{BW}.$$

The first term of this inequality is entirely dependent on each application. This means that high intensive computing tasks benefit from cloud computing. With respect to image processing, many of the filters that extract features used in point cloud processing have computational complexity orders higher than $O(N)$, being N the number of pixels. For example, in our experiments, a frame pair is computed by the Erratic in $t_{\text{exec}} = 0.96$ s (see Section V). As the Erratic CPU (Intel Celeron-M) runs at a frequency of $f = 1.4$ GHz and has a Clocks per Instruction (CPI) around 2.0 [16], then, the number of instructions is [15]

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

Besides, transmitted data in this experiment (see Section V) mainly consists of 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

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

On the contrary, the second member of the inequality is mainly technology dependent. Nowadays, IPS is f/CPI [15], so for the Erratic CPU, it rounds $IPS = 7.0 \cdot 10^8$ instr/s. Therefore, networking bandwidth is crucial to have success in the offloading. Using WiFi IEEE 802.11ac, 400 Mbps data rate

transmissions can be easily achieved and, hence, the second term has a very much lower value (1.75 for actual case) that the first one, which promises a successful offloading. On the whole, it can be concluded that many tasks from the top, middle, and even lowest, level of a common layered robot architecture are presently candidates to be remotely executed, or they would be it in a few years.

The rest of this paper is organized as follows. Section II summarizes several related works. Next, two sections analyze the practical case study: Section III explains the architecture of the system, especially the navigation assistant and a timing analysis. This timing analysis is vital for studying the computation offloading possibilities of the presented solution, which will be explained in Section IV. Experimental results are shown in Section V to quantify the benefits of the cloud approach, and finally conclusions are summarized in Section VI.

II. Related Work

Cloud robotics has two major lines of work. The first one corresponds to the creation of an "Internet for robots" [8], where all robots extend their knowledge using a predefined language and collaborative build and merge/retrieve information [17]–[20]. The second line of work (though not being completely separated of the former) is the one studied in this work: computation offloading. This paradigm is being used by many works and projects that accomplish high-level vision tasks, though they do not have real-time requirements [2], [3], [21]. However, some papers have appeared in the last few years that propose the external processing of several parts of the sensor feedback information, achieving close to real-time performance for these operations. In [22], they combine GPUs and cloud offloading (to a private cloud infrastructure) to perform SLAM, with successful results in different environments. They study different virtual resources configurations as well: 1 virtual machine per bare metal node, several VM's per node, amongst others, finding that virtualization overheads imply a degradation in speedup (they use Xen as virtualization technology). In this sense, the method for identifying the optimal balance between a cloud system overhead and performance presented in [7] can be useful. Executing SLAM in the cloud is also studied in [23], 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). Another computation offloading example is proposed in [24], where 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. The configuration of these external servers is specific to this work, so some properties of the cloud computing paradigm are not exploited.

The idea of Computation Offloading is studied in [25], and an estimation of the computation and communication times needed for the recognition and object tracking tasks is presented 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.

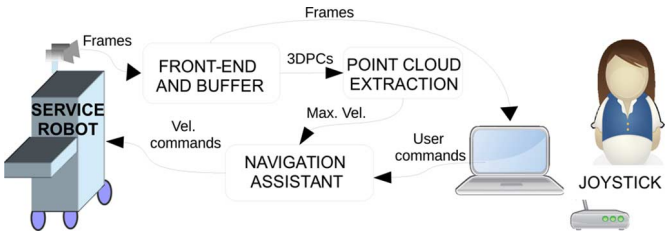


Fig. 1. Block diagram of the Cloud-Based Navigation Assistant for the teleoperation of a mobile robot. The different types of data transfer are the following: Stereo frame pairs, 3DPC, robot velocity commands, and user input commands. Also, bond status messages are required by ROS bond library.

Authors of [26] present an object-tracking scenario for a 14-DOF industrial dual-arm robot using a UDP transport protocol for transmitting large-volume image over an Ethernet network. Thanks to the very low sending and cloud image processing times that are achieved, a stabilizing control law can be implemented, with time-varying feedback time delay.

The work [27] also asks whether the performance of distributed offloading tasks can be compared with those executed on-board. The experiment performed consists of a simple controller that guides the robot in order to follow a line according to the images acquired from a single low-resolution (320×240) camera that points to the floor.

Compared with the described literature, the work presented in our paper is the first that tries to analyze the cloud offloading of such a real-time task as navigation assistance. In addition to this, a thorough explanation of all the offloading possibilities, together with the role of communication technologies, adds more novelty to our work.

III. Overview of the System

The analyzed robotic application consists of a teleoperated mobile robot using a shared control via on-board cloud-based stereo vision (continuing the work presented in [28]) and the robotic software framework ROS (http://www.ros.org). The experience of our Lab in shared control for wheelchairs [29] has inspired this present proposal, whose aim is to enable inexperienced or handicapped pilots to safely drive vehicles in challenging scenarios.

Section III-A summarizes the different modules of the application and Section B explains the stereo vision processing implementation. Section III-C explains how the navigation assistance system works and Section III-D addresses the timing analysis of the computation offloading.

A. Block Diagram of the System

Fig. 1 shows a simple diagram of a vision navigation assisted robot. The robot carries a stereo camera which sends frames to the teleoperator and to those processes responsible for 3D Point Cloud (3DPC) extraction. Once a 3DPC is obtained from a stereo frame pair, it is sent to a navigation assistant node. This navigation assistant node will also receive input commands from the local command interface (a joystick in this case), which are translated to the desired linear and angular speeds (v_j, ω_j) With the previously received obstacle information (that is, the

3DPC), and the historic buffering of robot velocities, the navigation assistant will be able to calculate the correct $v_{\max}$ that avoids an obstacle collision. Once this $v_{\max}$ has been computed, a velocity command (v_c, ω_c) which satisfies the same curvature suggested by the user through the joystick, is sent to the robot. This fusion of information is usually called "continuous shared control" [29], which is frequently preferable for most navigation assistance systems, because the desired commands were smoothly and continuously combined with a collision avoidance criteria [30].

The distributed implementation and the discussion of where should each node run, whether the robot or the cloud, are more deeply discussed in Section IV.

B. Stereo Image Processing

Processing stereo images is currently a very heavy computation task. However, the use of stereo cameras as the sensing technology implies several advantages against other simpler sensors (such as infrared or ultrasonic sensors). First, the information obtained with these cameras is more complete. Second, cameras have usually wider fields of vision. In addition to this, as explained in Section I, the stereo pairs will serve for the teleoperator or will presumably be required by other high-level application. Hence, images need to be transferred out of the robot in any case. Nevertheless, there is always the possibility of combining the visual processed information with that of more basic and reactive sensors. Besides, RGB-D cameras are being developed quickly and at relatively low prices [31]. However, low priced RGB-D cameras are currently aimed for the game market, and, hence, they present several disadvantages. Regarding to our system, their drawbacks are: they do not work well when there is lot of sunlight (even indoors), their FOV is much more limited, maximum and minimum distance detection is bounded, amongst others.

For a successful navigation assistance, the frequency of stereo frame processing must be sufficiently high, whereas its latency small enough. This requirement can become very difficult to meet when more accurate reconstruction of the environment is demanded. Hence, the heavy task of 3DPC extraction must be processed in a powerful computing system. Furthermore, this processing must be designed, not only to be parallel, but also to exploit the special properties that a cloud system offers (more importantly, that of dynamic scalability).

The 3DPC extractors (Fig. 1) are implemented thanks to a ROS package called *stereo_image_proc*, which uses two large-scale, open source libraries for 2D/3D point cloud processing and computer vision libraries: the *Point Cloud Library* (PCL, http://www.pointclouds.org) and *OpenCV Library*. More precisely, this *stereo_image_proc* ROS package offers a node which takes a pair of synchronized stereo frames and, after rectifying the images, produces a point cloud with all the 3D information. ROS implements an inner pipeline (using ROS nodelets) with several stages: image debayer, image rectification, disparity map creation and point cloud building. As each stage is dependent from the previous one, no parallelism can be achieved for an unique frame pair.

In order to make the image processing dynamically scalable, a parallel solution for different frame pairs has been implemented. The idea developed is based on the ability to replicate the 3DPC extractor nodes (see Fig. 2), in several virtual instances in the cloud. Each stereo frame pair will be sent to a different node in a round-robin fashion. If t_p were the time to process a frame pair and T_f the frame acquiring period, the minimum number of nodes would be $\lceil (t_P/T_f) \rceil$ (in practice some additional nodes are added due to the variance of processing times). This scattering method requires a front-end frame buffering node ("Buffer" node in Fig. 1) that will be responsible of determining how many 3DPC extractor nodes are alive at any moment, as well as distributing the stereo stream. This parallel solution increases the performance of the whole system, which is especially evident when bigger frame resolutions are used. The main advantage of this solution is its inherent dynamic nature. If the solution needs to scale out, more 3DPC extractors can be launched at runtime. Once the buffering node detects them (thanks to ROS *bond* library, using bond status messages), they will start receiving stereo pairs for processing. Reversely, if the system must be scaled back, some nodes will be shut down and hence resources would be freed.

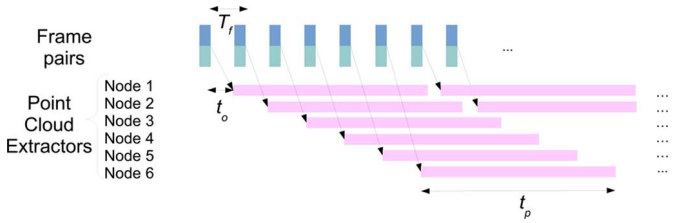


Fig. 2. Stereo frame pipeline process. 3DPC Extractor nodes process (in a pipeline fashion) the frame pairs that the front-end node delivers.

C. Overview of the Navigation Assistance

The "Navigation assistant" node receives the 3DPC, which includes the obstacle information. A shared control that continuously filters user commands by means of the obstacle information is implemented as follows. Due that frontal view of cameras only allows controlling small curvature arcs, a simplified version of [29] is used here. Nevertheless, as the user can receive visual feedback of the operated scene, he/she can command pure turns in those areas where no lateral obstacle have been seen by her/him.

The first step is the projection of all those points in the XY plane. After that, for each point (x_p, y_p) in the 3DPC, a trajectory from the driven wheel axle center O (axis origin in Fig. 3) is calculated. This process is done by finding a circumference whose radius has endpoints at O and at (x_p, y_p). This circumference supposes a feasible circular path of the robot, whose center is $(x_c, 0)$ and whose equation is

$$(x - x_c)^2 + y^2 = R.$$

As the circumference has endpoints at $(0, 0)$ and (x_p, y_p), the radius R, curvature value K and x_c can be calculated as follows:

$$x_c = \frac{x_p^2 + y_p^2}{2 \cdot x_p}, \quad R = |x_c|, \quad K = 1/R.$$

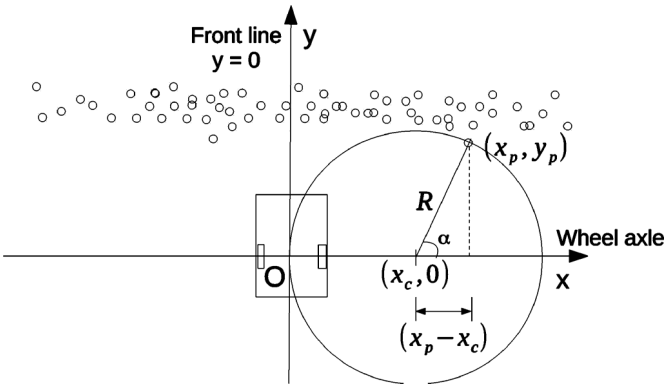


Fig. 3. Plot of the navigation assistant approach. The upper circles are points of the Point Cloud. The circumference that crosses a point and the middle driven wheel axis is calculated.

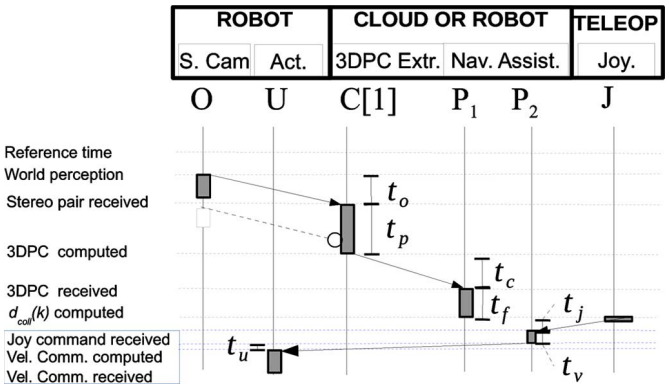


Fig. 4. Time diagram of the system. Camera O captures a frame pair that sends it to the Point Clod extraction C. This extracts the 3DPC and sends it to the Navigation assistant, which fuses this information with the user command coming from the joystick J. Finally, the actuators U receives the velocity commands. Different interval times increment the total latency of the system.

Finally, using the angle α between X axis and the radius, the arc distance from O to (x_p, y_p) is obtained (see Fig. 3)

$$s = \Pi R - \alpha R = R \cdot (\Pi - atan2(y_p, x_p - x_c))$$
$$s = y_p, \quad if\ R \to \infty.$$

For each point (x_p, y_p), a pair (K, s) is calculated, and thus it is possible to interpolate a function $s = d_{\text{coll}}(K)$ with all this information, that is, the distance to collision for each curvature. Every time a new stereo pair is received, $d_{\text{coll}}(K)$ will be updated.

Simultaneously, the user is sending joystick commands, which are proportionally translated to a desired pair of linear and angular velocities (v_j, ω_j), and a curvature $K_j = \omega_j / v_j$. This desired curvature is mapped to a distance value $s_j = d_{\text{coll}}(K_j)$.

Using the uniformly accelerated motion equations, the maximum linear velocity $v_{\max}$ that guarantees a stop without collision can be obtained: $v_{\max} = \sqrt{2 \cdot a_{\max} \cdot s_j}$, where $a_{\max}$ is the maximum braking acceleration of the robot. Every time $d_{\text{coll}}(K)$ is updated, $v_{\max}$ is reviewed and updated. If $v_{\max}$ is less than v_j, the input command cannot be allowed, and the speed sent to the robot is $v_c = v_{\max}$. Angular speed ω_c is calculated so the resulting pair sent to the robot (v_c, ω_c) retains the same curvature asked by the user. This way the teleoperator does not feel that the navigation aids change the desired trajectory. Should $d_{\text{coll}}(K)$ be updated fast enough, this event would occur naturally, and the robot brakes progressively. Otherwise, the navigation assistant will automatically send brake commands at a certain frequency.

D. Communication Time Analysis

When working with real-time navigation, an analysis over the average latency is required. Most of the latency comes from the time the system takes to process the visual information. A simplified timing diagram of system is shown in Fig. 4 (the front-end buffer node is not shown because its delay times are negligible with respect to the other involved times). The robot's on-board computer comprises the stereo camera O and the motor actuation subsystem U. Depending on the offloading model, the following nodes can be either in the robot or in the cloud (Section IV analyzes these possibilities): the point cloud extraction C and the navigation assistant (P_1 and P_2). P_1 is responsible for creating and updating the $d_{\text{coll}}(K_j)$ function explained in Section III-C and P_2 fuses the information from the joystick J with $d_{\text{coll}}(K)$.

Pairs of frames are continuously sent from camera node O to C (via the buffer) at a specified frequency. The transfer time of this message is named t_o. As explained in Section III.-A, the C nodes receive the stereo pairs in a round-robin fashion (in the figure only one node $C[1]$ is represented for simplicity). After computing time t_p, this new extracted 3DPC is sent to the navigation assistant P_1 (taking t_c). There, the function $d_{\text{coll}}(K)$is obtained and updated in a time t_f. On the other side of the figure, each time the joystick J sends a command, a new action will be computed by P_2. Time intervals t_j and t_v involved in this sending, are negligible with respect to the others. This action is sent back to the robot, where module U applies the desired speeds to the actuators. These speeds need to be recalculated until a new action arrives, as the period of the actuation subsystem is inferior to the time to transfer a new frame pair. At the moment, a simple interpolation taking into account the real robot speeds and the last sensed obstacle map is carried.

If, for instance, one new stereo pair tried to enter the *stereo_image_proc* pipeline (see Section III-C) and no nodes were free, this new pair would be discarded. In the case of only one node C, as the processing times t_p in Fig. 4) are usually longer than the period of O, many frames would be discarded (shown like clear rectangles in the figure) until the node were idle again. In order to cut down the number of discarded frames, this timing analysis gives us another bound to the minimum number p of 3DPC extractor nodes. As images are to be sent by a single physical channel (that is, wirelessly), transfer times (t_o) are not overlapped. Due to this, frame acquiring period T_f cannot be inferior than t_o. Furthermore, the mean time to process a pair of frames (t_p/p) must be less than the transfer time (t_o) to send it. In order to get rid of this issue and, as we are working with high scalability systems, p is overestimated. Hence, $t_p/p < t_o$ is always guaranteed.

Another two critical aspects arise from the proper network characteristics: 1) a strict periodical controller cannot be implemented as WiFi networks (proper candidate for mobile robots) have considerable fluctuations in transmission times and 2) the presence of delays in control loops tends to produce oscillations. Nevertheless, these oscillations are overcome by two reasons.

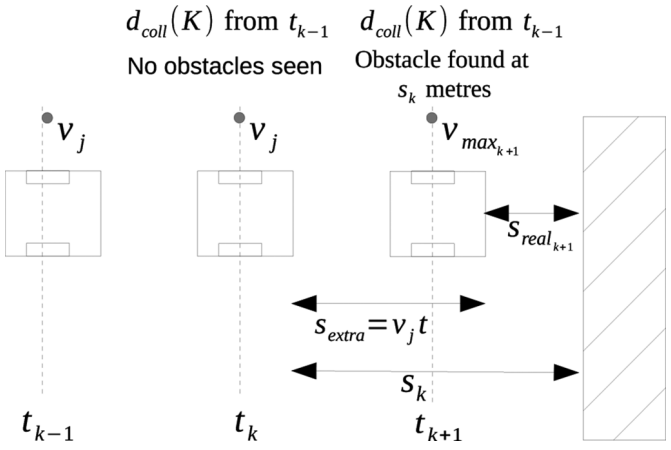


Fig. 5. Example case of delay correction in distance to collision s calculation. As the obstacle information is delayed, time predictive corrections must be applied. At t_{k+1}, the system would calculate that an obstacle is at s_k meters, but the robot is actually closer. A simple predictive correction consists of subtracting s_k an estimated distance $s_{\text{extra}} = v_j \cdot t$, where $t = t_{k+1} - t_k$, and v_j is the commanded speed.

First, the effect of these oscillations are mitigated or counteracted by the user, because he/she is conscious of this kind of problem when remotely driving a slow robot. Second, some delay corrections have been incorporated (see Fig. 5) in the actuation subsystem. As the information obtained from the stereo cameras is delayed, obstacles are actually nearer than $d_{\text{coll}}(K)$ states. Let us suppose that the system calculates at t_{k+1} that an obstacle is at s_k meters. Due to the timing latencies (see Fig. 4), the robot is actually closer. To be conservative, we have supposed that during this latency time the robot has moved at the maximum speed asked by the user, named v_j. Therefore, a simple predictive correction consists of subtracting s_k an estimated distance $s_{\text{extra}} = v_j \cdot t$, where $t = t_{k+1} - t_k$.

One final critical issue must also be considered: the case where the actuation subsystem does not receive any command for a long time. This case is addressed automatically due to the way the interpolation is implemented. The idea is that the robot speeds are gradually reduced at each actuation period to prevent a collision, according to the last obstacle reading. In this case, the robot speeds will tend to zero. There is an even more critical circumstance: when last camera processing did not find any obstacle, so P_2 sends to U an obstacle-free signal, and no other obstacle information is received anymore. Due to this, a deadline internal time is considered by U: if it is exceeded, robot speeds are progressively decremented to prevent a collision.

IV. Computation Offloading Analysis

This section analyzes and debates how the cloud can serve to offload the implied computation of the previously explained robotic application. It should be noted that the discussion presented here is generic enough to be applied to other applications. For example, a cloud offload autonomous robot has similar processing nodes to those shown in this section.

As stated in Section I, there are inherent tradeoffs when moving computation into the cloud: the communication overheads, the amount of computing resources used, along with

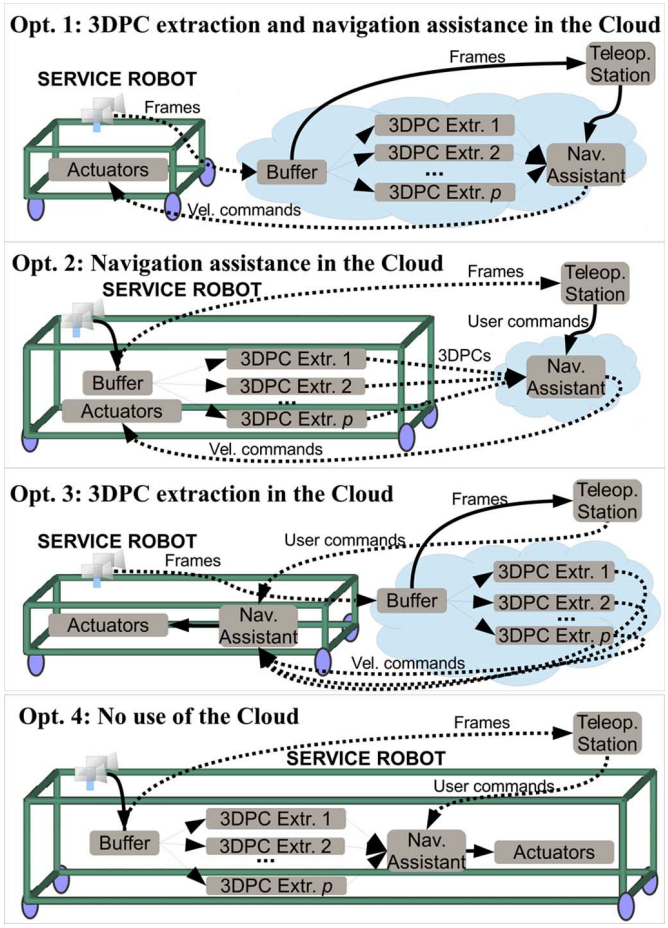


Fig. 6. Possible configurations for Cloud offloading. Mandatory wireless communications (those which origin or destination is the robot) are represented by dotted arrows while communications that can be wired are depicted by continuous arrows.

others. Different options will be presented, justifying which is the best one among them (for the current application).

For this reason, a summary table, which classifies the different options qualitatively, is presented at the end of this section. The selected parameters of this table have been chosen due to their relevance in cloud computing and robotic technologies. They are: a) Best scalability, which indicates if the involved processing can be easily scaled out or back. b) Least communication bandwidth, as bandwidth is a precious resource in cloud computing and a source of timing delays. c) Least virtual computing resources. It indicates the amount of cloud resources used by an option. d) Cheapest cloud pricing, which must be considered when using public clouds, such as Amazon EC2. In addition to this, Section V will include experimental numerical results that quantifies which option is the best for the presented application.

Fig. 6 shows the most interesting ways of distributing the main computation tasks to be carried out in the navigation assisted teleoperation. These tasks are: the Frame buffering process ("Buffer" in Fig. 6), 3DPC extractors ("3DPC Ext." replicated in p nodes in Fig. 6), and the navigation assistant. For the current system, experiments in Section V show that the 3DPC extraction is more computationally complex than the navigation assistant, which basically includes the $d_{\text{coll}}(K)$

TABLE I
QUALITATIVE CLASSIFICATION OF DIFFERENT OFFLOADING MODELS. THE HIGHER THE NUMBER, THE WORSE THE OPTION IS REGARDING THAT PROPERTY

Property	Option 1	Option 2	Option 3	Option 4
Best scalability	1	3	2	4
Least communication bandwidth	2	3	4	1
Least virtual computing resources	3	1	2	-
Cheapest cloud pricing	2	1	3	-

function processing (see Section III). The necessary messages between these tasks are: 1) Color frames from camera to buffer, whose size is approximately proportional to image resolution; 2) Color frames from the buffer to the 3DPC extractors; 3) 3DPCs sent to the navigation assistant, which size in bytes ranges from 2 to 3 times that of raw color frames; and 4) Velocity commands that will be received by the actuation subsystem, which suppose just a few bytes.

For obtaining more reliable performance measures, every node moved to the cloud is isolated in one virtual machine. As explained in Section III, the different nodes of this solution have inherent distributed nature thanks to ROS middleware. Fig. 6 shows p 3DPC extractor nodes, because this processing part was designed to be dynamically scalable (see Section III).

Last option 4 is the classical robot centralized approach, but, when cloud offloading can be considered, other options are conceivable. As moving the camera to the cloud makes no sense (it must stay in the robot), and the buffer must reside in the same platform that the 3DPC extractors to prevent a waste of frame transfers, three additional options appear (see Fig. 6).

Option 1: 3DPC extraction and navigation assistance in the Cloud. This first configuration aims to move all the existing computation to the Cloud. A first advantage is that the robot has practically all its computing power available for other robotic tasks. Moreover, this configuration allows the exploitation of the cloud properties, due to the following fact: should the robot require more computing power, more 3DPC virtual machines can be spun up at runtime, and therefore the quality of the navigation will presumably increase (as it will be able to reach higher frequency rates or to process bigger frame resolutions). In terms of communication bandwidth usage, this configuration requires the transmission of stereo frame pairs (which size in bytes increases linearly with the frame resolution) to the cloud and reception of velocity commands (which size is commonly very short).

Option 2: Navigation assistance in the Cloud. This choice tries to better balance the computing tasks between local and cloud computation, but it is less scalable. Scaling out and back in the robot is far more limited than doing it in the Cloud. In terms of bandwidth usage, instead of sending frames to the cloud, 3DPCs will be sent (whose size ranges from 2 to 3 times that of frames'), and velocity commands (a few bytes) will be received. Therefore, the wireless bandwidth usage is bigger than the previous case, not being the case of virtual computing requirements. For a fully autonomous robot, images must not come out of the robot. However, for our current teleoperated application, video streaming must also be sent to the teleoperator, thus increasing bandwidth interferences and consumption. With respect to the current application, in terms of cloud pricing, this solution is cheaper than the previous one. However, because 3DPC extraction is more computationally complex than the navigation assistance, the robot will probably be unable to respond correctly when a better quality of the navigation is required (for example a bigger frame resolution, as delays in Section V exhibit).

Option 3: 3DPC extraction in the Cloud. In this case, the heaviest processing is moved to the cloud, whereas the navigation assistant node (which could be more reactive if the robot included another simple sensor) stays in the robot. This configuration seems the sensiblest in terms of typical real-time applications. In terms of scalability, it is possible to scale out and back 3DPC nodes depending on the needs. In terms of virtual computing power requirements, it is very similar to the first choice since the main core of computation resides in the 3DPC extraction. However, the main drawback of this configuration for the current application is the big amount of wireless communication that takes place: the robot sends stereo frame pairs and receives 3DPCs. As a consequence, if the communication technologies are not good enough, this solution can yield bad performance results. In addition to this, this solution may be the most expensive in terms of pricing due to the high bandwidth consumption.

Option 4: All processing in the robot. In this case, the only communication overheads will be internal to the robot node, which nowadays are usually inferior to those of remote communication. This choice does not use the cloud, hence it requires the robot's embedded hardware to be powerful enough. For the real application considered here, the next section demonstrates its poor performance results (even for Erratic Robot, which incorporates medium-end hardware). Just like option 2, the use of remote communication would be necessary if robot camera images were sent to the teleoperator.

Table I classifies all the choices according to all the properties analyzed. Taking into account all the advantages and disadvantages, it seems that options 1 and 3 are the most suitable for the described application. Section V proves that option 1 yields the best performance results.

V. EXPERIMENTAL RESULTS

This section presents the performance results for different experiments using the aforementioned example robotic application. The main objective is to prove which cloud computing options are viable for a real-time robotic system. In [28], it was proved that the scalability of the cloud works properly when changing the number of virtual cloud nodes. The system was tested with the hardest computing and communication processing: high definition resolution images (HD 1080i frames), the most consuming bandwidth model (option 3 in Fig. 6), but with the fastest available TCP network (Gigabit). It was showed that a significant speedup (ratio between total time for

1 node and for p nodes) can be obtained. This means that cloud elasticity makes it possible for the robot to change between different computing resources depending on the frequency and image resolution required by the system. Despite the fact that in some applications low-quality images and small updating frequencies could be enough, the stereoscopic formula for the depth error [32] says that this error is proportional to the real size of a pixel (that is, for a given CCD size, the more horizontal resolution, the less depth error exists. As an extension of this idea, it is evident that with perfect lighting and essentially infinite SNR, the highest accuracy is achieved using a combination of high framerate and high resolution, with limits only set by the available computational budget [33].

Section V-A analyzes the performance impact of current communication technologies, which has direct impact in the offloading. In that sense, Section V-B shows the application qualities when choosing the different offloading options presented in Section IV. Finally, Section V-C presents a real navigation experiment, both using the cloud and the robot on its own.

The cloud infrastructure used for the experiments is the following: a cluster consisting of five bare-metal machines (one front-end and four compute machines) with AMD Phenom 965 x4 CPU (with virtual extensions enabled) and 8 GB of RAM each one. They are all connected using Gigabit Ethernet bandwidth and *Openstack Havana* is the Infrastructure-as-a-Service (IaaS) cloud middleware installed. KVM has been chosen as the virtualization technology. Other well known solutions such as Hadoop were not suitable for real-time systems. The Erratic robot hardware has a 1.4 GHz Core 2 Duo Processor with 1 GB of RAM.

When moving computation to the cloud, the following mapping between nodes and virtual machines has been considered: each 3DPC extractor node is executed in a separate virtual machine with 2 Virtual CPUs (VCPU) and 2 GB of RAM (the front-end buffer VM has exactly the same properties). In the case of the navigation assistant node, a larger VM was chosen, consisting of 4 VCPUs and 8 GB of RAM (as the parallelization of function $d_{coll}(K)$ is obvious).

A. Analysis of Communication Technologies

This first experiment is done to compare how different communication technologies affect to the achievable processing rates. Moreover, using the cloud for mobile robot navigation, more realistic technology choices such as IEEE 802.11n WiFi and IEEE 802.11ac WiFi must be considered. The second one is quite recent and promises an expected bandwidth close to that of Gigabit Ethernet. In this experiment, the stereo camera and the image transmission has been carried out by a real mobile robot (in this case Videre Erratic). The cloud offloading configuration chosen is option 3, so this experiment checks if the bandwidth limitations are being a burden in the whole performance of the system.

Table II shows the frequency of the system for different technologies and different frame resolutions. In order to give a proper comparison, the table contains the results of option 4 as well (that is, the robot without cloud). Two conclusions can be extracted from these results. First of all, when using small resolution frames (for extracting 3DPCs), a boost of performance when using an external platform is not obtained with the communication technologies used here. The reasons are the following: the robot hardware is fast enough, and the networks used do not have enough bandwidth (for Infiniband or 10 Gbps Ethernet results might be better). This means that, for other simpler robots, offloading this process might be beneficial. Despite this, when the quality of the frames is increased, the robot hardware limitations arise. Therefore, the Cloud can be used not only for a simple computation offloading, but also for speeding it up.

TABLE II
PERFORMANCE MEASURES FOR DIFFERENT COMMUNICATION TECHNOLOGIES

Average frequency of 3DPC reception (Hz)

	320x240	640x480	1024x768	1920x1080
Gigabit	16.31	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
Robot "Opt 4"	7.15	2.61	1.01	0.02

TABLE III
NAVIGATION ASSISTANT UPDATE FREQUENCIES FOR THE DIFFERENT OFFLOADING MODELS OF SECTION V USING 802.11AC WIFI

Frequency (Hz)

	320x240	640x480	1024x768	1920x1080
Option 1	12.55	5.72	3.73	1.06
Option 2	8.14	2.29	0.89	0.03
Option 3	7.48	2.16	0.96	0.25
Option 4	7.15	2.61	1.01	0.02

It must be pointed out that, when changing the hardware platform from that of the robot to a modern laptop, the frequencies obtained for Gigabit Ethernet differ from that of [28]. This fact is easily understood because both the CPU and RAM are four times better in the case of the laptop. Even though the robot has been freed from heavy computations, there are other factors that do affect in the whole performance of the system (peer to peer connection management, frame buffering and sending, 3DPC reception, along with others). The second fact discovered is that current wireless technologies are not enough to handle this process successfully due to the big amount of data transferred (not only the frames are transferred but also the 3DPCs). However, it must be noted that the offloading model chosen was not beneficial for limited communication technologies.

B. Comparison of Offloading Models

In order to compare all the offloading models of Section IV, two tests were carried out: the first one compares the frequencies in which $d_{\text{coll}}(K)$ is updated for different frame sizes, while the second one measures the latencies of the whole application.

Table III shows the results for different frame sizes and models using 802.11ac WiFi. The faster $d_{\text{coll}}(K)$ gets updated, the higher the quality of the navigation will be. Just as expected, the option 1, which has less bandwidth consumption, has the best results, even though option 3 seems a more common configuration for a real-time system. Only when the frame resolution is small, the required computation can be assumable by the robot (option 4) on its own. Due to the communication overheads, the speedup between options 1 and 4 becomes greater when the size of the frame increases, being worth having

TABLE IV
AVERAGE DELAYS OF THE SYSTEM FOR 1024 × 768 FRAMES AND DIFFERENT OFFLOADING MODELS USING 802.11AC WIFI

Mean Delays (s)						
	t_o	t_p	t_c	t_f	Total Comm.	Total
Option 1	0.199	0.382	0.154	0.026	0.353	0.761
Option 2	0.077	1.007	0.652	0.026	0.729	1.762
Option 3	0.511	0.382	1.223	0.064	1.734	2.180
Option 4	0.077	0.966	0.181	0.096	0.258	1.320

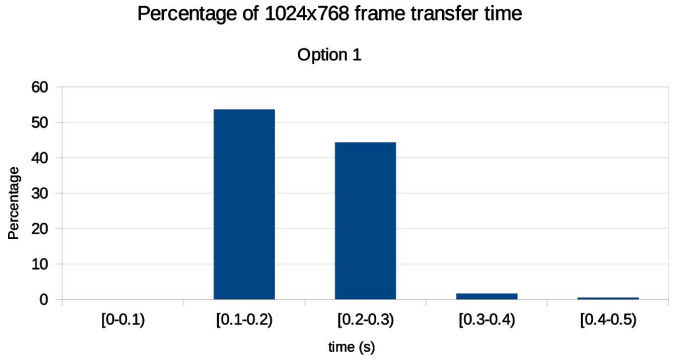


Fig. 7. Histogram of 1024 × 768 frames using option 1 and 802.11ac WiFi.

communication overheads in exchange of higher frequency of the whole system.

The second test measures the latencies of the whole application, that is, the difference between the time the stereo frame pair was taken and the time $d_{\text{coll}}(K)$ was calculated for them. Table IV shows that the main cause of delays in the system are again the communication overheads. Thus, an offloading model that minimizes them is benefited. The robot on its own is unable to get acceptable latencies; this time due to the heavy computation. With these two tests, it is certain that option 1 is the best offloading model for this case, obtaining decent results even with wireless technologies.

Finally, as communication latencies cannot be constant, it is interesting to show a histogram for t_o (in Fig. 7, 1024 × 768 stereo frame pairs were used for option 1 and Wifi AC). Although variance is not very big, some messages (only a 2%) present a high variability of the latency. This is due to four main factors: 1) the ROS middleware used; 2) complexity of 3DPC filters; 3) TCP protocols; and 4) interferences, and packet rejection with backoff periods in MAC layer. As percentage of late packet is very low, a simple predictive timing correction algorithm (see Section III) mitigates possible oscillations in the control loop. Besides, further improvements in *802.11ac* MAC layer like time-division multiple-access (TDMA) protocols, would reduce substantially this latency variance. An alternative to TDMA protocols is the use of the Contention Free Period in the infrastructure mode with fixed size packets. This could not reduce the variability as much as the use of TDMA, but it guarantees a minimum bandwidth reservation, which may be suitable for many real-time systems.

C. Analysis of a Real Navigation Case

Two final tests show that cloud-based robot navigation (option 1) is possible and even can improve its on-board counterpart. The vehicle has been configured with: $v_{\max} = 0,4$ m/s, $\omega_{\max\_\text{robot}} = 1$ rad/s, maximum linear acceleration $a_{\max} = 8m/s^2$, maximum linear brake $a_{\text{brake}} = 10$ m/s^2. Circuit of the first test contains obstacle free areas (where the robot is able to move at maximum speed) and four walls, where the robot should stop to prevent a collision. Figs. 8 and 9 demonstrate the viability of option 1. Fig. 8 shows $d_{\text{coll}}(K)$ when approaching one of these walls. Fig. 9 shows the difference between the joystick linear velocity and the final computed linear speed command when completing a turn of the circuit.

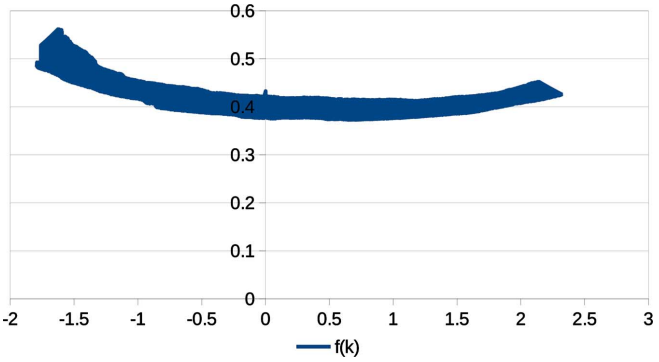


Fig. 8. Example of $s = d_{\text{coll}}(K)$ (in meters).

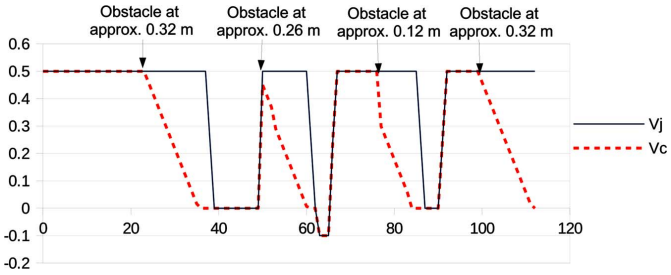


Fig. 9. Response of the navigation assistant to joystick commands.

TABLE V
NAVIGATION RESULTS FOR TEST 2 USING OPTIONS 1, 4, AND TWO FRAME RESOLUTIONS. THE RATIO IS CALCULATED OVER 70 MANEUVERS

Resolution	Option	# Collisions	Ratio
640x480	1	7	0.10
640x480	4	32	0.46
320x240	1	38	0.54
320x240	4	31	0.44

The second test compares the navigation of the robot in an office environment for options 1 and 4, and for low and medium resolutions (320 × 240 and 640 × 480 pixels) at 10 fps. The circuit is completed ten times per case. The number of collisions are measured as a quality magnitude, because, due to the delay variances in the navigation assistant computation, some collisions are difficult to avoid. For each obstacle approaching maneuver, the joystick is pushed to its extreme positions (desired commands are always those of maximum speeds), in order to test the shared control in the worst conditions. As there are seven of these maneuvers for each turn of the circuit, we have a total of 70 possible collisions. Table V shows the number and ratio of collisions for the four tested alternatives. The first conclusion is that for the stereo algorithm used here, low resolutions are not enough to detect properly some of the walls (when the robot is in movement), and collisions are very frequent for any computing option. The second important conclusion is obtained

for the 640×480 pixel resolution. Collision rate is very much reduced when the cloud is working, which is mainly explained because frame processing frequencies are more than double for option 1 than for 4 (see Table III). In this case, the only collisions are produced against an metallic furniture which texture is very much homogeneous. Another additional trouble is the limited FOV of the camera. This means that unavoidable collisions are produced when robot makes brusque turns (this must be solved by the inclusion of more cameras in the future). A demonstration video can be found in [34].

VI. Conclusion

This work analyzes and shows that running a vision based navigation assistant completely on an external cloud is feasible. Two main evidences allow this: a) cloud scalability is pretty well achieved and b) processing times run in parallel to the transmission ones, being the latter the bottleneck of the cloud offloading. In fact, the mean processing frequencies are almost proportional to bandwidth network. It is expected that the extension and natural evolution of wireless networks would improve their bandwidth in the next few years. Experiments demonstrate that the proper computation offloading model must be carefully chosen. In that sense, the computation versus communication tradeoff has to be analyzed. Moreover, the key to success in computation offloading is the scalability of the developed solutions. Finally, the implemented prototype (a teleoperated mobile robot using shared control) can be useful for the teleoperation of wheelchairs of other service robots, which can be done by the user or by an external caregiver. As a better navigation is obtained when using higher resolution images, the necessity of the cloud has been empirically demonstrated. It is also shown that the navigation assistant must contemplate a predictive temporal correction that should prevent a possible collision, caused by the sensoring delay, mainly due to the transmission latencies.

Acknowledgment

The authors wish to thank Prof. D. Cascado for his interesting comments. In addition to this, they really wish to thank the reviewers for all the feedback of our work, together with all the recommendations for future work.

References

[1] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The RoboEarth cloud engine," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013, pp. 438–444.

[2] E. Guizzo, "Robots with their heads in the clouds," *IEEE Spectrum*, vol. 48, no. 3, pp. 16–18, Mar. 2011.

[3] R. Arumugam, V. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. Kumar, K. D. Meng, and G. W. Kit, "DAvinCi: A cloud computing framework for service robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 3084–3089.

[4] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Autom. Sci. Eng. (T-ASE)*, vol. 12, no. 2, pp. 398–409, Apr. 2015.

[5] *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer-Verlag, 2008.

[6] J. Laird, *The Soar Cognitive Architecture*. Cambridge, MA, USA: MIT Press, 2012.

[7] B. D. Gouveia, D. Portugal, D. C. Silva, and L. Marques, "Computation sharing in distributed robotic systems: A case study on SLAM," *IEEE Trans. Autom. Sci. Eng. (T-ASE)*, vol. 12, no. 2, pp. 410–422, Apr. 2015.

[8] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "RoboEarth," *IEEE Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, Jun. 2011.

[9] J. Sevillano, D. Cascado, D. Cagigas, S. Vicente, C. Lujan, and F. del Rio, "A real-time wireless sensor network for wheelchair navigation," in *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. (AICCSA)*, May 2009, pp. 103–108.

[10] J. Fu, W. Hao, T. White, Y. Yan, M. Jones, and Y.-K. Jan, "Capturing and analyzing wheelchair maneuvering patterns with mobile cloud computing," in *Proc. 35th Annu. Int. Conf. IEEE Eng. Med. Bio. Soc. (EMBC)*, Jul. 2013, pp. 2419–2422.

[11] A. Civit-Balcells, F. Diaz Del Rio, G. Jimenez, J. Sevillano, C. Amaya, and S. Vicente, "SIRIUS: Improving the maneuverability of powered wheelchairs," in *Proc. Int. Conf. Control Appl.*, 2002, vol. 2, pp. 790–795.

[12] S. V. Diaz, C. A. Rodriguez, F. Diaz del Rio, A. C. Balcells, and D. C. Muniz, "TetraNauta: A intelligent wheelchair for users with very severe mobility restrictions," in *Proc. Int. Conf. Control Appl.*, 2002, vol. 2, pp. 778–783.

[13] P. I. Blasco, F. Diaz-del Rio, M. C. Romero-Ternero, D. Cagigas-Muiz, and S. Vicente-Diaz, "Robotics software frameworks for multi-agent robotic systems development," *Robot. Auton. Syst.*, vol. 60, no. 6, pp. 803–821, Jun. 2012.

[14] E. Charfi, L. Chaari, and L. Kamoun, "PHY/MAC enhancements and QoS mechanisms for very high throughput WLANs: A survey," *IEEE Commun. Surveys Tutorials*, vol. 15, no. 4, pp. 1714–1735, 2013.

[15] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Waltham, MA, USA: Morgan Kaufmann, 2012.

[16] B. P. John, *Effectiveness of SPEC CPU2006 and Multimedia Applications on Intel's Single, Dual and Quad Core Processors*. Ann Arbor, MI, USA: ProQuest, 2009.

[17] M. Tenorth, A. Perzylo, R. Lafrenz, and M. Beetz, "The RoboEarth language: Representing and exchanging knowledge about actions, objects, and environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 1284–1289.

[18] M. Tenorth, A. Perzylo, R. Lafrenz, and M. Beetz, "Representation and exchange of knowledge about actions, objects, and environments in the RoboEarth framework," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 643–651, Jul. 2013.

[19] M. Tenorth, K. Kamei, S. Satake, T. Miyashita, and N. Hagita, "Building knowledge-enabled cloud robotics applications using the ubiquitous network robot platform," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Nov. 2013, pp. 5716–5721.

[20] L. Wang, M. Liu, and M. Meng, "Real-time multi-sensor data retrieval for cloud robotic systems," *IEEE Trans. Autom. Sci. Eng. (T-ASE)*, vol. 12, no. 2, pp. 507–518, Apr. 2015.

[21] H. M. Do, C. J. Mouser, Y. Gu, W. Sheng, S. Honarvar, and T. Chen, "An open platform telepresence robot with natural human interface," in *Proc. IEEE 3rd Annu. Int. Conf. Cyber Technol. Autom., Control, Intell. Syst. (CYBER)*, Nanjing, China, May 2013, pp. 81–86.

[22] K. Ayush and N. K. Agarwal, "Real time visual SLAM using cloud computing," in *Proc. 4th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Tiruchengode, India, Jul. 2013, pp. 1–7.

[23] L. Riazuelo, J. Civera, and J. M. M. Montiel, "C2tam: A cloud framework for cooperative tracking and mapping," *Robot. Auton. Syst.*, vol. 62, no. 4, pp. 401–413, Apr. 2014.

[24] H. Bistry and J. Zhang, "A cloud computing approach to complex robot vision tasks using smart camera systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2010, pp. 3195–3200.

[25] Y. Nimmagadda, K. Kumar, Y.-H. Lu, and C. S. G. Lee, "Real-time moving object recognition and tracking using computation offloading," in *Proc. EEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2010, pp. 2449–2455.

[26] H. Wu, L. Lou, C.-C. Chen, S. Hirche, and K. Kuhnlenz, "Cloud-based networked visual servo control," *IEEE Trans. Ind. Electron.*, vol. 60, no. 2, pp. 554–566, Feb. 2013.

[27] L. Agostinho, L. Olivi, G. Feliciano, F. Paolieri, D. Rodrigues, E. Cardozo, and E. Guimaraes, "A cloud computing environment for supporting networked robotics applications," in *Proc. IEEE 9th Int. Conf. Dependable, Auton. Secure Comput. (DASC)*, Dec. 2011, pp. 1110–1116.

[28] J. Salmerón-García, P. I. Blasco, F. Diaz-del Rio, and D. Cagigas-Muñiz, "Mobile robot motion planning based on cloud computing stereo vision processing," in *Proc. 41st Int. Symp. Robot. (ISR/Robotik)*, Jun. 2014, pp. 1–6.

[29] P. I. Blasco, F. Diaz-del Rio, S. Vicente-Diaz, and D. Cagigas-Muńiz, "The shared control dynamic window approach for non-holonomic semi-autonomous robots," in *Proc. 41st Int. Symp. Robot. (ISR/Robotik)*, Munich, Germany, Jun. 2014, pp. 1–6.
[30] P. Nisbet, "Who's intelligent? Wheelchair, driver or both?," in *Proc. Int. Conf. Control Appl.*, Sep. 2002, vol. 2, pp. 760–765.
[31] A. To, G. Paul, and D. Liu, "Surface-type classification using RGB-d," *IEEE Trans. Autom. Sci. Eng.*, to be published.
[32] M. Kytö;, M. Nuutinen, and O. Pirkko, "Method for measuring stereo camera depth accuracy based on stereoscopic vision," in *Proc. SPIE/IS&T Electron. Imag.*, 2011, pp. 1–9.
[33] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison, "Real-time camera tracking: When is high frame-rate best?," in *Computer Vision ECCV 2012*, ser. Lecture Notes in Computer Science, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Germany: Springer-Verlag, Jan. 2012, pp. 222–235, no. 7578.
[34] "Cloud-based robot navigation assistant using stereo image processing," (accessed Nov. 12, 2014.) [Online]. Available: http://www.rtc.us.es/cloud-based-robot-navigation-assistant-using-stereo-image-processing/

Javier Salmerón-García received the B.Sc. degree in computer engineering and the M.Sc. degree in computer engineering and networks from the University of Seville, Seville, Spain, in 2011 and 2012, respectively, and the M.Sc. degree in software engineering for technical computing from Cranfield University, Cranfield, U.K., in 2011. Currently, he is working towards the Ph.D. degree focusing on cloud robotics at the University of Seville.

His main area of expertise is cloud computing applied to both HPC and robotics. Moreover, he works as a part-time Lecturer at the University of Seville.

Pablo Iñigo Blasco received the Degree in computer engineering degree and the M.S. degree in software engineering and technology from the University of Seville, Seville, Spain, in 2008 and 2010, respectively. Currently, he is working towards the Ph.D. degree at the University of Seville, where his research work is focused on parallel computing on mobile robots navigation and obstacle avoidance and also on robotic software architectures.

For four years he worked for the Spanish Research Council (CSIC) as a Software Architect. In 2010, he joined the Department of Computer Architecture and Technology as a Lecturer. He also became a member of the Robotics and Computer Technology for Rehabilitation Laboratory, where he participated in a couple of national research projects related to mobile robot navigation.

Fernando Díaz-del-Río received the M.S. degree in physics (electronics) and the Ph.D. degree from the University of Seville, Seville, Spain, in 1990 and 1997, respectively.

After working for Abengoa-Control-Data and for the University of Huelva for some time, he joined the Department of Computer Architecture, University of Seville, and has been an Associate Professor since 2000. He has served as Vice-Dean of the Computer Engineering School (2007–2010). He is author/coauthor of more than 40 papers in refereed international journals and conferences, most of them in the field of mobile robot navigation. Mobile robot navigation, bioinspired systems and distributed computing systems are his main research topics. He has participated in more than 20 research projects and contracts (between them, in EU projects FLEX and CAVIAR).

Daniel Cagigas received the Degree in computer science and the Ph.D. degree from the University of the Bask Country, Bilbao, Spain, in 1997 and 2001, respectively.

Since 2001, he has been working full-time as a Lecturer in the Department of Computer Architecture and Technology, and a Researcher with the Robotics and Computer Technology for Rehabilitation Lab, University of Seville (Spain). His main research interests include robotics, embedded systems and artificial intelligence applied to rehabilitation technologies. He has participated in more than ten national and international research projects and contracts, including EU projects CAVIAR and CARDIAC. He is author of over 20 publications in international journals and conferences in the field of mobile robot navigation and user interface design.

Appendix D

Extending Amdahl's Law For the Cloud Computing Era

Authors

- Fernando Díaz del Río
- Javier J. Salmerón García
- José Luis Sevillano Ramos

Publication

Title: IEEE Computer.

Type: Journal Article.

Publisher: IEEE.

Date: Not published yet.

ISSN: 0018-9162.

Extending Amdahl's Law For the Cloud Computing Era

ABSTRACT

This is the era of Cloud Computing. A gradual change from centralized to distributed execution is taking place, but not without some practical issues related to process migrations (transparent middleware, efficient and secure data transmissions, …). Extending Amdahl's law to Cloud Computing will provide a better understanding of the involved factors related to the offloading process such as the time and energy speedups, together with some hints about the future of this era.

Keywords: cloud computing, energy savings, computation offloading

INTRODUCTION

Cloud computing is here to stay. According to Gartner's Hype cycle[1], Cloud computing is mature enough to be in a productive phase. Nowadays, Cloud computing services can be exploited at several levels: Infrastructure-as-a-Service (IaaS), if customers are provided with a virtual machine that can be used in a customized way (e.g., Amazon EC2); Platform-as-a-service (PaaS), when providers offer a framework where customers can develop applications (Google App Engine, Windows Azure); and Software-as-a-Service (SaaS), if users only have access to specific applications (like Microsoft Office Web or Dropbox).

The cloud offers considerable advantages over computing on local devices: automatic scaling; no need to purchase, upgrade or maintain bare-metal hardware; amongst others[2]. Even more of them can be listed in the case of mobile devices: energy savings, together with scarcity of local resources (which is largely tied to energy consumption). Indeed, the potential use of cloud offloading opens new doors to researchers (new programming paradigms, mobile agent software, security and privacy topics, balancing and deploying tools, etc.). However, when applications require some kind of real-time constraint, whether remote execution is faster than local execution would inevitably be called into question. Hence, not only energy consumption but also performance is crucial for these devices.

But why moving certain applications to the Cloud yield extraordinary results whereas offloading others is simply out of the question? Consider for instance applications based on some type of searching in huge information databases stored in servers. Ranging from the

simple Internet search of a term to a feature-based image searching using Google Goggles app, all of them perform a search over a gigantic amount of possible matching terms. Hence, the offloading overhead time is way lower than the time to obtain the search results. Obviously these apps are done on the server side for two reasons: the stored information and the high amount of computation that is required. Similarly, if user information to be processed was already stored in the cloud, almost no data transfer[5] would be required (only a pointer to the data), so cloud computing would be preferred. But what about those apps where information is captured online from the local device? Can they benefit by offloading today? And tomorrow?

Software developers may want to weigh the pros and cons prior to moving their applications to the Cloud. In order to do so, a simple extension of Amdahl's Law is here presented. Moreover, this will help us to foresee what the future holds for Cloud Computing, together with its forthcoming research challenges.

EXTENDING AMDAHL'S LAW

While originally stated for uniprocessor vs. multiprocessor execution time comparisons, the famous Gene Amdahl's law can be widely applied to any system. Let speedup S be the original execution time divided by an enhanced execution time[3]. If a fraction F (of the original time) is enhanced by a speedup $S_{fraction}$, the overall speedup is:

$$S = \frac{1}{(1 - F) + F / S_{fraction}}$$

Note that Amdahl assumed the extreme case: fraction F was infinitely parallelizable (no overhead times were included), and the remaining fraction, $1–F$, was totally sequential.

To extend Amdahl's law for cloud computing, let us remember the application centralized execution model. Figure 1 depicts the main architectural pieces involved in centralized computation (red square). Next, we will extend it to describe the process involved in remote computation offloading.

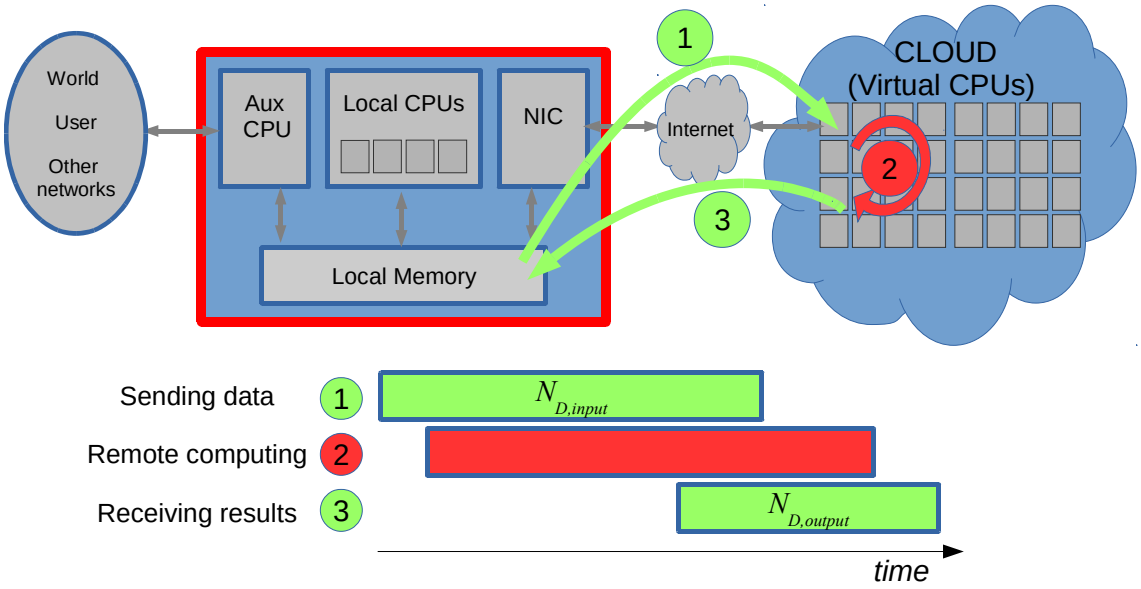


Figure 1. Centralized (red square) and Offloading computation (right and bottom) models.

A. Centralized Computation Performance

Classical centralized architectures consist of a Central Processing Unit (CPU) responsible for doing the following: capturing code instructions and operand data from its local hierarchic memory, executing them, and finally storing the result back in memory. Inside each core, instructions are executed in a sequential fashion according to the Von Neumann model. As a result, execution time for a uniprocessor can be expressed as $N_I\ CPI\ T$ (see sidebar 1), where N_I is the Number of program instructions, CPI, the mean number of cycles per instruction, and T, the clock period. Nowadays, a CPU is composed of a number $N_{c,local}$ of "cores", so according to Amdahl's model, the $N_{c,local}$ cores execute in parallel the fraction F of the program, while the rest is executed by a single core. Namely, $F\ N_I$ instructions are executed in parallel whereas $(1\text{-}F)N_I$ are not. For the sake of simplicity, let us suppose that CPU is a symmetric multicore chip (all cores are identical, currently the most widespread type). Likewise, the interaction with the outer world (input/output subsystem) is commonly assumed to be irrelevant in terms of execution time. Only the connection with the Internet, usually called NIC (Network Interface Controller), is to spend a significant time in our model.

With the previous model, local program execution time is:

$$t_{local} = (1 - F)\, N_I\, CPI_{local}\, T_{local} + \frac{F\, N_I}{N_{c,local}}\, CPI_{local}\, T_{local} = \left[(1 - F) + \frac{F}{N_{c,local}}\right] N_I\, CPI_{local}\, T_{local}$$

B. Cloud performance

If an application is offloaded onto a cloud, the local device must first send its data and code through the Internet (green arrow, step 1, Figure 1). Secondly, the transferred application is executed on the cloud, and finally results are sent back to the local device (green arrow, step 3, Figure 1). From a time-based point of view, the bottom of Figure 1 schematizes the involved times in steps 1, 2, 3, where $N_{Data} = N_{D,intput} + N_{D,output}$ is the total amount in bits of data exchanged with the Cloud.

Some extra simplifications will be assumed as well to calculate cloud execution time:

- The program's code size can be neglected: either it is much smaller than the data size (this is evident when images, videos, big data, etc., are processed) or it already resided in the cloud in its major part (e.g. libraries).
- Data transfer is done at a constant rate BW (communication bandwidth), while its startup latency is negligible (or done in parallel with transmissions).
- Internal Cloud overhead times are not considered, as most of them should occur in parallel with other times.

Indeed, some overlapping can exist between communication and computation times, being the two extreme overlapping cases:

1) If no overlapping existed, cloud execution time would be the sum of communication and computation times:

$$t_{cloud} = \frac{N_{Data}}{BW} + \left[(1-F) + \frac{F}{N_{c,cloud}} \right] N_I CPI_{cloud} T_{cloud} = \frac{N_{Data}}{BW} + (1-F) N_I CPI_{cloud} T_{cloud}$$

As cloud resources can be dynamically scaled up, the number of virtual processors is supposed to be large enough so as $F/N_{c,cloud} \rightarrow 0$.

2) Conversely, if overlapping were complete, communication times would be completely hidden, and hence:

$$t_{cloud,\,overlapping} = (1-F) N_I CPI_{cloud} T_{cloud}$$

C. Comparing Cloud and local performance

For the sake of simplicity, local and cloud CPU technologies are supposed to be similar, that is, $CPI_{cloud} \approx CPI_{local}$; $T_{cloud} \approx T_{local}$. In fact, the cloud should be composed of cutting-edge technology, which means that cloud cores may well be probably faster than local cores. Hence, this simplification works in favor of local machines.

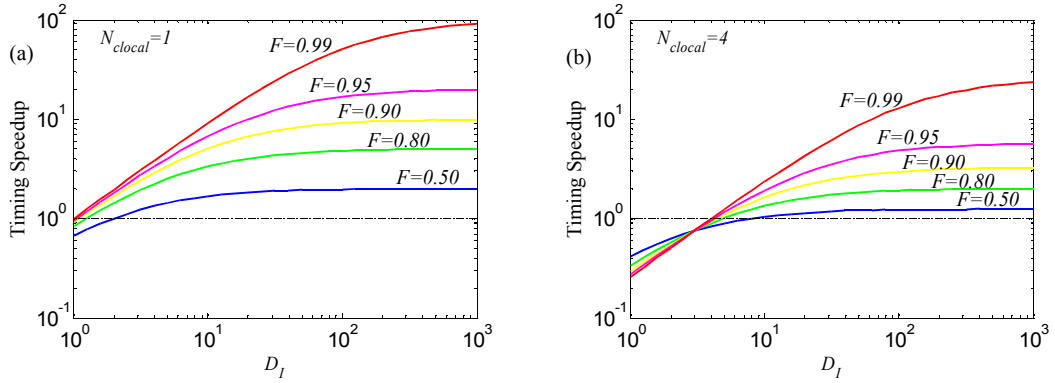
With complete communication-computation overlapping, local execution time will be longer than cloud execution time. This is because communication penalties are not paid while cloud resources are far bigger than local ones. To ponder the worst Cloud case scenario, we will refer

to the aforementioned "no overlapping" model. Dividing numerator and denominator by N_I CPI T and reordering, we get the following speedup for this worst case:

$$S_t(F, N_{c,local}, \mu, D_I) = \frac{\frac{F}{N_{c,local}} + (1 - F)}{\frac{\mu}{D_I} + (1 - F)}$$

For purposes of description, we have defined two parameters. Technological factors are comprised in $\mu = (CPI\ T\ BW)^{-1}$, which can be seen as the ratio between the local machine's capacity of executing instructions per second and core $(CPI\ T)^{-1}$, by its capacity of sending data bits per second (BW). That is, it is reciprocal to the "offloading capabilities" that a machine has. The other parameter $D_I = N_I/N_{data}$ is the "computing density" of an application, that is, the mean number of instructions that are executed for any data bit to be exchanged with the cloud.

Figure 2a shows the speedups S_t as a function of D_I for different F, supposing a modest local device ($N_{c,local}$=1). Note that scales are logarithmic, because S_t rises to high values for large D_I. For high values of F, remote execution is clearly very advantageous when D_I is moderate. But for F<0.5, offloading is faster even for low values of D_I (bigger than 2). To sum up: for simple devices cloud computing can be beneficial in a huge range of applications. This situation is less favorable when the local device is more powerful ($N_{c,local}$=4, Figure 2b). Nevertheless, a new effect then appears: for moderate values of D_I>8, offloading is advantageous for any $F \geq 0.5$. This effect is similar for an extremely powerful device ($N_{c,local}$=16, Figure 2c): from D_I>20, the high consuming local device would not be beneficial.



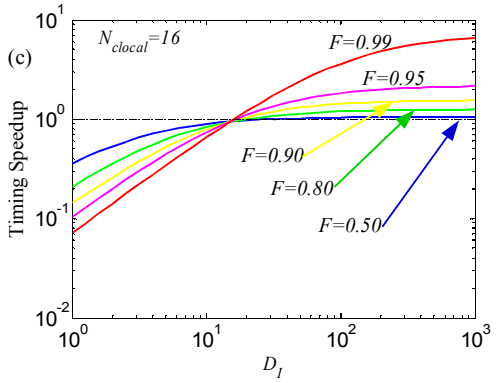


Figure 2. Speedups of remote versus local execution ($N_{c,local}$=1, 4, 16) as a function of D_I for different F and μ=1.

Result 1. Total cloud execution time depends strongly on the amount of communication and computation times that can be overlapped.

Implication 1. This overlap should be thoroughly analyzed by the middleware that manages the task offloading (Sidebar 2 depicts an application case for mobile robots).

Result 2. $\frac{\mu}{D_I}$ plays a critical role in speedup. Nowadays, μ is estimated by a few units (*CPI* rounds 1.0 for most programs, and the orders of T and network BW are 1 ns and 1 Gbps respectively). However, in the near future μ is expected to progressively decrease, as the bandwidth will presumably continue increasing at a geometric rate, and *CPI* T has reached a fixed value (difficult to surpass with present technology, see Sidebar 1). D_I is application dependent and will be analyzed in section 'Characterizing applications'.

Implication 2. As $\frac{\mu}{D_I}$ is expected to decrease considering current technology trends, using the cloud will be more and more beneficial as years go by.

Result 3. When using uniprocessors in the local device ($N_{c,local} = 1$), $S_t = \frac{1}{\frac{\mu}{D_I} + (1 - F)}$; thus offloading would be convenient for applications with low D_I (even if F is less than 0.5).

Implication 3. As long as $\frac{\mu}{D_I}$ decreases in the future, simplifying device hardware may set a trend. This has the additional benefits of saving energy and reducing its software complexity. Note that in IoT (Internet of Things) or "bare" thin client devices, local CPUs (see Figure 1)

would not even exist.

CONSIDERING PERFORMANCE AND ENERGY

Performance speedup is useful for any system, but for mobile devices there is another extremely serious constraint when executing computation-demanding applications[4]: their battery lifetime. With current technology, it is known that most of the energy consumption when offloading an application is due to transmission if data to be transmitted is big enough. Therefore, the question is: can an Amdahl's law extension predict if the energy saved by offloading would compensate the energy consumed by local processing[5]?

Energy consumption is the sum of power multiplied by times for the different periods: $\sum_i P_i \times t_i$. Using the model by Woo and Lee [6], the local execution has two periods: the parallel and the sequential one. During the fraction of parallel execution time F, all the cores are involved, so power is $N_{c,local}P_1$, P_1 being the power of a single core. During the sequential period (1-F), the power is $k_{idle}P_1(N_{c,local}-1)+P_1$, as one core is fully active while the rest are in an idle state (which consume $k_{idle}P_1$, $k_{idle}<1$). All along this study only the energy consumed by the analyzed application is to be considered (ignoring the rest of the system or processes). With these assumptions, consumed energy will be[6]:

$$E_{local} = [(1-F)(N_{c,local}-1)k_{idle}+1]P_1(N_I CPI T)$$

Once again for the case of cloud computing, local energy consumption depends on the amount of overlapping between communication and computation times, being obviously lowest in case of complete overlapping. Similarly to previous sections, we consider the least favorable case for the cloud (no overlapping). Given that case, local energy would be obtained by the sum of communication periods and the cloud computation one. Data transmission adds an extra power P_t. If we assume that NIC is transmitting data by directly accessing the local memory, during both periods all the local cores could be in an off state. Hence, the energy wasted by the local device is:

$$E_{cloud} = \frac{N_D}{BW} P_t + [(1-F)(N_I CPI T) + \frac{N_D}{BW}] N_{c,local} k_{off} P_1$$

During the periods when the local device is waiting for the cloud response, the current application does not need the local device to be running. It is evident that the device can be awake in order to manage other inner tasks, but this energy consumption cannot be attributed to the offloaded application that we are analyzing. From the application point of view, the local device could be almost fully off (only waiting for the NIC), which means that k_{off} is negligible.

The magnitude that comprises all the energy-efficiency factors is the performance achievable in the same battery life cycle[6] (that is, with the same energy). The resultant speed-up $S_{t\times E}$ is given by:

$$S_{t\times E}=\frac{E_{local}\,t_{local}}{E_{cloud}\,t_{cloud}}=S_E S_t$$

Finally, reordering and using the same parameters μ, D_I, we obtain (assuming that k_{off} =0):

$$S_{t\times E}=\frac{[(1-F)(N_{c,local}-1)k_{idle}+1]P_1}{\frac{\mu}{D_I}P_t}\times\frac{\frac{F}{N_{c,local}}+(1-F)}{\frac{\mu}{D_I}+(1-F)}$$

Result 1. D_I is the fundamental parameter in order to know if offloading is energetically beneficial. Using technological magnitudes for a typical mobile device (μ=1, k_{idle}=0.3 -same as Woo and Lee[6]-, $P_1 = P_t$ = 1W), $S_{t\times E}$ becomes very favorable to the cloud from moderate values of D_I. For simple devices ($N_{c,local}$=1), having a D_I greater than 1.3, migration is advantageous ($S_{t\times E}$>1) for any $F\geq 0.5$. For more powerful devices, the bounds grow a little. Moreover, for applications with low F, migration is more favorable even for lower D_I (the more powerful the device, the more notable this effect).

Implication 1. Middleware designers should target how to increase D_I (by compressing techniques and good data coding). Furthermore, software engineers should estimate if future versions of an application would tend to increase D_I (see next section). In general, cloud offloading would not be beneficial for applications with low D_I, for instance those that do not reuse input data like video or audio streaming.

Result 2. If μ continues with its expected reduction, these bounds will decrease close to a proportional rate. That is, if μ were reduced by one-tenth, the bounds on D_I would go down to approximately 0.1, 0.2, 0.4 (for $N_{c,local}$=1, 4, 16 respectively).

Implication 2. Future evolution for current technology is promoting offloading computation for embedded devices.

Result 3. The cases $F\rightarrow 1$ (which is common in most of scientific applications, see next section) or $N_{c,local}$=1 show a speedup almost proportional to $S_{t\times E}=\left(\frac{D_I}{\mu}\right)^2$.

Implication 3. For simple devices or very parallel applications, energy efficiency of cloud migration is expected to be reached way earlier than that of timing speedup (quadratic order).

CHARACTERIZING APPLICATIONS

Would a given application be benefited by offloading? Let us analyze several applications in view of our model.

The first example is a very computationally intensive algorithm, matrix multiplication, which is the kernel of many scientific applications. Let $A=B\times C$ be a product of $n\times n$ ranked matrices. Basically, it consists of these three loops:

```
for row = 1...n
    for col = 1...n
        for  k = 1...n
            A[row][col] += B[row][k] * C[k][col]
```

The two outer loops iterate over the elements of A, while the inner loop computes the dot product of a row of the first matrix by a column of the second matrix. The total amount of multiply operations is the same as the iterations of the inner loop: an order of $O(n^3)$. For most scientific applications, the bigger n is, the more accurate the results will be. However, the high computational order (together with the elevated amount of memory) makes many programmers reluctant to use big matrices.

What if the application were offloaded? B and C must be sent to the cloud, and after the processing, A must be returned back, that is, $O(n^2)$ bits to be exchanged. Hence, the order of D_I becomes: $D_I = \frac{O(n^3)}{O(n^2)} = O(n)$. Besides, all the elements of A can be computed in parallel, and only the sum of products -i.e. $O(n)$- must be done sequentially, which gives this dependence of $1\text{-}F$: $1 - F = \frac{O(n)}{O(n^3)} = O(n^{-2})$. Finally, the dependence of speedups is: $S_t = O(n)$, $S_{t \times E} = O(n^2)$. Hence, the more accurate the results were wanted, the more speedups from cloud computation could be extracted. To sum up, not only μ, but also F, D_I are playing in favor of offloading.

The second example plays a crucial role for many robotic applications: processing frame pairs captured by a stereo camera. In Sidebar 2, it is shown that this processing can be designed, not only to be parallel but also with an efficient pipeline between processing and transmission times. Hence parallel ratio F is very near to 1 for a medium resolution image. This sidebar shows that usual complexities of these algorithms are $O(N^3)$, being N the horizontal image size. As transmitted data are proportional to image resolution, that is, $O(N^2)$, we find again that D_I is $O(N)$. Once again, the more resolution the image has, the more speedups from cloud computation can be extracted.

The previous reasoning can be extended to most cases. For instance, scientific applications[3] are usually parallel and have complexity orders that vary between $O(n \log n)$ -e.g. Fast Fourier Transform- and $O(n)$ -e.g. finite element-based applications- for a data size n. This means that while F approaches 1, D_I grows like $O(\log n)$ or remains constant. In the first case, once again, the more data (i.e. accuracy) are used, the more speedups are achieved, while in the second case the benefits will be only accomplished when technological progress makes μ bigger.

Result 1. For many applications, F and D_I grow with the order of the problem.

Implication 1. As new applications require more accurate solutions, using the cloud becomes a more viable solution. Furthermore, one of the software designers and researchers' objectives

should be to decrease N_{Data} (by compressing techniques and good data coding[7] in order to make D_I grow).

Result 2. For the case F=1, the extreme performance that can be achieved for a complete system is depicted in Figure 3. Abscissa represents different values of D_I, and ordinate, GIPS (Giga Instructions Per Second). This graph has been obtained for the real case of a common current system (Snapdragon 610 S4 Pro, with 1.5 GHz quad-core Krait 300). This can be seen as an extension of the Roofline Model[8] for GIPS, instead of GFLOPS (Giga FLoating point Operations Per Second). The result is a Two-Roofline Model. Maximum device GIPS is obtained as the inverse of $\frac{clocks}{instruction} \times \frac{seconds}{clock}$, multiplied by the number of cores $N_{c,local}$, that is, $(CPI_{minimum}\ T)^{-1} N_{c,local}$. Theoretical maximum Cloud GIPS would be obtained in the same way (if cloud resources were finite). When execution implies many RAM accesses, this maximum cannot be reached: the product $\frac{instructions}{databit} \times \frac{databit}{seconds}$, i.e., $D_I\ BW_{RAM}$, gives the left line (the first roof-line). Second roof-line is obtained for the network connection between the device and the cloud: $D_I\ BW$.

Implication 2. Contrary to initial thoughts, remote execution of "bad" applications (those that have poor CPI and real GIPS are far from the maximum[3], as marked with down arrows in Figure 3) may be the best option. If the prolongation of second roof-line (dotted red line) crossed real device GIPS, remote execution would obtain the same GIPS as local one. Other benefits, like energy savings, may stimulate the remote option.

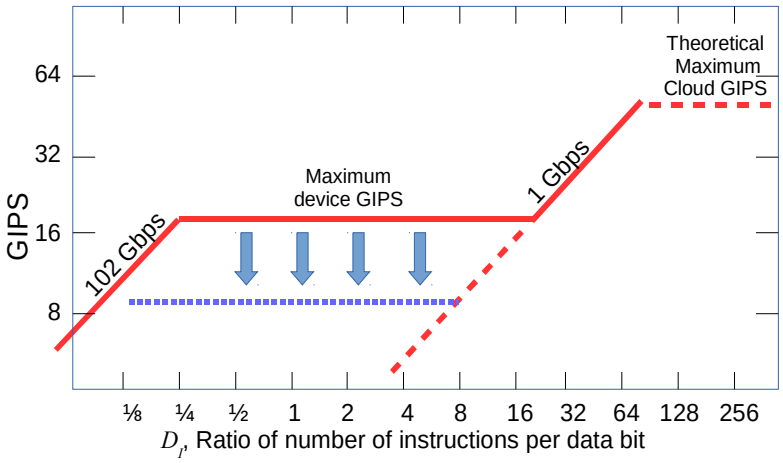


Figure 3. Two-Roofline Model for GIPS (Giga Instructions Per Second) vs. D_I for the Snapdragon 610 S4 Pro. The diminution of maximum device GIPS is marked with down arrows.

LESSONS FROM HISTORY

For those applications that required vast amounts of data residing in a server, cloud computing is the only option. But, what can our cloud computing Amdahl's law predict about the rest of applications that elude offloading?

With respect to technological factors like μ, it is clear that its evolution would benefit cloud computing for the next decade if they continue following Moore's Laws.

What about those factors, like F and D_I, which depend on holistic aspects? This is no simple matter. Nevertheless, there are some valuable lessons from history. Amdahl argued that values of 1-F were large enough to favor single processors. But now enhanced machines allow massive computations difficult to be envisioned by him[9], making F soar to values very close to 1. Thus, F can be stretched as more computation resources become available.

Let us visualize the expected evolution of D_I. History has shown us that "software is a gas: it expands to fill any size hardware container"[10] as one of Myhrvold's Laws states. Imagine that users or developers noticed that a remote application is more powerful (and maybe faster and more energy-saving) than its local counterpart, which runs in a smaller hardware container. Or let us consider the case when mobile programmers discover that they have unlimited computation power in the Cloud. We would wager that users would ditch the local option and embrace the Cloud, and that the efforts needed to transform a complex algorithm into one that would run fast on a low-power computer, could instead be used for other more productive purposes. Hence, it is highly likely that D_I will increase when all programmers easily and transparently use the cloud. To sum up, S_t and $S_{t \times E}$ are being pushed up by the three factors.

Even those applications that must run close to the target system show a tendency to be offloaded. Nowadays, some platforms like *mbed.org* offer a remote compilation for embedded systems and the possibility of generating a development project to be loaded into a certain device. Let us canvas two embedded system programmers: one is developing directly over the system, and the other one is supplied with the cloud to develop with her/his comfortable, competitive, usable, accessible, favorite, beloved (and so on) environment. Who will develop faster and more pleasantly? Obvious, isn't it?

ENDINGS

Undoubtedly, the real world is much more complex than a theoretical model, which tries to encompass the main trends and must include several assumptions. Some of them benefit the cloud execution time, like forgetting virtualization, middleware, O.S. overheads, or the scheduling software tasks to divide a program on local and external computation. Others benefit the local execution time, like considering no overlap between communication and computation, supposing that resources (memory, caches, bus speeds, etc.) are identical for cloud and local system, etc. Some studies in the literature show how complex this puzzle of practical issues can

be[11].

Two main practical issues make the cloud model more attractive. Firstly, the commercial success of cloud facilities is not without substantial R&D investments, approaching the real cloud execution time to that of our assumptions. Secondly, migrating an application to the cloud would be very cost-effective, because it would suppose a simplification of the local device, being the offload cost minimal. This probably will be brought out with efficient networking and distributing computing techniques. Which indeed will highly likely pave the way for new programming paradigms that contemplate automatic code migration and distributed computing as a new form of computation.

SIDEBAR 1: COMPONENTS OF THE EXECUTION TIME

Almost any computer machine works like a finite state machine (*FSM*). It can be considered that almost everything is synchronous with a clock which period is *T*, being the Central Processing Unit (CPU) the heart of this *FSM*. Therefore, the execution time t_{exec} of any program must be multiple of N_{clocks} clock periods:, that is, $t_{exec} = N_{clocks} T$. Likewise, Von Neumann model, settled 60 years ago, is the current model of the vast majority of modern computers. Von Neumann postulated that a program would consist of a number, namely N_I, of instructions that would be executed in sequential order. e sequential execution is not an efficient (fast) way to make a computation, it is the common form that humans have to express the solution of problems, and hence most computing languages stick to this representation.

N_{clocks} can be split like: $N_{clocks} = N_I \frac{N_{clocks}}{N_I} = N_I CPI$, and therefore: $t_{exec} = N_I CPI T$. This is the fundamental formula of computer architects, and each of the 3 factors that contains (N_I, *CPI*, *T*) do play a role in the design of a microprocessor.

CPI (Clocks Per Instruction) comprises the "ability" that a CPU has to execute many instructions per clock. During the first stages of computer era, *CPI* was relatively high (several clocks per instruction), but with the advent of RISC (Reduced Instruction Set Machines) machines from 1985, architects could design efficient CPU pipelines that execute several instructions in just one clock cycle (*CPI*<1). Thus execution times could be reduced progressively due to *CPI* diminution and period contraction. But at the end of last century architectural innovations come to a limit. Today ideal *CPI* is given by the width of processor issue stage, which has been stuck around 5 for the last 15 years, although CPU stall cycles makes this quantity to be more than one (for a representative set of benchmarks). To make matters worse, around 2005 CPU period comes to a limit as well. These constraints had promoted the advent of the so-called "multicore Era".

SIDEBAR 2: STEREO VISION OFFLOADING FOR MOBILE ROBOTS

Mobile robots have reached an elevated degree of maturity during the last decade. As an example, the time where fully autonomous cars are commercially available is now approaching. One of the points that has motivated this is the advancements on sensoring information processing, like stereo vision. This was allowed mainly due to the advent of more powerful parallel architectures, and to the availability of distributed operating systems, like the so-called RSF (Robotics Software Frameworks). RSFs have recently allowed to improve aspects such as scalability, reusability, deployment, and debugging[1], and to deploy tasks either on board or on a cloud computing system, depending on their constraints[2].

Stereo vision is currently a very active field of research. When processing stereo frame pairs, the step with a higher degree of time complexity is the analysis of their differences, that is, the difference between what left and right eyes see. The so-called disparity map permits to calculate the distance of the objects to the cameras, more or less accurately in function of the used algorithm and the image features. Most representative algorithms have time complexities ranging from $O(N^2)$ to $O(N^{11})$, where N is the horizontal image size[3]. One of the most extended algorithm is that included in OpenCV[4], which has a complexity of $O(N^3)$ -the usual for most of them-. The key problem is that real time requirements can become very difficult to meet when more accurate reconstruction of the environment is demanded. The frequency of stereo frame processing must be sufficiently high, whereas its latency small enough. For example, when the robot is moving, the distance to the nearest obstacle must be computed soon enough to avoid a crash or an emergency stop. Hence, this heavy task must be processed in a powerful computing system, or -as it is usual nowadays- carried out with low resolution images.

Thus, cloud-based stereo vision is a possibility that is gaining more place. The offloading processing must be designed, not only to be parallel, but also to exploit the dynamic scalability of the cloud system. Concretely, if multiple CPU cores were available, processing times can run in parallel to the transfer ones, being the latter the bottleneck of the cloud offloading[2]. As a result, the mean processing frequencies result almost proportional to bandwidth network.

REFERENCES (SIDEBAR 2)

1. P. Iñigo-Blasco et al., “Robotics software frameworks for multi-agent robotic systems development,” *Robot. Auton. Syst.*, vol. 60, no. 6, pp. 803–821, Jun. 2012.
2. J. Salmeron-Garcia et al. “A trade-off analysis of a cloud-based robot navigation assistant using stereo image processing,” *IEEE TASE*, vol. 12, no. 2, pp. 444–454, April 2015.
3. M. Sizintsev and R. P. Wildes, “Coarse-to-fine stereo vision with accurate 3D boundaries,” *Image Vis. Comput.*, vol. 28, no. 3, pp. 352–366, Mar. 2010.
4. “OpenCV.org: About.” [Online]. Available: http://opencv.org/about.html. [Accessed: 07-

Aug-2014].

Acknowledgements

This work has been supported by the Spanish grant (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02/01).

References

1. "Hype Cycle for Application Architecture, 2013." [Online]. Available: https://www.gartner.com/doc/2569522/hype-cycle-application-architecture-. [Accessed: 31-Jul-2014].
2. M. Armbrust et al. *Above the Clouds: A Berkeley View of Cloud Computing*. EECS Department, University of California, Berkeley, February 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.
3. J. L. Hennessy, D.A. Patterson, *Computer Architecture, Fifth Edition: A Quantitative Approach*, 5 edition. Morgan Kaufmann, 2011.
4. R. Friedman, A. Kogan, and Y. Krivolapov, "On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones," *IEEE Trans. Mob. Comput.*, vol. 12, no. 7, pp. 1363–1376, Jul. 2013.
5. K. Kumar and Y.-H. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
6. D. H. Woo and H.-H. S. Lee, "Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era," *Computer*, vol. 41, no. 12, pp. 24–31, Dec. 2008.
7. E. Tilevich and Y.-W. Kwon, "Cloud-Based Execution to Improve Mobile Application Energy Efficiency," *Computer*, vol. 47, no. 1, pp. 75–77, Jan. 2014.
8. S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009.
9. M. D. Hill and M. R. Marty, "Amdahl's Law in the Multicore Era," *Computer*, vol. 41, no. 7, pp. 33–38, Jul. 2008.
10. W. Wayt Gibbs, "Taking Computers To Task," *Sci. Am.*, Jul. 1997.
11. M. S. Gordon et al. "COMET: Code Offload by Migrating Execution Transparently." *Proc. 10th USENIX Conf.* (OSDI'12), 2012, pp. 93–106.

BIOGRAPHIES

Fernando Díaz-del-Río *received his Master in Physics (Electronics) and his Ph.D. from the University of Seville (Spain) in 1997, and currently is an Associate Professor at this University. Mobile robot navigation, bioinspired systems and distributed computing systems are his main research topics. He has participated in more than 20 research projects and contracts. Contact him at fdiaz@us.es*

Javier Salmerón-García *received his BsC in Computer Engineering (2011) and MSc in Computer Engineering and Networks (2012) from the University of Seville. He also received an MSc in Software Engineering for Technical Computing from Cranfield University (UK, 2011). He is now doing a PhD focused on Cloud Robotics in the University of Seville. Moreover, he works as a part-time lecturer at the same University. Contact him at jsalmeron2@us.es.*

José Luis Sevillano *is an Associate Professor at the University of Seville, Spain. He served as VP Membership of SCS (2009-11), and currently serves as Associate Editor of Simulation (Sage) and of the International Journal of Communication Systems (Wiley). He is an IEEE Senior Member, has co-authored 80+ papers and has participated in 20+ research projects. His research interests include real-time communications and architectures, Mobile Robots and eHealth and Rehabilitation Systems. Contact him at jlsevillano@us.es*