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 However, instead of disposing these "outdated" embedded systems, Cloud technologies for computation offloading can be used.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

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

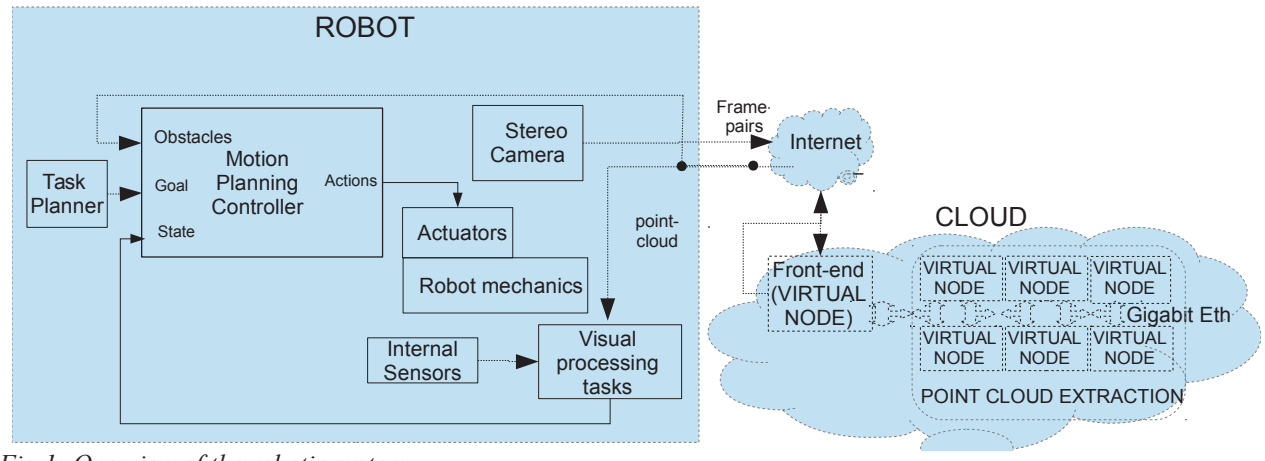

Fig 1: Overview of the robotic system

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).

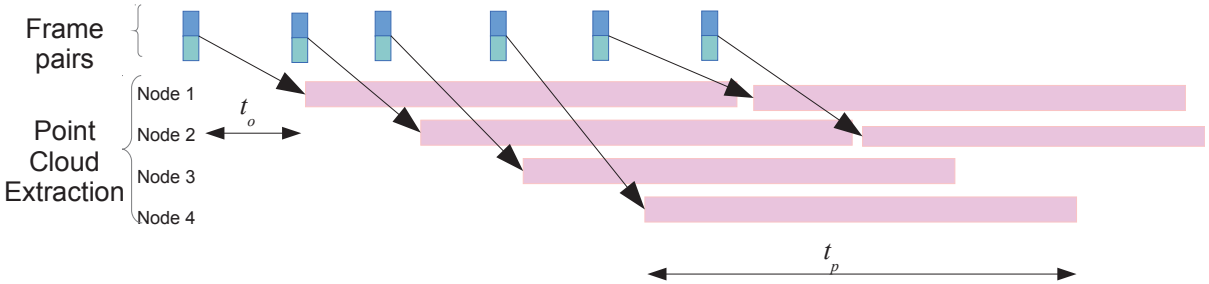

Fig 2: Stereo frame pipeline process. Four nodes process (in a pipeline fashion) the frame pairs that the front-end node delivers in a round-robin form.

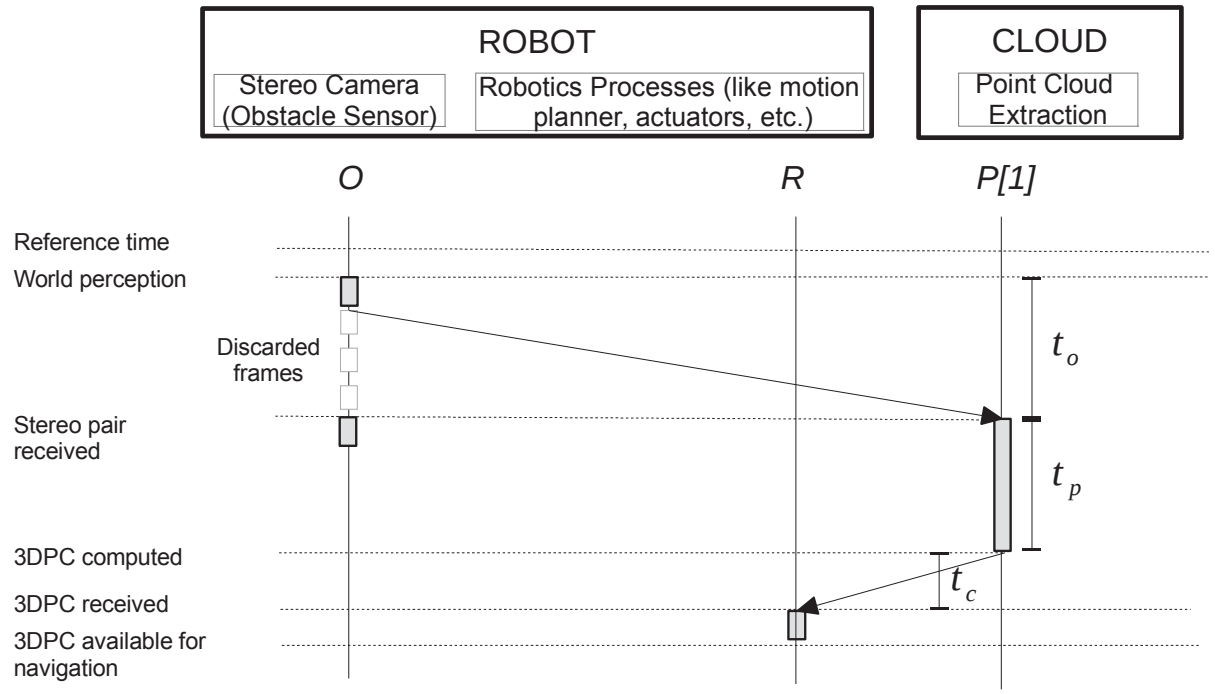

Fig 3: Time diagram of the point cloud extraction for one virtual node. More virtual nodes suppose that more processes P will be running in parallel with different frames, so a little number of frames would be discarded.

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.

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.

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

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.

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-

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. .