Remote Control within the UJI Robotics Manufacturing Cell using FPGA-based vision

Raul Marín, Germán León, Raul Wirz, Jorge Sales, José M. Claver and Pedro J. Sanz

Abstract— **In this paper we present a work in progress of a new remote control system based on networked robots and FPGAs technology. The experimental validation has been carried out within the UJI (i.e. the acronym for University Jaume I) Robotics Manufacturing Cell. The main devices included in this Cell are: a SCARA manipulator (AdeptOne), a robot arm with six degrees of freedom (Motoman), an industrial belt, several sensors and cameras, an FPGA that takes care of the computer vision algorithms (i.e. including grasping determination), and a distributed architecture that allows any user to control remotely via Internet a specific manufacturing task. The different components of this system are connected by a 100BaseT Ethernet network and follow the SNRP architecture (i.e. Simple Network Robot Protocol), which grants simple access to generic networked devices using enhanced IP (i.e. Internet Protocol) based connections. Moreover, every device in the system is connected to the Internet through a router that permits the user to control the networked devices according to security constraints. The experiments presented in this paper are concerned with learning visual servoing loops in a direct manner. To this end, the industrial Robotics Cell includes a FPGA, which significantly improves the performance of the computer vision algorithms, and then greatly enhances the efficiency of the remote visual servoing control. Results show the system performance of remotely programmed visual servoing controls using different network architectures.**

I. Introduction

Real-time robot vision tasks such as visual servoing and object tracking require high computational power and data throughput, which often exceed the processing capabilities of the processor on computer platforms. In this case, application specific hardware (e.g. ASICs, DSPs, or FPGAs) are considered as alternatives in dealing with this problem [1-5].

Today, FPGAs are competitive to ASICs in terms of capacity and performance. The main disadvantage of the ASIC approach is that the circuit is usually limited to operate for one specific application. This limitation is overcome by the programmability of processor cores included in the ASIC and the introduction of reconfigurability. In [4], reconfigurable multiprocessor networks enable the implementation of a variety of image processing algorithms for low and intermediate level computer vision. Indeed, FPGA can be reprogrammed and allows the rapid prototyping of circuits that has to be designed and able to operate in real time conditions [5]. Thus, FPGA is a promise solution to alleviate the problem of processing speed in computationally intensive applications like image processing and industrial vision systems [6].

Recently, several efforts have focused on the design of FPGA-based DSP processing systems ([7], and therein). But requirements are rapidly changing and increasing in complexity. Furthermore, solutions need to be rapidly designed and updated, portable to the latest most powerful platforms, and integrated into a variety of front-end software application environments.

Current efforts attempt to compile high–level languages such as Matlab directly into FPGA implementations [8]. Other tools use derived languages based on C such as Handel-C, based on C++ class libraries such as System C , or Java classes such as JHDL [9]. Recently, a more affordable approach for system designers has been to use "block-based design", where a graphical tool (GUI) allows the interconnection of parameterisable IP cores from a library for creating processing systems. PixelStreams [10] is an example of this sort of libraries for digital image and video processing.

On the other hand, our research laboratory is very concerned with the necessity of using robotic telelaboratories in order to give the undergraduate and master students the opportunity to learn using a real robotic scenario that can be programmed *in-situ* or even from home via Internet (i.e. open telelaboratory) [11][12].

Moreover, there are some control experiments (e.g. high performance industrial applications) that need more

Manuscript received October 16, 2006. This work has been partially funded by the Spanish Ministry (MEC) under Grants TIC2003-08496, DPI2004-01920, TSI2004-05165-C02-01, TIN2006-15516-C04-02, by the Fundació Caixa Castelló under Grants P1-1B2002-07, P1-1B2003-15, and P1-1A2003-10, and by the Generalitat Valenciana under grant GV04A-698.

R. Marín, G. León, R. Wirz, J. Sales and P. J. Sanz are with the Computer Science Department, University Jaume I (UJI), Castellón 12071, Spain (email: {rmarin, leon, wirz, salesj, sanzp}@uji.es).

J. M. Claver was with University Jaume I (UJI), Castellón 12071, Spain. He is now with the Computer Science Department, University of Valencia, Burjassot 46100 Valencia, Spain (e-mail: jclaver@uv.es).

sophisticated robotic environments and offer the scientists and students a higher precision in the robot positioning and speed. For that, the UJI robotics manufacturing cell was designed, which allows people to remotely program complex and high-performance Industrial Applications via web (see Figure 1).

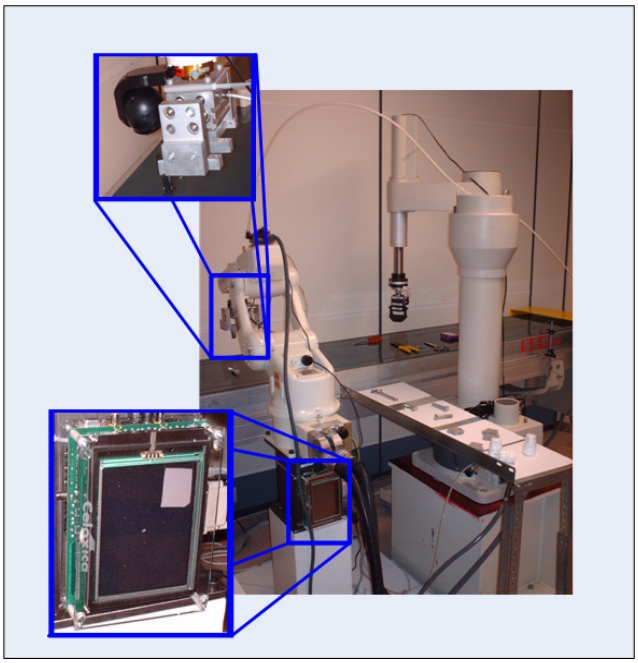

Figure 1. The UJI robotics manufacturing cell: (Top-left) camera-in-hand configuration and (Bottom-left) FPGA used to accelerate the image processing.

One essential part of an Industrial Telelaboratory is the distributed architecture, which enables the interaction between the robotic devices and the user commands. In fact, in this paper we consider that every device (i.e. industrial robot, conveyor belt, FPGA, etc.) is connected to the same Ethernet network, and they act as single Network Robot that communicate with each other through the SNRP web-based protocol (see next section for details). This architecture offers many advantages like scalability and maintainability, and it introduces interesting issues like device synchronization, bandwidth and time-delays.

As it will be explained in detail in the following sections, enabling remote programming of robotic systems permits us to develop external programs that take control over the whole set of robotic functions. Thus, for example, we could design an experiment in Java for performing a closed loop manipulation (i.e. remote visual servoing), or we could even use this interface for designing a voice-operated robot.

A very good example of remote robot programming in order to validate these architectures is in fact the remote visual servoing control. It uses sequences of camera inputs in order to bring the robots to the desired position, in an iterative way. In fact, in this paper we enabled the students and researchers in our university to experiment with their remote visual servoing algorithms through a remote real environment instead of using simulation tools.

Finally, the paper describes the implementation of a SNRP Network camera implemented using a FPGA (see Figure 1). A camera-in-hand gives the input images to the FPGA, which performs the image processing and object segmentation procedures at almost 15 frames per second (fps). This device provides excellent improvements in the remote programming of visual servoing control loops, as explained in detail in the results.

II. Design and Implementation Issues

As explained in [13], Networked Robotics is an emerging research area for creating intelligent robotic architectures that integrate embedded systems, sensor and actuator networks.

The challenge is defining software and network architectures within the network robotics context, providing the following features: (1) Simple, (2) Open, (3) Flexible (4) Dynamic, (5) Robust, (6) Scalable, (7) Efficient, (8) Secure, (9) Platform independent.

Simplicity is maybe the most important challenge of network robotics architecture, due to the fact that it must be possible for a very broad range of devices to be part of it. In fact, as we will describe later, thanks to this simplicity we were able to implement a prototype of SNRP Network Camera using a FPGA. Where SNRP (Simple Network Robot Protocol) is an open based protocol developed by researches at the RobInLab (Robotics Intelligence Laboratory, http://www.robot.uji.es).

In the scientific literature several works can be found that propose different ways and architectures to organize task-oriented applications of multiple network robots [14-17]. Some of these architectures are focused on Internet software frameworks (e.g. Web Services) and have been extended from previous works in single remote controlled robots [18].

Other works focus on the Internet network protocols themselves and study internet transport protocols that enable real-time control and teleoperation of network robots over IP. In fact, as explained in [19, 20], solutions can be found to cope with the problems associated to the Internet in order to control networked robots: (1) time-varying transmission delay, and (2) not-guaranteed bandwidth.

The software architecture of the SNRP framework provides the following modules:

1. SNRPRobot: Every robot/device in the SNRP framework would provide a SNRP_Robot network interface, which allows any client (e.g. user experiment) to use a service provided by itself (e.g. “motoman.service.moveToPosition(x, y, z)”. Examples of these interfaces are “SNRPConveyorBelt”,

"SNRPMotoman", and "SNRPFPGAVision".

2. SNRPRobotsGroup: A SNRP robot can be the union of several SNRP robots (e.g. a Mobile manipulator is the union of a mobile robot, an arm). Moreover, the SNRP module for the arm can be the union of two modules, the one for the gripper and the one for the arm itself. Thus, SNRPRobotsGroup permits defining new services for the several networks robots that work together as if they were a unique robot.

3. SNRPNamingService: A SNRP network robot can register to a naming service in order to select a name (e.g. UJI/telelabs/industrial/motoman) and inform other peers of which IP and port he is attending to.

4. SNRPServiceHolder: The services provided by a SNRP robot can be programmed in a static manner within the SNRP Module itself, or on the other hand, they can be added dynamically in runtime. For that, an SNRP service that follows a given interface most be uploaded into the SNRPServiceHolder. At the moment of writing the industrial telelaboratory has only a SNRPServiceHolder for the whole system. Anyway, the architecture would permit having a holder in every SNRP robot.

5. SNRPExperiment: A SNRP Experiment is a robot service that can be allocated into a service holder. In fact, the experiments that we are performing in this moment provide a unique service holder for the telelaboratory that is located in the Experiment's server computer. Further experiments could be defined as the union of several SNRP services (i.e. agents) that are run concurrently on different service holders and that all together provide a certain robotic task.

Once we have seen the software architecture for the SNRP framework, we are going to focus on the SNRP protocol itself that permits the communication between SNRP experiments, holders, naming services and robots (see a summary displayed in Figure 2).

First of all, as we want the devices to be accessed through the internet; they should manage the IP protocol. On top of it, the SNRP framework enables the device to accept TCP, and UDP connections. As explained before, UDP and TCP are not the best solutions to perform remote control through the Internet, so the SNRP framework provides the possibility to transport the internet datagram's through other protocols like "trinomial" [20], RTP (real-time transport protocol), the RTP (Rate-based adaptation protocol) [21], etc.

On the other hand, concerning the FPGA design, an important improvement to obtain a high-performance SNRP vision system has been demonstrated. A Celoxica RC203E platform based on an FPGA Virtex II XC2V3000 chip was selected. This platform provides a 100BaseT Ethernet connection, video input/output, high-resolution color camera and a TFT display. The video input/output interface provides an easy way to test and verify the FPGA design.

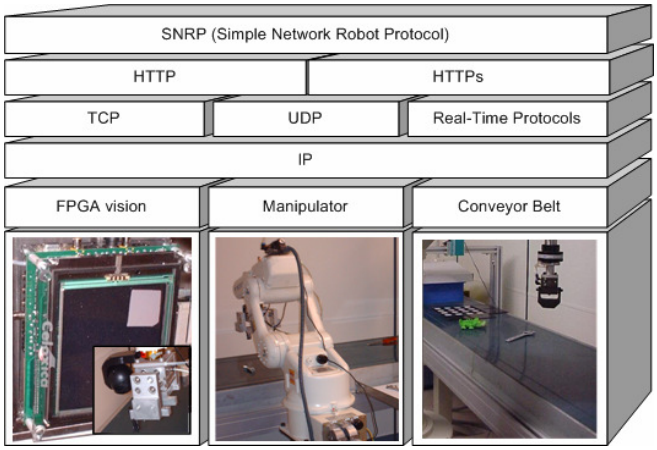

Figure 2. SNRP network architecture

Moreover, the RC203E platform provides a development environment (i.e. DK) based on the Handel-C hardware programming language, which has a similar syntax to the ANSI C language, and it includes the PixelStreams library to design video systems.

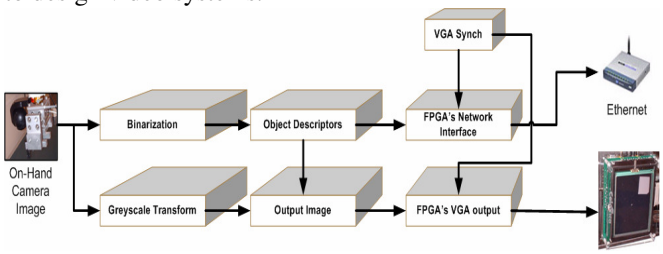

Figure 3. FPGA processing data flow architecture.

The PixelStreams library is very appropriate for these sorts of applications since the circuit supplied is already segmented, it has specific video input/output modules, and it comes with IPs for creating video filters and transforms.

The image processing task performed by means of that FPGA is divided in two principal data flows, as shows the Figure 3. In the first data flow, the image taken by the "on-hand" camera is binarized. Then, object descriptors are obtained by using a specific module designed *ad-hoc* for this purpose. This module is not present in the PixelStreams library and has been implemented from scratch using a high level design tool [23]. Finally, object moments are sent to the network through the FPGA network interface using the SNRP protocol.

In the second data flow, the image is transformed in a grayscale image and combined with visual information from the "Object Descriptors" module. Hence, an augmented reality image, in which a cursor appears indicating the object centroid position, is shown on the TFT display of the FPGA board. The two data flows are synchronized by mean of the "VGA Synch" module in order to avoid conversion and visualization problems.

The FPGA provides a service for informing the client about the object properties in the robotic environment. Since the client/server synchronization is on-demand, the FPGA waits for a UDP datagram containing an SNRP command (i.e. GET /service/object/centroid). Once the input command is checked, the FPGA generates a new UDP datagram contained on an Ethernet frame, which uses the sender MAC and IP address. This process is performed in a few FPGA clock cycles due to the fact that the FPGA does not need to implement the whole TCP/IP stack.

The implementation results are the following:

1. FPGA area usage: 3834 slices (4449 LUTs and 55248 bits of BRAM). It is approximately a 25% of the FPGA area.
2. The maximum FPGA clock cycle is 52.23 ns.
3. The computing cost of object moments is 1,292,835 FPGA clock cycles, that is, approximately 14.8 frames per second (i.e. segmented scenes per second).

III. EXPERIMENTAL RESULTS

A. *The UJI robotics manufacturing cell*

The layout of this cell can be appreciated in Figure 4. It is noticeable that for the present work only the Motoman robot arm is used, discarding the other one (i.e. AdeptOne robot arm).

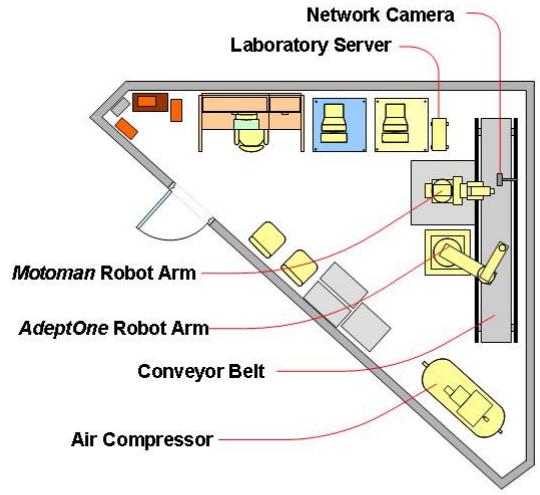

Figure 4. Layout of the UJI Robotics Manufacturing Cell

In order to let a user to implement their industrial application experiments, the UJI robotics manufacturing cell is composed of the main following devices:

1. Motoman manipulator: This 6 degree of freedom robot arm enables users to manipulate the objects coming along the conveyor belt and classifying them in the auxiliary table.

2. FPGA and on-hand camera: The Motoman manipulator is equipped with an on-hand high resolution camera (1024x768) that allows the implementation of visual servoing controls, as well as object tracking algorithms.

3. Conveyor Belt: It enables user to move the objects in any direction, as well as with different speeds. Once an object reaches the end of the belt, two sensors indicate its situation and allow the operator to act accordingly.

4. Fixed Network Camera: On top of the Conveyor Belt a network camera is located that provides real-time video streaming of the objects present onto the conveyor belt. This camera is calibrated with the conveyor belt, which permits the implementation of robot control algorithms using on-top visual servoing techniques.

5. Robot's Server: A pentium III PC is connected to the Motoman manipulator and the conveyor belt, enabling its remote control and programming through the corresponding SNRP servers. This computer holds the software SNRP framework for user authentication, SNRP naming service, the Telelaboratory educational webpage (i.e. moodle based [22]), and the Experiment's Holder Server, which enables uploading a Java experiment to the server and launching it when appropriate.

Figure 5 shows the software architecture, including Network connectivity of the UJI robotics manufacturing cell. As a way to enhance the security measures, the whole telelaboratory is accessible through a unique router/firewall, which permits defining the devices and services (i.e. ports) accessible by the user remotely in a very detailed way.

In order to establish a comparative evaluation with the proposed system based on FPGA and SNRP protocol (i.e. "FPGA/SNRP" experiment), an eye-in-hand control experiment has been designed. A standard PC and a CORBA protocol are used with the same test to perform this comparativeness (i.e. "PC/CORBA" experiment).

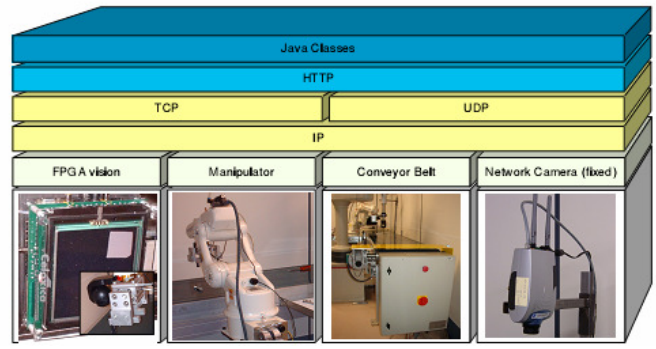

Figure 5. SNRP Software architecture.

B. *Remote Experimental Results*

Note that the whole scene must be binarized and segmented every time the robot moves, in order to acquire the required mathematical features for every object.

The gripper fingers of the robot will appear in the scene as two independent objects, so that the user has an

additional difficulty in order to calculate the next robot movement, bringing the centroid of the object to the centroid of the gripper by iterative movements of the robot.

Figures 6 and 7 show the time employed for both, the client and the server side for the "PC/CORBA" experiment. Most of the time is spent sending images through the Internet connection and waiting the robot accomplishes the required movement.

The whole computer vision process takes about 2.5 seconds of the whole manipulation operation, which means the overall system performance can be improved by optimizing the binarization, segmentation, and features extraction procedures.

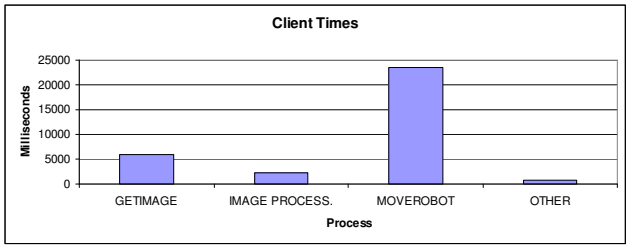

Figure 6. Time employed by the client for "PC/CORBA" experiment.

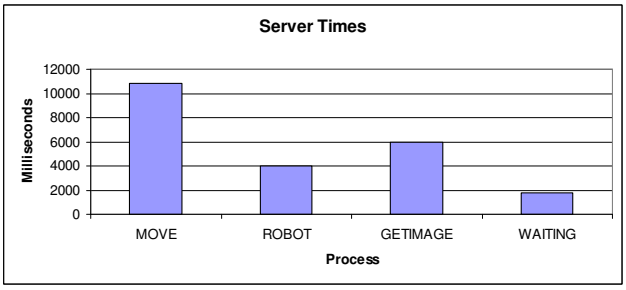

Figure 7. Time employed by the server for "PC/CORBA" experiment.

Thus, in the "FPGA/SNRP" experiment, the image acquisition, its processing, and pose determination are performed by an FPGA, which provides the user information of feature and pose determination through the network using the SNRP architecture. In this situation, as the FPGA capabilities are much bigger, we are using a higher image resolution (i.e. 1024x768) instead of the small images used in the previous case (i.e. 352x288). Moreover, as we want to move the robot in a smoother way, the increments used in the control law (i.e. the visual servoing gaining factor) are decreased. In fact, for this situation the whole visual servoing task is executed in an average of twenty-five loops per experiment instead of the ten loops we used for other cases. The average loop time is 221 milliseconds, taking into account that the robots take 200 milliseconds to perform any movement. Comparing the results with previous experiments, we can observe a great improvement in the average loop time.

In the FPGA side (See figure 8) we can observe how the FPGA has wasteful resources. In fact, the FPGA spends more time waiting than processing the image. Moreover, if we compare the time that the FPGA uses to perform the image processing and pose determination (i.e. Image column in Figure 8) with the same time in the other experiment, we can observe that using a FPGA is about ten times faster than using a personal computer. So, thanks to this improvement, the client can compute the object centroid faster.

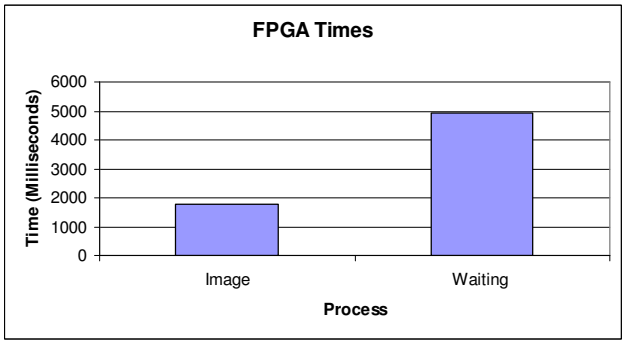

Figure 8. Global FPGA time for the "FPGA/SNRP" experiment.

Moreover, the implementation of the SNRP protocol provides also a very big improvement. In fact, the average time invested to obtain the object features from the network is about 166 ms in the "PC/CORBA" experiment. But, for the "FPGA/SNRP" experiment, the object features are obtained from the SNRP FPGA vision system in an average of 10 ms, varying the responses from 3 ms to 15 ms.

IV. Discussion

In this paper we have presented recent progress in the UJI robotics manufacturing cell, which permits to perform remote programming experiments over a real robotic platform. This kind of industrial tele-laboratory uses a distributed network architecture called SNRP (Simple Network Robot Protocol) that simplifies a lot the interaction between the different components of the system (i.e. robots, cameras, experiments, etc.). Results show that the SNRP protocol enhances very much the performance of the whole system (over 10 times faster).

Moreover, a FPGA has been used to implement a real-time vision system that provides SNRP services to the network. The FPGA takes as input the images from a webcam (camera-in-hand). By having such an improvement in the computer vision module we get the opportunity to program fast and reliable visual servoing controls over this industrial tele-laboratory.

The paper has presented a real remote programming experiment which demonstrates that this technique is very appropriate for education, research and even industrial applications. In fact, the remote visual servoing experiment has been selected to demonstrate that remote experiments

could be used even in those situations where time response is crucial for performance.

The next research goal will focus on the enhancement of the low-level implementation of the SNRP protocol, which allows a given experiment to get access to every device belonging to a tele-laboratory. In fact, a new version of this protocol (SNRP v2) is already in progress that includes a SNRP device browser in order to dynamically present a 3D virtual environment for real SNRP devices. Thus, this interface opens the door to the design of semantic web services, knowledge storing, learning, etc.

Moreover, future work will pursue the development of more sophisticated visual servoing loops using external cameras, pan/tilt and also stereo cameras. FPGA systems will be considered to implement such a vision devices.

Finally, the experiments on visual servoing which we have carried out at the moment are based on position. In the near future, we will be able to allow users to control the speed of robot movement, given that the Motoman robot that we provide in the industrial tele-laboratory permits this kind of control. With this new configuration it will be possible to make experiments with more sophisticated remote visual servoing techniques.

REFERENCES

[1] A. J. Lacey, N. A. Thacker, S. "Crossley, and R. B. Yates, "A Multi-Stage Approach to the Dense Estimation of Disparity from Stereo SEM Images", Image and Vision Computing, vol. 16, pp. 373-383, 1998.

[2] S. Fathnam and G. Slavenburg, "Processing the New World of Interactive Media: The Trimedia VLIW CPU Architecture," IEEE Signal Processing Magazine, vol. 15, pp. 108-117, 1998.

[3] S. Purcell, "The Impact of Mpact 2: The Mpact 2 VLIW Media Processor Improves Multimedia Performance in PCs", IEEE Signal Processing Magazine, vol. 15, pp. 102-107, 1998.

[4] S. M. Bhandarkar and H. R. Arabnia, "Parallel Computer Vision on a Reconfigurable Multiprocessor Network," IEEE Transactions on Parallel and Distributed Systems, vol. 8, pp. 292-309, 1997.

[5] R.C. Cofer, B. Harding. Rapid System Prototyping with FPGAs: Accelerating the Design Process, Newnes, 2006.

[6] W. Mangione-Smith and B. Hutchings. "Configurable computing: The road ahead". In R. Hartenstein and V. Prasanna, editors, Reconfigurable Architectures: High Performance by Configware, pp. 81-96, IT Press, 1997.

[7] G. Spivey, S.S. Bhattacharyya and K. Nakajima. "A component architecture for FPGA-based, DSP System Design", IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP'02), 2002.

[8] P. Barneerje et al. "MATCH: A MATLAB compiler for configurable computing systems", Technical Report, Center for Parallel and Distributed Computing, Nosthwestern University, CPDC-TR-9908-013, 1999.

[9] P. Bellows an B. Hutchings. "JHDL – an HDL for Reconfigurable Systems", IEEE Symposium on FPGA's for Custom Computing Machines, pp. 175-184, 1998.

[10] Celoxica. PixelStreams. http://www.celoxica.com/techlib/files/CEL-W0602151ECF-318.pdf

[11] R. Marin, P. J. Sanz, P. Nebot, and R. Wirz. "A Multimodal Interface to Control a Robot Arm via Web: A Case Study on Remote Programming". IEEE Transactions on Industrial Electronics (Special Section on Human-Robot Interface), vol. 52, no. 6, pp. 1506–1520, 2005.

[12] R. Marín, P.J. Sanz., A.P. del Pobil, "The UJI Online Robot: An Education and Training Experience". Autonomous Robots, vol 15, Number 3, 2003.

[13] G T McKee, D I Baker, P S Schenker : "Network robotics: Dynamic reconfigurable architectures", Proceedings SPIE Intelligent Robots and Computer Vision XXII: Algorithms, Techniques and Active Vision (2004)

[14] G T McKee, D I Baker, P S Schenker : "Robot Scpaces, Module Networks and Distributed Robot Architectures", Proceedings of the IROS 2004 Workshop on Networked Robotics:issues, architectures and applications, Sendai, Japan (2004).

[15] B. K. Kim et al., "Web Services Based Robot Control Platform for Ubiquitous Functions" In Proc. of the IEEE Int. Conf. On Robotics and Automation (ICRA). Barcelona, Spain, April 2005.

[16] D. Lee and M. W. Spong, "Bilateral Teleoperation of Multiple Cooperative Robots over Delayed Communication Networks: Theory" In Proc. of the IEEE Int. Conf. On Robotics and Automation (ICRA). Barcelona, Spain, April 2005.

[17] R. Wirz, R. Marín, E. S. Quintana-Orti. "Distributed System for Remote Programming of Multiple Network Robots: System Performance & Parallization Issues", CEDI 2005 Ist Spanish Conference on Computer Science CEDI 2005, Workshop on Parallelization (JP 2005), Granada, September, 2005.

[18] R. Marín and P. Sanz. "Grasping Determination Experiments within the UJI Robotics Telelab", Journal of Robotic Systems, Internet and Online Robots for Telemanipulation Special Issue (Part 2), vol 22, Number 4, 2005.

[19] P. X. Liu, M. Q. H. Meng, S. X. Yang, "Data Communications for Internet Robots", Autonomous Robots, vol 15, 2003.

[20] P. X. Liu, M. Q. H. Meng, P. R. Liu, S. X. Yang. "An End-to-End Transmission Architecture for the Remote Control of Robots Over IP Networks", IEEE Transactions on Mechatronics, vol 10, Number 5, 2005.

[21] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for real time streams in the Internet", in Proc. IEEE Infocom, Mar. 1999, pp. 1337-1345.

[22] R. T. Fielding and R. N. Taylor, "Principled Design of the Modern Web Architecture", in Proc. ICSE 2000, pp 407-415, 2000.

[23] G. León, J.M. Claver, G. Fabregat. "Optimizing area on the generation of specific circuits on FPGAs for SIMD applications", Int. Workshop on Applied Reconfigurable Computing, pp. 160-167, 2005.