

An Open Architecture for Sensory Feedback Control of a Dual-Arm Industrial Robotic Cell

*Contributed paper for the Industrial Robot Special Issue on Control and Programming
Editor: Clive Loughlin*

Abstract—An open architecture for real-time sensory feedback control of an industrial robotic cell is presented in this paper. The experimental setup is composed of two industrial robot manipulators equipped with force/torque sensors and pneumatic grippers, a vision system and a belt conveyor. The original industrial robot controllers have been replaced by a single PC with software running under a real-time variant of the Linux operative system. The new control architecture allows advanced control schemes to be developed and tested for the single robots and for the dual-arm robotic cell, including force control and visual servoing tasks. An advanced user interface and a simulation environment have been developed, which permit fast, safe and reliable prototyping of planning and control algorithms.

Index Terms—Real-time operating systems, sensory feedback control, industrial robots, cooperative manipulators, rapid prototyping.

I. INTRODUCTION

The development of advanced control algorithms for industrial robots requires open control architectures where software modules can be modified and exteroceptive sensors like force/torque sensors and vision systems can be easily integrated. As a matter of fact, rapid prototyping, i.e., the capability of designing and testing new control and supervision algorithms in short time and with limited costs, is becoming a fundamental issue in industrial robotic applications. Hence, the availability of flexible and highly reconfigurable robotic setups is of the utmost importance.

Various open control architectures for industrial robots have already been developed by robot and control manufacturers as well as in research labs (see, e.g., [1], [2]). Notice that the “degree of openness” in a robot controller may vary from one system to the next. Usually some components of the system (e.g., the power system, the low level control) are proprietary and cannot be modified by the user, others may be considered open (e.g., the communication interface hardware, the higher level control), i.e., are based on standard hardware and software with open interface specifications.

Most of the existing open robot architectures are based on a standard PC hardware and a standard operating system. In fact, a PC-based controller can more easily integrate many commercially available add-on peripherals such as mass storage devices, ethernet card and other I/O devices. Moreover, standard software development tools (e.g., Visual C++, Visual Basic, Delphi, etc.) can be utilized.

An important issue of control software architectures deals with real-time operating systems. In recent years the real-time variants of the Linux operating system are becoming widely adopted, especially in research labs [3], [4]. This choice is motivated by the fact that Linux operating system and its two main real-time variants (RT-Linux [5] and RTAI-Linux [6]) are distributed under the GNU public licence [7], so that they are freely available and configurable to meet desired requirements. Moreover, all the source codes, as well as powerful development tools and detailed documentation, are available.

In this paper, an environment for open real-time control of the industrial cell of PRISMA Lab [8], based on two robots Comau SMART-3 S, is presented. The control architecture is based on the open version of the industrial Comau C3G 9000 controller [9], produced by TecnoSpazio SpA, which allows controlling the robot using a standard PC working with MS-DOS operating system. In the new open controller, named RePLiCS, the software running on the PC was completely replaced by a real-time control environment based on RTAI-Linux operating system [11].

RePLiCS allows advanced control schemes to be designed and tested, including force control and visual servoing. An advanced user interface and a simulation environment have been also developed, which permit fast, safe and reliable prototyping of planning and control algorithms.

A noticeable feature of RePLiCS, which is an enhancement of the existing industrial multi-robot controllers, is that it allows not only the time synchronization of the sequence of operations executed by each robot, but also real cooperation between the robots. Namely, two kinds of cooperation are possible, *loose* cooperation and *tight* cooperation. Loose cooperation means that the robots do not physically interact during the task execution, e.g., one robot moves a workpiece while the other perform laser welding process on it. Tight cooperation means that the robots physically interact through a common manipulated object, e.g. two or more robots transport the same heavy or large object.

Moreover, an enhanced visual system named VESPRO has been developed and interfaced with RePLiCS, which is able to manage a multi-camera visual system [12]. This system is oriented to perform visual pose (position and orientation) estimation of moving object with known geometry.

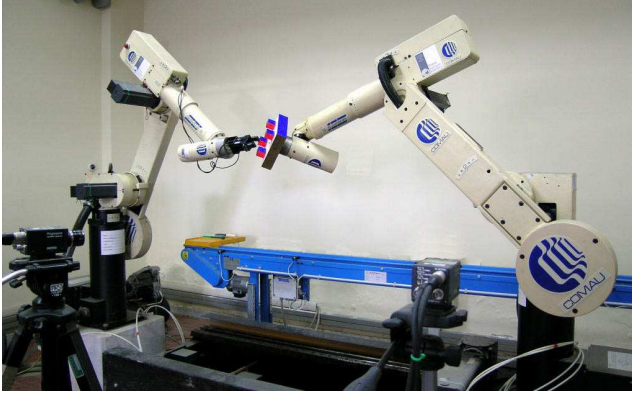


Fig. 1. The dual-arm industrial robotic cell

II. THE EXPERIMENTAL SETUP

The setup in the PRISMA Lab consists of two industrial robots Comau SMART-3 S (see Fig. 1). Each robot manipulator has a six-revolute-joint anthropomorphic geometry with nonnull shoulder and elbow offsets and non-spherical wrist. One manipulator is mounted on a sliding track which provides an additional degree of mobility. The joints are actuated by brushless motors via gear trains; shaft absolute resolvers provide motor position measurements.

Each robot is controlled by the C3G 9000 control unit which has a VME-based architecture with 2 processing boards (Servo CPU and Robot CPU) both based on a Motorola 68020/68882. The Servo CPU has an additional DSP and is in charge of trajectory generation, direct and inverse kinematics, micro-interpolation of the joint references and joint position servo control; independent joint control is adopted where the individual servos are implemented as standard PID controllers. The Robot CPU is responsible for the man-machine interface and for interpreting the user's programs written in the PDL 2 programming language. This board includes also a shared memory area accessible by the other boards connected to the VME bus.

Upon request, COMAU supplies the proprietary controller unit with a BIT3 bus adapter board, which allows the connection of the VME bus of the C3G 9000 unit to the ISA bus of a standard PC with MS-DOS operating system, so that the PC and C3G controller communicate via the shared memory available in the Robot CPU. In this way the PC can be used to implement control algorithms [9], and time synchronization is achieved by means of a flag set by the C3G and read by the PC. A closed proprietary C library (PCC3Link produced by Technospazio SpA) is available to perform communication tasks, e.g., reading shaft motor positions and/or writing motor reference currents or reference positions from/to the shared memory.

A schematic of the open control architecture for one robot is sketched in Fig. 2.

Seven different operating modes are available in the C3G control unit, allowing the PC to interact with the original controller both at trajectory generation level and at joint control level. The most useful operating modes are

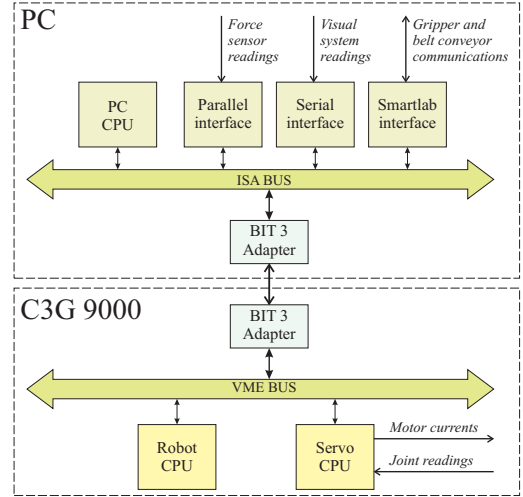


Fig. 2. Schematic of the C3G open control architecture for one robot

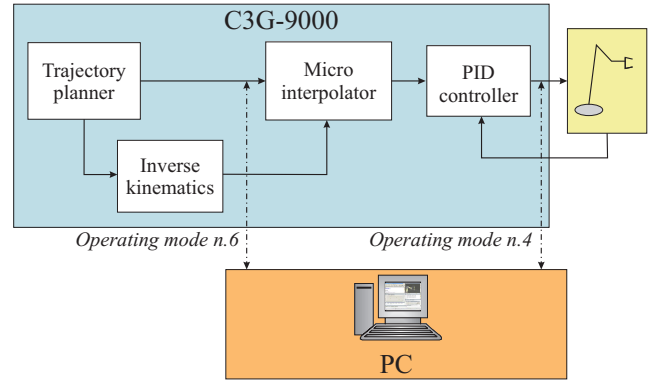


Fig. 3. The two operating modes of the C3G open controller

number 4 and number 6, that are conceptually represented in Fig. 3.

In Operating Mode 4, the joint position servos managed by the C3G are opened and the PC is in charge of acquiring data from the resolvers, computing the control algorithm and passing the references to the current servos at 1 ms sampling time. Hence, the C3G controller is only used as an interface between the PC and the resolvers and the brushless motors of the robot.

In Operating Mode 6, the PC computes the joint position references for the micro-interpolator of the Servo CPU of the C3G controller at 2 ms sampling time. Therefore, the PC is in charge of trajectory planning and kinematic inversion, while the native joint position servos of the C3G controller are used to move the robot.

The sensing capabilities of the robotic cell are completed by force/torque sensors and a stereo vision system. In detail, a six-axis force/torque sensor ATI FT30-100 with force range of ± 130 N and torque range of ± 10 N-m can be mounted at either arm's wrist. The sensor is connected to the PC by a parallel interface board which provides readings of six components of generalized force at 1 ms. The vision system is composed of a PC with Pentium

IV 1.7GHz processor and Windows XP operating system, equipped with two MATROX Genesis boards and two SONY 8500CE B/W cameras. The MATROX boards are used as frame grabbers as well as for a partial image processing (e.g., windows extraction from the image), while the PC host is in charge of executing vision-based algorithms and guarantees communication with the PC performing robot control via a standard serial and/or parallel connection.

Each robot can be also equipped with a pneumatic gripper with two parallel jaws. The completion of the open and close operations are detected using Hall-effect sensors. The grippers can be directly commanded by the C3G industrial controller using PDL2 instructions, or by the PC through a SMARTLAB ISA interface board. Through this board it is also possible to operate a belt conveyor installed from one side of the cell, parallel to the two robots, which can be driven in the two directions and is equipped with an optical presence sensor at the two sides and a magnetic sensor that can be used for the identification of particular pieces carried by the belt.

Notice that, in Fig. 2, the complete control architecture for one robot, including force/torque sensor, vision, gripper and belt conveyor is described. In the case of the dual-arm robotic system, a single PC is used, controlling the whole cell. Hence, on the ISA bus of the PC are connected two different BIT3 boards, which allow connection with the two separate VME buses of the C3G 9000 controllers, two parallel interfaces for the two force/torque sensors, one serial interface for the vision system and one SMARTLAB interface for the two grippers and the belt conveyor.

The above experimental setup, thanks to the open control architecture, allows advanced control algorithms to be designed and tested, both for the single robot and for the two cooperative robots. In fact, a large amount of experimental tests have been carried out during the last years, which include kinematic control [10], resolved acceleration control [13], force/position control [14], dual-arm manipulators control [15], visual servoing and tracking [16].

It should be pointed out, however, that the above system suffers from many problems due, essentially, to the limits of MS-DOS operating system. The main drawbacks are:

- the RAM available for programs and data storage is limited to 512 Kbyte;
- the integration of advanced sensing capabilities, like vision, is not easy;
- the coordination between the two robots is difficult because a tight time synchronization between the two control units cannot be realized;
- the system is not multitasking and it is not possible to realize a user interface to visualize data or input commands while the controller is running;
- the robotic cell cannot be connected to Internet.

In view of the above limitations, a new real-time control system has been designed for the cooperative cell available at PRISMA Lab. The aim was the creation of a flexible experimental set-up that allows using the robots in different

tasks involving one single robot, time synchronization of the two robots, the grippers and the conveyor belt, as well as loose and tight cooperation of the dual-arm system. Further requirements were the possibility of executing interaction tasks or tight cooperation using force control as well as visual servoing tasks. Last but not least, a crucial point was the development of an advanced user interface for monitoring the system during the experiment as well as for rapid prototyping of task planning and control algorithms.

III. RePLiCS

The new control software, named RePLiCS (REal-time PrismaLab Linux Control System), was developed using RTAI-Linux, which is a real-time variant of the Linux operating system.

RePLiCS can be structured into a real-time module, which is a driver for the kernel of RTAI-Linux, and a set of non real-time applications that provide a user interface for the real-time module. The real-time module is periodically activated by an external interrupt signal generated by the C3G 9000 controller. Suitable communication channels exist between the real-time module and the user applications.

A. RePLiCS real-time module

The real-time module of RePLiCS implements all the real-time functions required for the control of the robotic cell. All those functions are collected in a API (Application Programming Interface) software library written in the C language. These functions can be grouped in:

- communication with the C3G-9000 controllers,
- force sensors reading,
- synchronization for cooperative control
- safety checks,
- robot kinematics,
- robot control,
- trajectory planning,
- serial and parallel communication,
- data storage,
- I/O functions (file or console).

The functions for the communication with the C3G-9000 controllers implement the drive on/off commands and have access to the shared memory area of the Robot CPUs to read the joint positions and write the current set-points or the position set-points, depending on the selected operating mode. For example, in Operating Mode 4, the C3G executes the following operations:

- set the synchronization flag `IntActive` (interrupt signal),
- write the values of the motor shaft angular positions,
- read the desired values for the motor current set-points;

while the PC is in charge of the following operations:

- reset the synchronization flag `IntActive`,
- read the values of the motor shaft angular positions,
- compute the current set-points for the next time interval,

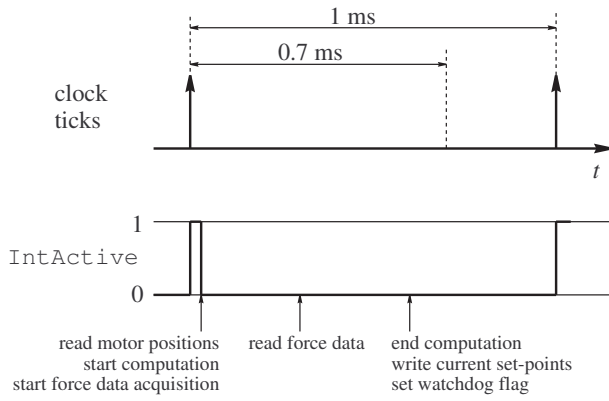


Fig. 4. Timing diagram of the real-time control loop

- write the desired values for the motor current set-points,
- set the watchdog flag.

To fulfill the real-time constraint at 1 ms sampling time, the PC must execute the above steps within 0.7 ms. In the remaining 0.3 ms time window, called “forbidden window”, the PC must not access to the shared memory. The sequence outlined above is depicted in Fig. 4. The watchdog flag implements a watchdog timer mechanism, which is aimed at preventing harmful consequences of unpredictable events, e.g., when the PC crashes and thus is no longer able to write a correct current set-point to be actuated. If this situation happens, the system reaches an alarm state, the robot drives are switched off and brakes are activated.

To obtain the force measurements, the PC has to send a data acquisition request, which starts the data A/D conversion on the sensor conditioning electronics; after a time lapse of about 0.25 ms, the six components of the force and torque are available in a memory buffer. Since the conversion is executed on remote hardware, during the conversion time interval, the PC can continue its elaboration. Hence, to avoid simply waiting for the end of conversion, it is convenient to ask for the start of conversion at the beginning of the control cycle, perform all the computations not requiring force measurements, e.g. kinematics, dynamic model compensation, then read the force data and, finally, complete the control algorithm. The timing of these operations is outlined in Fig. 4.

To implement cooperative control of the two robots, the PC should write the positions or the current set-points for both robots at the same time, on the basis of the motor angular positions read at the same time. This ideal behavior, however, is difficult to achieve because the two C3G controllers have two separate clocks. Hence the temporal windows for reading and writing on the shared memory, as well as the forbidden windows, are usually not aligned. Moreover, since the two clock frequencies are slightly different, a time drift between the two trains of clock ticks is experienced. To solve this problem the synchronization module of RePLiCS, in a preliminary phase, checks the

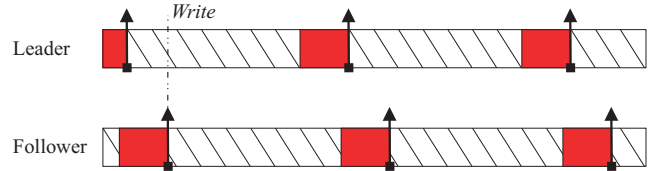


Fig. 5. Timing diagram of the leader and follower clocks

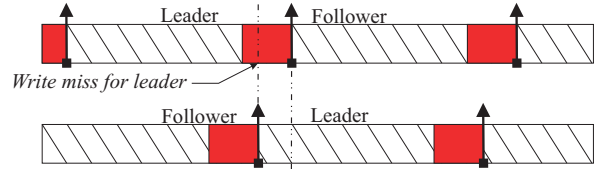


Fig. 6. Exchange of role between leader and follower

state of the IntActive flags and of the watchdog flags of the two C3G controllers and defines one robot as *leader* and the other as *follower*. The follower is the robot whose clock tick does not arrives during the forbidden window of the other robot (see Fig. 5, where the forbidden windows for the two clock trains are in red color).

At this point, the control loop can start and the following operations are executed:

- write the desired values for the motor current set-points of both the robots,
- read the values of the motor shaft angular positions of both the robots,
- set the watchdog flags,
- compute the current set-points for the next time interval.

Due to the time shift between the two clocks, a situation like that represented in Fig. 6 may occur, where the clock tick of the follower arrives during the forbidden window of the leader. In this case, the role of the leader and of the follower are exchanged, so that the computed current set-points are written when the clock tick of the new leader arrives. This implies that the previous leader loses the reference only for one time interval, which is irrelevant. Particular solutions must be adopted for handling singular cases, e.g., when the clocks of the leader and the follower are almost aligned.

Besides the watchdog timer, several safety checks can be carried out to monitor the system functioning and prevent damage, for example, set and check joint limits, maximum joint velocities, maximum instantaneous currents, maximum sustained currents, maximum forces and torques.

Two different decisions can be taken in case one of the above checks fails, namely immediately stop the robot by invoking a special emergency routine, or set an error flag to exit from the main control loop and switch off the drives. The former approach is faster and thus safer, but requires a complete reboot of the system. The latter approach implies a delay of one sampling period, but leads the system to a less critical state, which does not require a complete reboot.

The best policy is to decide on the particular safety check which failed, e.g., if one of the motor currents exceeded the maximum sustained value, an immediate stop is not necessary; on the contrary, if the force is over the maximum allowed value, an immediate stop is advisable.

The functions used for robot kinematics allow the computation of the direct kinematics of the robots and their Jacobians. The inverse kinematics is computed by means of CLIK algorithms [15]. The damped least-squares inverse Jacobian is adopted to cope with singularity problems. Several utility functions are available, e.g., for the unit conversion of the joint variables (degrees, radians, Bit resolvers), for the coordinate conversion between different frames, for the representation of the orientation (Euler angles, angle/axis, quaternion). The kinematic library is still under development to include modules for redundancy resolution and inverse kinematics for dual-arm systems.

The robot control functions implement decentralized joint control as well as centralized control, e.g., inverse dynamics or resolved acceleration in the task space [13]. Interaction control strategies [14] based on force measurements are also implemented. Moreover, software modules that realize loose and tight cooperation [17] are available. New control schemes can be easily programmed by modifying the control module of a template program file (written using the standard C language), which includes the API library of RePLiCS.

As for trajectory planning, a set of functions are available for the point-to-point motion, both in the joint and in the task space (along straight lines for the position) which use time laws with trapezoidal velocity profile. A function which allows generating a path (in the joint space or in the task space) through assigned via points is also implemented. Special functions have been realized to achieve synchronization of the two robots at trajectory planning level, and generate smooth trajectories when the target is not known in advance (e.g., in visual servoing applications).

The serial and parallel communications allow the controller to communicate with an external device (e.g., the PC used for the vision system) using the standard serial and parallel port. A set of functions have also been developed to command the two grippers and the belt conveyor through the SMARTLAB interface board.

The storage of the time history of significant variables in a given experiment is managed by suitable functions of the API library. It should be pointed out that, because of the real-time constraints, the values are saved in the RAM memory of the PC with an assigned sample time and are saved on the hard disk only at the end of the experiment. This procedure can be guided by using suitable facilities of the user applications.

Finally, the real-time module of RePLiCS includes also I/O functions on file or console. These functions cannot be executed while the robot control is active, because they may cause a watchdog alarm. A special monitor application of RePLiCS suspends these instructions and allows their

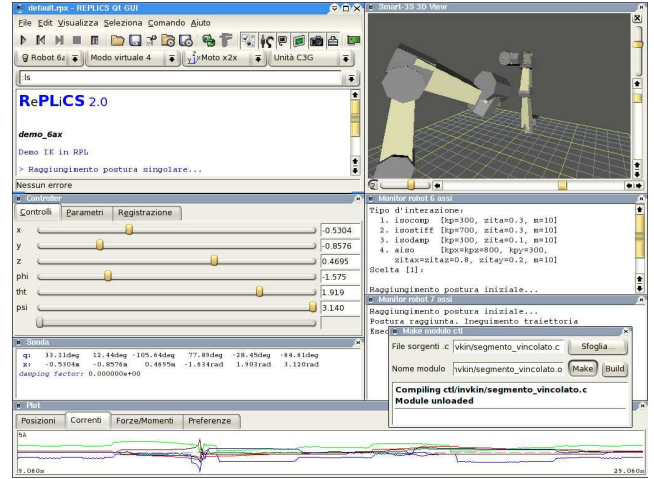


Fig. 7. A screen-shot of all the RePLiCS GUI



Fig. 8. A screen-shot of the GUI of the main user application

execution only in the absence of real-time constraints.

All the real-time functions are compiled in a kernel module and dynamically linked to the real-time kernel of the operating system. Since the user needs to interact with the robot, some communication channels between the kernel space and the user space are used.

From the user space it is possible, e.g., to send the drive on/off command, to change the joint limits and the other safety checks, to open or close the grippers. The real-time RePLiCS module can receive information about the desired joint or end-effector set-points and send back information about the current value of the robot internal variables (positions, velocities, motor currents, contact force). In this manner, it is possible to move the robots by a virtual teach-pendant or to visualize the internal variables on the screen while the robots are moving.

B. RePLiCS user applications

The applications in the user space are essentially aimed at helping the human user to communicate with the dual-arm robotic cell through a Graphical User Interface (GUI). In Fig. 7 all the GUI windows are collected in the same graphical page.

The window on the top-left of Fig. 7, which is also reported in Fig.8, is the GUI of the main user application

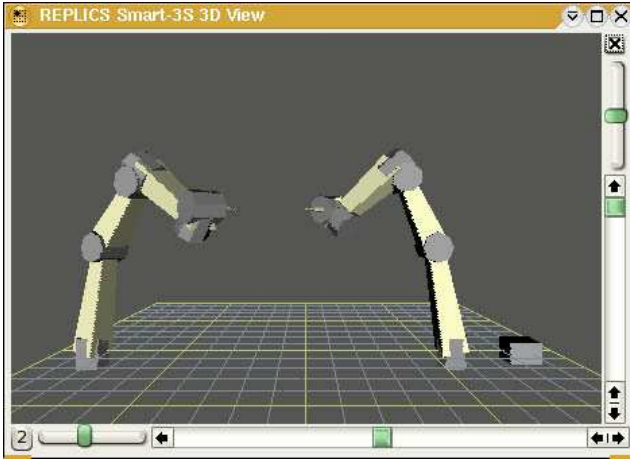


Fig. 9. 3D representation of the dual-arm system

of RePLiCS, which allows performing the most important operations on the system. In particular, by using the menu bar or the toolbar, it is possible: to select one of both the robots, to select the operating mode (4 or 6), to send drive on/off commands, to select the type of motion (joint space or task space) and to select the real or the virtual mode of operation for RePLiCS. This latter feature is of the utmost importance, because it allows testing all the functionalities of the controller on a simulation environment which respects the real-time constraint and includes the C3G 9000 controllers, the robot dynamics and the interaction with a virtual environment. Hence, the whole control prototyping process can be developed off-line in a very fast, safe and reliable way; moreover, the same code developed in the virtual mode can be executed in the real mode without any modification, also using the real measurements of the exteroceptive sensors, if connected.

From the main window, it is also possible to set the joint limits, the maximum currents and other safety checks, to select a point-to-point motion in the joint space or in the task space, to set the parameters of the velocity profile time law, to start the selected motion, to pause the motion before completion and resume it. For task space motions, the GUI allows selecting the parameters of the CLIK algorithm that computes the corresponding joint motions. All the parameters can be set by using the window Controller (on the middle-left of Fig. 7). This window implements also a virtual teach pendant to move the robots in the joint space or in the task space; moreover, it allows selecting the variables to be recorded in the RAM memory of the PC. These variables are visualized in the Plot window (the window on the bottom of Fig. 7) during the execution of the task and can be stored in the hard disk at the end of the motion.

A 3D graphical representation of the dual robot system is available, which allows changing dynamically the point of view and zooming in and out; the graphical window is continuously updated during task execution (see Fig. 9).

From the main window it is also possible to compile

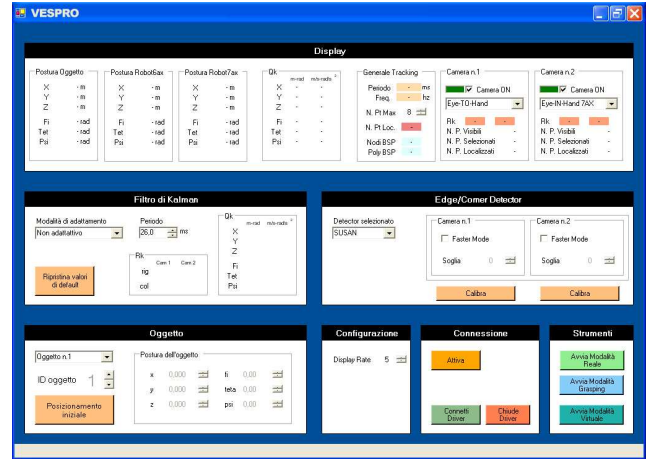


Fig. 10. A screen-shot of the GUI of VESPRO

and to execute the control modules (linked to the real-time kernel) written by the user. The console input/output of the control modules is realized through Monitor windows (on the middle-right of Fig. 7).

In order to facilitate the interaction of the user with the robotic cell, a simple programming language named RPL (RePLiCS Programming Language) has been developed, and an RPL interpreter has been realized. The RPL instructions can be directly input through the console of the RePLiCS main window (see Fig. 8), or can be grouped in script files that are executed as batch programs. These instructions also allow programming synchronized tasks for the two arms of the cell, the grippers and the belt conveyor. Further details on the RPL language and on RePLiCS software can be found in [18].

C. VESPRO

VESPRO (Visual Enhanced System for PrismaLab Robot Operation) is a software environment able to manage a multi-camera visual system, oriented to perform visual pose estimation of moving objects with known geometry. It is structured into a low level driver (entirely written in C and in C++ languages) and a GUI for Windows NT operating system (Fig. 10).

This visual system may be used to perform both position based and image based visual servoing tests. For brevity, only position based visual servoing is considered in this paper. This approach requires the estimation of the pose of a target object with respect to a reference frame by using a vision system; the estimated pose is then used by a pose controller. Hence, the two main operations to be performed are *pose control* and *pose estimation*.

Notice that the pose estimation is a computationally demanding task, because it requires processing of the measurements of some geometric features extracted from the images of one or more cameras. Hence, the sampling time of the pose estimation algorithm is usually higher than the sampling time of the pose control loop. In the best case, the pose estimation can be performed at the camera frame

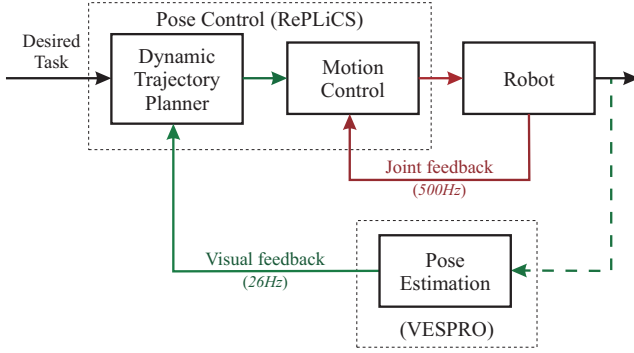


Fig. 11. Position based visual servoing scheme

rate (between 25 Hz and 60 Hz).

A schematic representing the position based visual servoing algorithm implemented in the experimental setup is represented in Fig. 11.

The pose control is performed through an inner-outer control loop. The inner loop, running at 2 ms sampling time, implements motion control (independent joint control or any kind of joint space or task space control). The block named dynamic trajectory planner computes, depending on the desired task, the trajectory for the end-effector on the basis of the current object pose. The output of this block is available at 500 Hz frequency, while the input is updated only at 26 Hz frequency, corresponding to the frame rate of the employed cameras. Obviously, RePLiCS implements the pose control. The pose estimation algorithm implemented by VESPRO, which running under Windows NT operating system on a dedicated PC, provides the measurements of the target object pose at 26 Hz frequency.

The vision system can be based on eye in hand cameras, i.e., one or two cameras mounted on the robot end-effector, or a system of multiple fixed cameras. Hybrid configurations, including both eye in hand and fixed cameras may represent a good solution, which integrates the advantages of both kinds of configuration.

The use of a multiple camera system requires the adoption of intelligent and efficient strategies for the management of highly redundant information (a large number of image object features from multiple points of view). This task has to be realized with hard time constraints and thus the extraction and interpretation of all the available visual information is not possible. To solve this problem, an efficient technique has been developed which is able to improve the accuracy and robustness of the visual system by exploiting a minimal set of significant information suitably selected from the initial redundant set. This technique, implemented in VESPRO, is described in [16].

The software architecture of VESPRO is represented in Fig. 12. It can be recognized that the driver is composed by a collection of software modules that can be activated and configured by a Module Manager which translates the high level commands from the GUI.

The Hardware Interface module allows interfacing with the two MATROX Genesis boards, used as frame grabbers

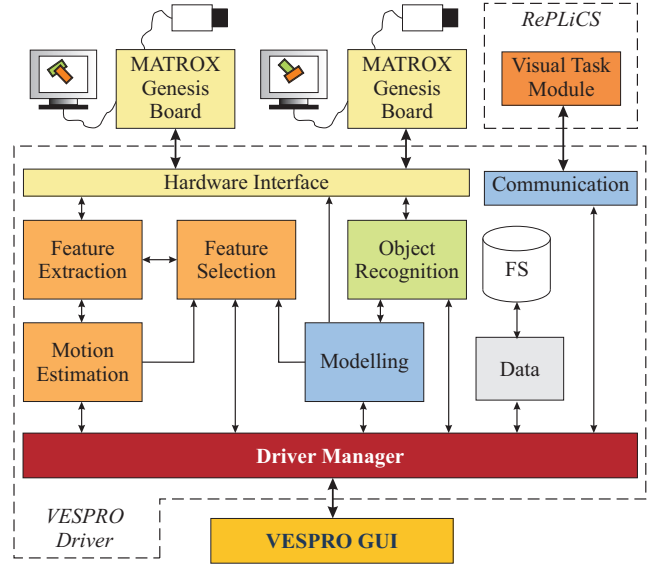


Fig. 12. Schematic of the VESPRO software architecture

and image pre-processing. In particular, the low level functions for the control of the boards are organized in high level functions that allows, for example, the image acquisition, image windows extraction, image preprocessing as binarization, convolution, etc. This is the unique VESPRO module that is hardware dependent (i.e., depends on the boards used as frame grabbers). Moreover, it includes a software library for the characterization of the cameras, the compensation of the main geometrical distortion phenomena. Notice that VESPRO allows managing both fixed cameras and eye-in-hand cameras.

The Modelling modules is in charge of managing a library of CAD model of objects and building BSP-tree presentations of a single objects or a set of objects.

IV. CONCLUSION

In this work an open architecture for real-time sensory feedback control of an industrial robotic cell based on RTAI-Linux has been described. The environment allows fast prototyping of advanced control algorithms, both for a single robot and for two cooperative robots. Force/torque sensors and vision sensors can be used too. Future work will be devoted to further improve the features of the current system, e.g., build a library of advanced control modules, enrich the RPL language of more powerful instructions, develop Internet tools for programming and operating the robotic cell from a remote location.

ACKNOWLEDGMENTS

The authors wish to thank Nello Grimaldi for the significant support to the development of RePLiCS. The contribution of Fabrizio Caccavale on the dual-arm robotic cell is gratefully acknowledged as well.

REFERENCES

- [1] <http://www.mitsubishielectric.com>
- [2] <http://www.robot.lth.se>

- [3] <http://www.gnu.org>
- [4] A. Macchelli, C. Melchiorri, "Real time control system for industrial robots and control applications based on real time Linux," *Proc. 15th IFAC World Congress*, Barcelona, Spain, July 2002.
- [5] <http://www.rtlinux.org>
- [6] <http://www.rtai.org>
- [7] <http://www.orocos.org>
- [8] <http://www.prisma.unina.it>
- [9] F. Dogliani, G. Magnani, and L. Sciavicco, "An open architecture industrial controller," *Newsl. of IEEE Robotics and Automation Soc.*, vol. 7, no. 3, pp. 19–21, 1993.
- [10] F. Caccavale, S. Chiaverini, B. Siciliano, "Second-order kinematic control of robot manipulators with Jacobian damped least-squares inverse: Theory and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 2, pp. 188–194, 1997.
- [11] F. Caccavale, V. Lippiello, B. Siciliano, L. Villani, "RePLiCS: An environment for open real-time control of a dual-arm industrial robotic cell based on RTAI-Linux," *Proc. 2005 IEEE/RJS International Conference on Intelligent Robots and Systems*, Edmonton, Canada, Aug. 2005.
- [12] V. Lippiello, B. Siciliano, L. Villani, "An experimental setup for visual servoing applications on an industrial robotic cell," *Proc. 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Monterey, CA, July 2005.
- [13] F. Caccavale, C. Natale, B. Siciliano, L. Villani, "Resolved-acceleration control of robot manipulators: A critical review with experiments," *Robotica*, vol. 16, pp. 565–573, 1998.
- [14] S. Chiaverini, B. Siciliano, L. Villani, "A survey of robot interaction control schemes with experimental comparison," *IEEE/ASME Transactions on Mechatronics*, vol. 4, pp. 273–285, 1999.
- [15] F. Caccavale, C. Natale, B. Siciliano, L. Villani, "Achieving a cooperative behaviour in a dual-arm robot system via a modular control structure," *Journal of Robotic Systems*, vol. 18, pp. 691–700, 2001.
- [16] V. Lippiello and L. Villani, "Managing redundant visual measurements for accurate pose tracking," *Robotica*, vol. 21, pp. 511–519, 2003.
- [17] F. Caccavale, L. Villani, "Impedance control of cooperative manipulators," *Machine, Intelligence & Robotic Control*, vol. 2, pp. 51–57, 2000.
- [18] N. Grimaldi, *RePLiCS ver. 2.0*, PRISMA Lab internal report, 2004.