Space Robotics: an Experimental Set-up based on RTAI-Linux

A. Macchelli, C. Melchiorri, R. Carloni, M. Guidetti
DEIS - Dept. of Electronic, Computer Science and Systems
LAR - Lab. of Automation and Robotics
University of Bologna
viale Risorgimento 2, 40136 Bologna, Italy
{amacchelli,cmelchiorri,mguidetti}@deis.unibo.it, rcarloni@thesis.deis.unibo.it

Abstract

In space application, it is of great interest the development of autonomous or semi-autonomous robotic devices that can substitute the astronauts in routine operations in order to free them from repetitive tasks and reduce mission costs. In this work, an experimental setup based on a 6 degrees of freedom (dof) manipulator with a 3 dof gripper designed for a possible application within PaT, the Payload Tutor proposed by ASI (Italian Space Agency), is presented. This system consists of a robotic arm, a vision system, and a gripper. Since the gripper has to interact with free-floating and irregular objects, the vision subsystem provides all the information needed for grasping unknown objects in an optimal way.

1 Introduction

Thanks to a constant research and development activity in the field of robotics and, more in general, of automation, reliable and relatively low-cost machines that can substitute or assist human operators in repetitive or dangerous tasks are now available. In robotics, this is particularly true for applications in structured environment, i.e. when the workspace in which the robot operates is known with great accuracy and precisions. On the other hand, it is obviously of interest the development of applications in unstructured or unknown environments. In this case, control algorithms that can give a sort of "human behavior" to the machines are needed: in other words, the machines should be characterized by a functional autonomy, i.e. they should be able to modify their behavior on the basis of information acquired in real-time from the environment.
It is of great interest to achieve these performances in space applications where autonomous or semi-autonomous robotic devices can substitute the astronauts in routine operations in order to free them from repetitive tasks and reduce mission costs.

In this work, an experimental setup based on a 6 degrees of freedom (dof) manipulator with a 3 dof gripper designed for a possible application within PaT, the Payload Tutor proposed by ASI (Italian Space Agency), is presented. This system consists of a robotic arm, a vision system, and a gripper. Since the gripper has to interact with free-floating and irregular objects, the vision subsystem provides all the information needed for grasping unknown objects in an optimal way.

The underlying software architecture is based on a real-time variant of the popular desktop operating system Linux. This kind of systems provide noticeable performances that, together with availability of the source codes, powerful development tools and, generally, well-done documentation, could be the starting point for setting up a good development environment. Moreover, these operative systems are distributed under the GNU Public License, so they are freely available and configurable to meet desired requirements. According with this development model, the RT-Linux [1], [2] and RTAI-Linux [3], [4] projects took place, both with the aim of giving Linux the possibility to implement hard real-time applications.

2 The experimental set-up. An overview

The robotic arm is a Comau SMART 3-S robot, a standard industrial 6 dof anthropomorphic manipulator with a non-spherical wrist, equipped with the

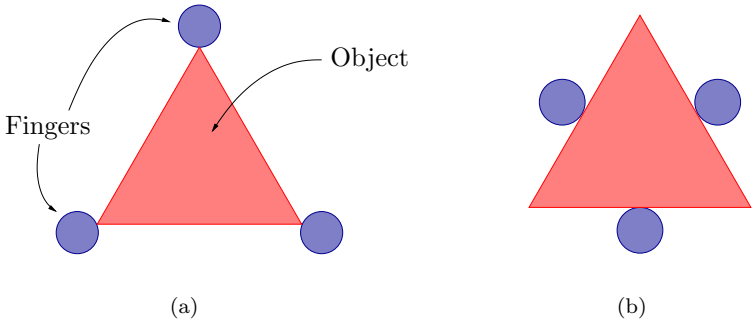


FIGURE 1: Selection of the finger's target points. According to a first order analysis, the configurations presented in (a) and (b) are equivalent, but differ if a second order analysis is carried out. Clearly, the configuration (b) is more *stable*.

standard controller C3G-9000. Each joint is actuated by a DC-brushless motor, and its angular position is measured by a resolver. In our setup, the controller is only used as an interface between the resolvers and drives on the robot and a PC running RTAI-Linux. In each sampling period generated by the controller, the real-time control system running on the PC acquires the data from the encoders, computes the new control input for the actuators and sends their values to the C3G-9000.
This procedure is possible since the C3G-9000 controller is *open*. In fact, its internal (VME) bus is connected to the ISA-PCI bus via a pair of Bit3 boards, one inside the robot controller and the other inside the PC running the control algorithms. A data exchange between PC and C3G-9000 is possible via a shared memory area inside the controller and synchronization can be achieved by an interrupt signal generated by the controller itself. In this configuration, position and velocity loops managed by the C3G- 9000 are opened, and all the safety protections are disabled.

On the robot wrist, both the video-camera and the gripper are installed. The vision system is used to provide visual information about the environment and, in particular, about objects within the workspace of the robot. These informations are needed to track an object moving in the robot workspace, to move the robot in a desired position in order to grasp it with the gripper and to automatically calculate the optimal grasping configuration. At the moment, the vision system consists of a monocular camera, connected to a frame-grabber board installed on the same PC that implements the robot control algorithms. By properly moving the robot arm and, at the same time, acquiring images of the object from different points of view, the vision algorithm can give a good estimate of the distance of the object from the robot wrist. This information is essential to correctly move the robot in order to grasp the object with the gripper.
Moreover, by means of the vision system, the shape of the object is recognized in order to calculate the better grasping points (target points). The object is caught on these points and the contact forces are properly controlled. Generally, the target points are selected on the basis of a kinematic analysis of a first order model: the resulting points do not depend on the shape of the object and on the geometric characteristics of the gripper. In our case, the target points are calculated by means of a kinematic analysis of a second order model that takes into account also the shape of the object and of the gripper's finger: in this manner, the resulting grasping configuration is more stable. An example is presented in Fig. 1. Two grasping configurations that are *equivalent* if evaluated according to considerations based on first order models differ if a second order model is used.

The third component of our setup is the A.S.I. Gripper, [7]. It has three degrees of freedom, and is particularly suited for no-gravity manipulation tasks (i.e. in space applications), since it can interact with free-floating and irregularly shaped objects. Its control algorithms are executed on a custom DSP board (based on the TMS320C32 chip). For this board, a loader and a DSP-monitor have been developed under Linux, together with some drivers for the DSP board. Once the object distance and the target points are calculated, the gripper is correctly positioned in the workspace with respect to the object. Then, the fingers are closed and the object is grasped on the target points. At this point, the DSP board executes the control algorithm in order to assure proper contact forces.

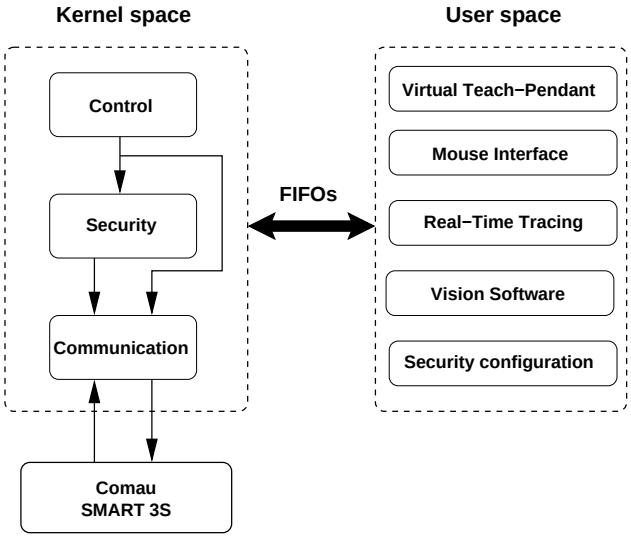


FIGURE 2: The real-time modules and user-space applications. Organization and communications channels.

3 Control of Comau robot with RTAI-Linux

The software developed for the control of the Comau SMART 3-S is divided into two distinct modules: a real-time module, which executes the control algorithms and communicates directly with the plant (robot), and a set of non real-time applications that exploit all the functionalities provided by the real time module. Clearly, some communication mechanisms exist to provide an information exchange between the two parts. The real-time module is periodically activated by an external interrupt signal generated by the C3G-9000 controller.
The adopted real-time Linux variant is RTAI, already successfully used in other robotic applications developed in the past within LAR, [5, 9]. A noticeable experimental activity has been carried out in this context, also comparing the performances of control system based on RTAI-Linux with other commercially available OS, such as QNX, [12].

In order to create a modular and flexible software structure for a quick test of new control algorithms and fast implementation of new robot applications, the code is divided into three main sub-modules, providing:

- communication with robot SMART 3-S;
- security tests;
- implementation of the robot control algorithms.

The communication module is used for reading and writing data on the shared memory area in the C3G-9000 controller. In particular, this module reads the six joint positions and writes six current set-points for the DC drives. Moreover, it implements the drive on/off function. This is the only module that gets access to the shared area inside the C3G-9000 controller. A schematic representation of the software *structure* is presented in Fig. 2.

The security module implements software range delimiters (saturation), limits the joint speeds, checks if some of the joints is blocked and, if necessary, saturates the current set points the maximum allowed value. In particular, the last two tests are needed to prevent drives damages.

The control module implements the control algorithms and, in particular it is responsible for trajectory planning, both in joint and Cartesian space, and for robot regulation. The sub-module that implements the regulation algorithms may change according to the control scheme under development. e.g. decentralized control, multi- variable centralized control, and so on.
Since in most of the application (e.g. with the vision system) the desired trajectory is not known before the execution of a task, we have implemented a trajectory generator using a non- linear filter with constraints on maximum speed and acceleration, [6, 8].

All these real-time functions are compiled in a kernel module and dynamically linked to the real-time ker-

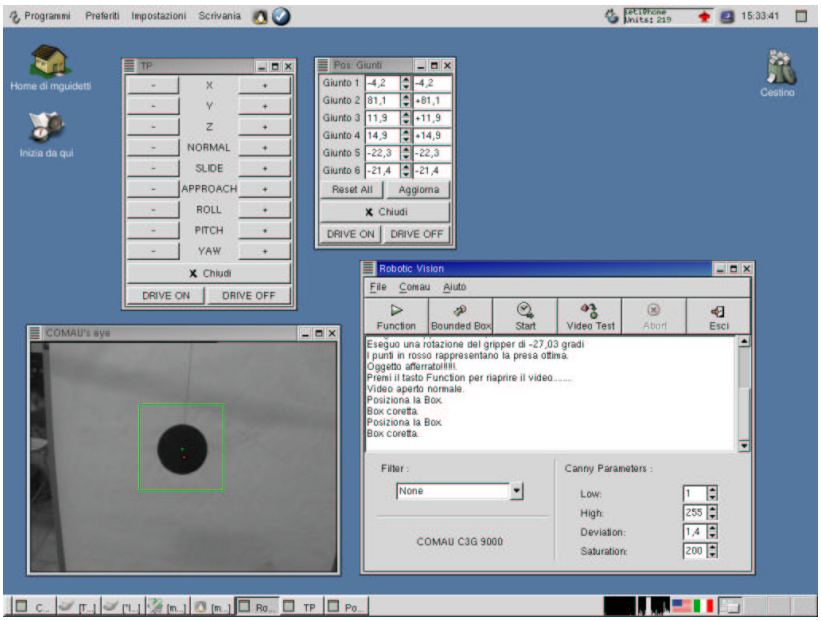


FIGURE 3: Screen-shot of the vision software.

nel of the operating system. Since the user needs to interact with the robot, some communication channels between *kernel space* and *user space* are needed. Each module can exchange data with the user space applications by its own channels. Since in all the situations the amount of data that we send from/to the kernel module is not relevant, we decided to implement all the communication channels using FIFOs. This solution provides robustness and a built-in coordination/synchronization mechanism between sender and receiver.

From the user space it is possible to send the drive on/off command to the communication module of the real time process, and it is possible to change in runtime some parameters in the security module (this is a function that must be disabled when testing new control schemes). Concerning the control module, it can receive commands from user space applications and send back to them information about internal variables of the robot (e.g. joint positions and drive currents). In this manner, it is possible to move the robot with a keyboard, a mouse, a joystick or a *virtual* teach pendant, to interface it with other applications for state-tracing or for movements under vision system control. More details can be found in [9].

4 The vision system

The vision system consists of a monocular videocamera mounted on the robot's wrist. The related software runs as a user-space application and communicates with the real-time (kernel) module by means of a set of commands in order to make the robot to execute specifics tasks/movements.

In Fig. 3, a screen-shot of the vision software with main window `Robotic Vision` is presented. Moreover, the `TP` window is the *Virtual Teach Pendant* mentioned above, while `Pos:Giunti` is another tool that allows the user to move the robot acting on each joint separately. The scene framed by the videocamera is reproduced in the `COMAU's eye` window.

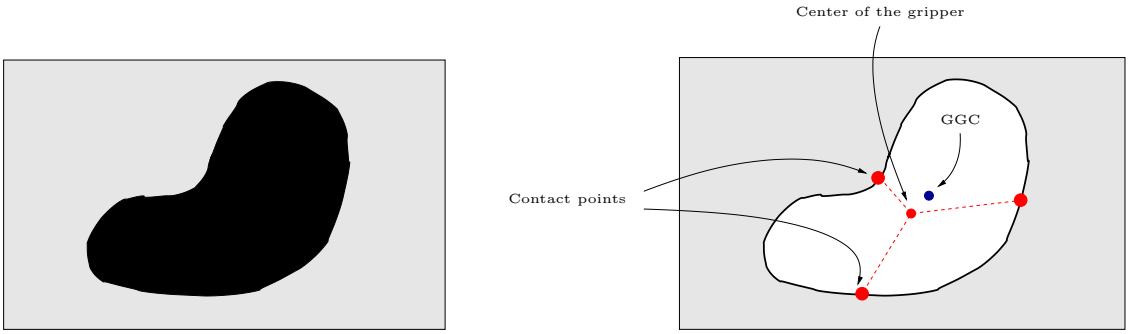
A typical task for this system is to automatically grasp a user selected object within the robot's workspace by using the gripper mounted on the robot's wrist. Moreover, the grasp has to be *optimal* and *stable*. This procedure consists of two main steps:

- the evaluation of the object distance from the robot wrist;
- the determination of the best grasping configuration.

After that, the robot is automatically moved in order to position the gripper such that the object can be cached in the most suitable way. In the following subsection, a detailed description of these two main steps is presented.

4.1 Distance evaluation

The first step is to move the robot in order to frame the object that has to be cached by the gripper with the video camera. Once the object is selected by



(a) Object to be grasped.

(b) Fingers configuration and contact points.

FIGURE 4: Calculation of the best grasping configuration. From the image framed by the video-camera, to the finger contact points.

the operator, its *geometric gravity center* (GGC) is calculated and, by sending proper *pitch* and *yaw* commands to the real-time module that controls the robot, the video-camera optic axis is aligned with the GGC.

Since the vision system is not stereo, an estimation of the actual distance between object and end-effector is calculated by moving the robot along the direction camera-object and taking (at least) two pictures of the object itself. The first picture is taken in the actual position of the robot, while the second one after a negative *approach* is executed. This is necessary since the distance from the object is unknown. From a *comparison* of the same object framed from two different but aligned points of view, a first estimation of the unknown distance is calculated.

Analyzing some experimental results, it has been determined that the best performances can be achieved if the distance between video-camera and object is about of $35cm$. For this reason, performances can be improved by taking (if necessary) more than two pictures of the same object. In particular, if the first estimate provides a distance between 32.5 and $37.5cm$ then this estimation is assumed to be correct. If it is not the case, the robot executes an approach in order to move the camera at a (estimated) distance of $35cm$ from the object and the previous procedure is repeated. The result is assumed to be correct if it belongs to the range $32.5 \div 37.5cm$.

Once the position of the object with respect to the video-camera, and clearly from the gripper, is determined, a first *approach* is executed in order to position the video-camera at $20cm$ from the object. At this point, a procedure that analyzes the object and calculates the *best* contact points in order to catch the object with the gripper is started. More details are in the next subsection.
By now, assume to know how the object has to be grasped. Then, by means of *normal* and *slide* commands, the center of the gripper is aligned with the GGC of the object. Finally, the gripper is rotated and its fingers can grasp the object in the desired contact points. More details in [13].

4.2 Evaluation of the optimal grasping configuration

The only informations the system owns about the object to be grasped are provided by the video-camera. A typical situation is represented in Fig 4(a). The first step is to extract the contour of the object by means of the well-known Canny's algorithm and to calculate its GGC. Since the Canny's algorithm provides only a bitmap describing the object contour, it is necessary to ordinate these points in order to obtain a discrete parametrization of the border of the object. This procedure is called *edge tracking*: more detail about the one we implemented can be found in [14].

Given the border parametrization, it is possible to extract the all the geometric informations that are useful for the determination of the *best* grasping configuration, that is the set of tangent and normal vectors and its curvature at a given point. The vector t_i tangent to the point P_i of the border is assumed to be the line passing from P_i and P_{i+1}, while the normal vector n_i and the curvature r_i are determined by calculating the circumference passing through P_{i-1}, P_i and P_{i+1}.
Starting from this set of geometrical parameters that

describes the border of the object, the algorithm is able to provide the three contact points for which an index describing the *performances* of the corresponding grasp is maximum. Clearly, the resulting contact points have to be compatible with the mechanical characteristics of the A.S.I. gripper. In particular, since the contacts with the grasped object can occur along three intersecting lines equally spaced of 120°, the resulting configuration has to respect this mechanical constraint. In Fig. 4(b), the best grasping configuration calculated for the object of Fig. 4(a) using only the informations provided by the video-camera are presented. The contact points are compatible with the mechanical architecture of the gripper.
The mathematical details can be found in [10, 11] and are summarized in [14].

5 Conclusions and future work

In this work, an experimental setup based on a 6 degrees of freedom (dof) manipulator with a 3 dof gripper designed for a possible application within PaT, the Payload Tutor proposed by ASI (Italian Space Agency), has been presented. This system consists of a robotic arm, the Comau SMART3-S industrial manipulator, a monocular vision system, and the A.S.I gripper. In this set-up, the gripper has to interact with free-floating and irregular objects and all the informations needed for grasping unknown objects in an optimal way are provided by the vision subsystem. Once an unknown object inside the robot's work-space is selected by the user, the robot is able to grasp it in an optimal way using only the informations provided by the vision subsystem.

Moreover, our future work will be aimed at improving performances of the current system, in particular introducing a stereo-vision system in order to easily obtain informations about the position of the object within the environment so that an object tracking can be implemented. Other activities will be devoted to further study the real-time performances of the several real-time variants of Linux, to compare them to commercial products, and to develop applications for high-dynamic motion-control systems.

References

[1] M. Barbanov, 1997, *A Linux-based Real-Time Operating System*, Master's thesis, New Mexico Institute of Mining and Technology, Socorro (NM), USA.

[2] *RT-Linux Home Page*, http://www.rtlinux.org/

[3] E. Bianchi, L. Dozio, P. Mantegazza, D. Martini, 1999, *Applications of a New Variant of RT-Linux in Digital Control of Dynamic Systems*, Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano.

[4] *RTAI-Linux Home Page*, http://www.rtai.org/

[5] D. Arduini, P. Arcara, C. Melchiorri, 1999, *Real-Time Linux in Robotics and Control System Research: A Practical Experience in the VIDET Project*, Real-Time Linux Workshop, Vienna, Austria.

[6] C. Melchiorri, 1999, *Traiettorie per azionamenti elettrici*, Esculapio Ed.,

[7] L. Biagiotti, F. Fabbri, C. Melchiorri, G. Vassura, 2000, *Control of a Robotic Gripper for Grasping Objects in No-Gravity Conditions*, 39th CDC'2000, Conf. on Decision and Control, Sydney, Australia.

[8] R. Zanasi, R. Morselli, *Second order smooth trajectory generator with nonlinear constraints*, European Control Conf. ECC'01, Oporto, Portugal, Sept. 2001.

[9] A. Macchelli, C. Melchiorri, *Real-time control system for industrial robots and control applications based on real-time Linux*, 15th IFAC World Congress, Barcelona, Spain, July 21-26, 2002.

[10] E. Rimon, J.W. Burdick, *Mobility of bodies in contact. I. A 2nd-order mobility index for multiple-finger grasps*, Robotics and Automation, IEEE Transactions on , Volume: 14, Issue: 5, Page(s): 696-708, Oct. 1998.

[11] E. Rimon, J.W. Burdick, *Mobility of bodies in contact. II. How forces are generated by curvature effects*, Robotics and Automation, IEEE Transactions on , Volume: 14, Issue: 5, Page(s): 709-717, Oct. 1998.

[12] P. Melucci, *Confronto di applicazioni di controllo in tempo reale sviluppate in RTAI- Linux e QNX*, Laurea Thesis (in italian), DEIS. Univ. of Bologna, Italy, 2000.

[13] M. Guidetti, *Integrazione di visione artificiale e controllo real-time per un robot industriale*, Laurea Thesis (in italian), DEIS. Univ. of Bologna, Italy, 2002.

[14] R. Carloni, *Manipolazione robotica: strategie di presa ottima per oggetti planari*, Laurea Thesis (in italian), DEIS. Univ. of Bologna, Italy, 2002.