23

Open Software Structure for Controlling Industrial Robot Manipulators

Flavio Roberti, Carlos Soria, Emanuel Slawiñski,
Vicente Mut and Ricardo Carelli
Universidad Nacional de San Juan
Argentina

1. Introduction

Automatic control has become an important part of the modern industrial processes. Progress both in basic research as applied to automatic control, provide a way to obtain the optimum performance of the dynamical systems, improve the quality and reduce the costs.
Robotics, as a part of automatics, represents nowadays an important research area, and it has an essential role in the productive modernization (UNECE and IFR, 2005). The inclusion of industrial manipulators in the manufacturing process allows obtaining better and cheaper products. Therefore, the development of an open software structure for the industrial robots controlling is a very important objective to be achieved (William, 1994), (Frederick and Albus, 1997).
The main characteristic of an open software structure for robotics applications is the interface that relates the components of the robot with the basic internal structure. In industrial area, one of the most important works was developed in the framework of the European project OSACA (Open System Architecture for Control within Automation Systems). Similarly, significant contributions were reached in Japan through OSEC (Open System Environment for Controllers) under IROFA (International Robotics and Factory Automation Center), (Sawada and Akira, 1997), and in the United States of America through OMAC (Open Modular Architecture Control). The objective of all these research projects is to develop an open control system including the reference model of the components, the general application interface and the structure so that all the components work together. So far manufacturers do not work together to develop standard control software that could be applied to any industrial robot.
On the other hand, several commercial software packages, that run under Windows, for mobile robots can be found. Among the best known ones, Advanced Robotics Interface for Applications (ARIA) is used in the robots manufactured by Mobile Robots Inc., BotController software were developed by MobotSoft and it is used for the well known Khepera and Koala robots. Even when these software packages are powerful and have many benefits, they can be applied only to the robots that were developed.

The main objectives of this chapter are the development and the implementation of an open software structure with reusable components, which works as a link between the hardware of an industrial robot manipulator and its control algorithm in order to implement these control algorithms with minimum efforts. Having this kind of software structure is very useful for researching and teaching in robotics as well as for industrial applications. The software structure runs under QNX Real Time operating system (Krten, 1999), and can be used for a large number of industrial robots.

With the aim of achieving the raised objectives, the developed system is compound by two different programs. First one is the responsible for the sensors' data acquisition and sending the control action to the servos. This program uses a shared memory block to save the data obtained from the sensors and to get the control action to be sent to the servos. In the second one runs the control algorithm. This program, similar to the first one, uses the same shared memory block to get the sensors' data and to save the control action to be sent to the servos. This way, the control algorithm execution is isolated from the signals transmission between the software and the robot's hardware, allowing a time and efforts reduction in the implementation of different control algorithms.

Then, two different controllers have been implemented in order to evaluate the performance of the proposed open software structure, applied to the SCARA robot manipulator Bosch SR-800. First, a classical PD (proportional-derivative) controller is used to allow the robot to achieve a desired position on the workspace. This controller uses the position information from the encoders of the robot. Finally, an advanced passivity based visual servoing with "*eye-in-hand*" camera configuration (Weiss et al., 1987) is implemented to allow the robot to reach a position relative to some static target. Additionally, finite L_2-gain for the passivity based control system is proven when a moving object is considered, allowing the robot to track the moving target with L_2-gain performance. Experimental results for both, the classical PD controller and the passivity based visual controller are presented in order to show the good performance of the proposed open software structure when it is applied to industrial robot manipulators.

This chapter is organized as follows. Section 2 describes the used industrial robot manipulator. Section 3 presents the open control software developed. Section 4 comments the control strategies used to evaluate the software structure and shows the experimental results. Finally, Section 5 presents same conclusions of the work.

2. Industrial robot Bosch SR-800

The robot manipulator Bosch SR-800 is 4 dof SCARA like industrial robotic arm. This kind of manipulator is useful for smooth and fast movements, especially for assembly tasks. First, second and fourth joints are rotational and they move on the horizontal plane; and third joint is linear and it moves on the vertical plane. Figure 1 shows the robot's configuration and it physical dimensions. It is important to remark that the third joint is uncoupled by a mechanical system based on toothed belts. This way, the end effector is always in the same orientation when no control action is applied to the third joint.

The manipulator Bosch SR-800 has a Riho control unit, provided by the manufacturer, consisting of four servo-amplifiers and a CPU. The servo-amplifiers command the joints of the robot and the CPU is used to compute a position control algorithm with internal velocity loop for each joint.

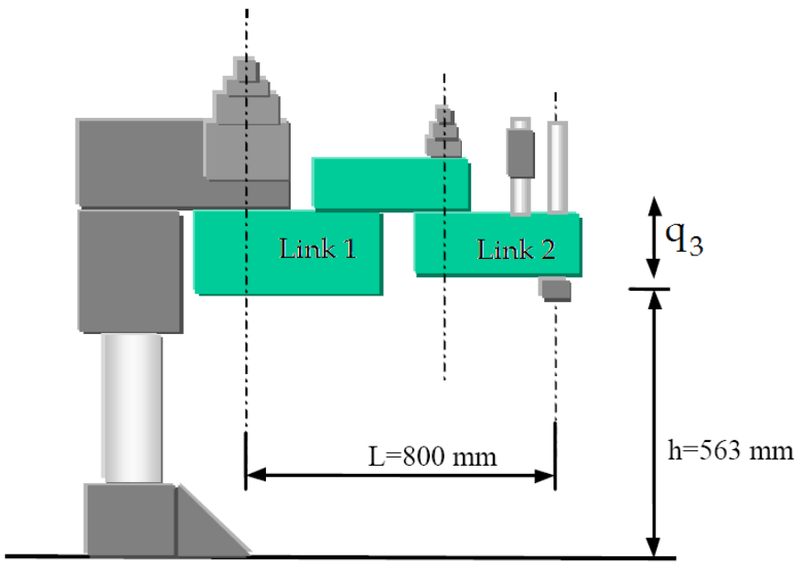


Fig. 1. Industrial robot Bosch SR-800

In order to reach the proposed objectives, the closed control system, i.e. the CPU provided by the manufacturer, was replaced by an open control system, i.e. a PC based control system. This new control system has input-output data boards AD/DA-Q12 from Microaxial®, to make the data interchange between the control system and the robot's hardware. A block diagram of the described system is shown in Fig. 2.The robot was also equipped was a force sensor FS6-120 and a vision camera Sony XC77, both located at the end effector of the robot.

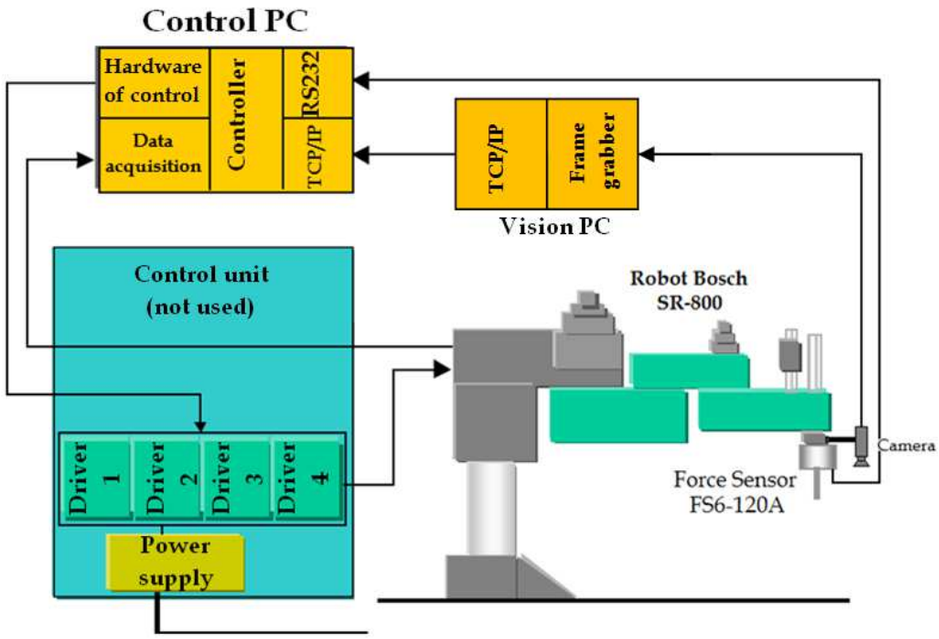


Fig. 2. Control diagram of the robot Bosch SR-800

2.1 Robot kinematic model

Let's consider the industrial manipulator briefly described above, with a global coordinate system whose origin is located at the intersection between the rotation axis of the first joint and the horizontal plane $\langle x, y \rangle$, as Fig. 3 shows.

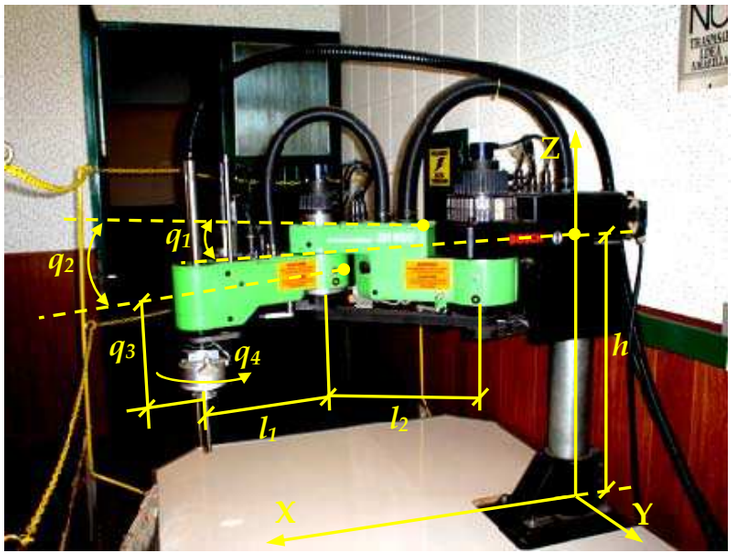


Fig. 3. Geometric description of the robot Bosch SR-800

In Fig. 3, l_1 and l_2 are the length of the first and second links respectively, q_i are the joint positions of each link, and h is the distance between the first link and the base of the robot. Then, the kinematic model that relates the position of the end effector with the joints variables are represented by the following set of equations,

$$\begin{aligned} x &= l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ y &= l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \\ z &= h - q_3 \end{aligned} \tag{1}$$

2.2 Dynamic model

In the absence of friction or other disturbances, the dynamics of a n-link rigid SCARA robot manipulator can be written as (Spong and Vidyasagar, 1989),

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} = \boldsymbol{\tau} \tag{2}$$

where:

$\mathbf{q} \in \mathrm{R}^{n\times 1}$ is the vector of joint displacements;

$\boldsymbol{\tau} \in \mathrm{R}^{n\times 1}$ is the vector of applied joint torques;

$\mathbf{M} \in \mathrm{R}^{n\times n}$ is the symmetric positive definite manipulator inertia matrix;

$\mathbf{C}\dot{\mathbf{q}} \in \mathrm{R}^{n\times 1}$ is the vector of centripetal and Coriolis torques.

Some important properties of the robot dynamics are the following.

Property 1— The time derivative of the inertia matrix, and the centripetal and Coriolis matrix satisfy

$$\mathbf{x}^{\mathrm{T}}\left[\frac{d}{dt}\mathbf{M}-2\mathbf{C}\right]\mathbf{x}=0 \quad \forall \mathbf{x}\in \mathrm{R}^{n}$$

that is, $\left[\dot{\mathbf{M}}-2\mathbf{C}\right]$ is an antisymmetric matrix.

Property 2 – The dynamic structure of the manipulator can be written as,

$$\mathbf{M}\ddot{\mathbf{q}}+\mathbf{C}\dot{\mathbf{q}}=\phi\left(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}\right)\theta$$

where $\phi\left(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}\right)\in \mathrm{R}^{n\times m}$; and $\theta\in \mathrm{R}^{n}$ is a vector of parameters.

Property 3 – Matrix $\mathbf{M}$ has the following properties,

- $\mathbf{M}=\mathbf{M}^{\mathrm{T}}>0$
- $\exists \inf\left\|\mathbf{M}\right\|$

For the considered robot manipulator Bosch SR-800,

$$\mathbf{M}=\begin{bmatrix}1.7277+0.1908\cos\left(q_2\right) & 0.0918+0.0954\cos\left(q_2\right)\\ 0.0918+0.0954\cos\left(q_2\right) & 0.0918\end{bmatrix};$$

$$\mathbf{C}=\begin{bmatrix}31.8192-0.0954\sin\left(q_2\right)\dot{q}_2 & -0.0954\sin\left(q_2\right)\left(\dot{q}_1+\dot{q}_2\right)\\ 0.3418\sin\left(q_2\right)\dot{q}_1 & 12.5783\end{bmatrix}$$

2.3 Camera model

A vision camera transforms a 3D space into a 2D projection on the image plane, where the vision sensor is located. This projection causes the lost of the depth perception, which means that each point on the image plane corresponds to a ray in the 3D space.

Several projection models for the representation of the image formation process have been proposed. The most used is the perspective projection model or "pin-hole" model. In this model, a coordinate system $\left\langle O_C,{}^{C}X,{}^{C}Y,{}^{C}Z\right\rangle$ attached to the camera is defined in such a way that the X and Y axes define a base for the image plane and the Z axis is parallel to the optic axis. The origin of the framework $\left\langle O_C,{}^{C}X,{}^{C}Y,{}^{C}Z\right\rangle$ is located at the focus of the camera lens.

From Fig. 4, a fixed point $\mathbf{P}$ in the 3D space with coordinates $\mathbf{P}=\begin{bmatrix}X_C & Y_C & Z_C\end{bmatrix}^{T}$ on the framework attached to the perspective camera will be projected on the image plane as a point with coordinates $\xi=\begin{bmatrix}u & v\end{bmatrix}^{T}$ given by (Hutchinson et al., 1996),

$$\xi=-\frac{\lambda}{Z_C}\begin{bmatrix}X_C\\ Y_C\end{bmatrix} \tag{3}$$

where λ is the focal length of the camera expressed in pixels.

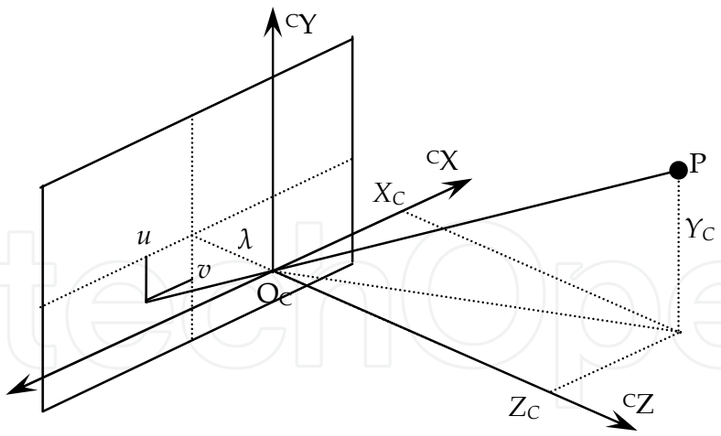


Fig. 4. Perspective projection camera model

2.3.1 Punctual image feature

An image feature is usually defined as a quantifiable relation on the image plane. In (Jang et al., 1991), a formal definition for image features is given,

$$f = \iint \Im(u, v, I(u, v))\, du\, dv \tag{4}$$

where $I(u,v)$ is the intensity of the pixel at the position (u,v). Function $\Im$ can be a linear or a non lineal mapping, depending on the considered image feature. It may even be a delta function.

Some common examples of image features are:

- Cross-correlation correspondence or sum of squares' difference to determine the coordinates of a known pattern of pixels in the image.
- Spatial or central moments of the image.
- Length or orientation of objects' borders.
- Length or orientation of the segments that connect different objects in the scene.

In (Kelly et al., 2000), it is presented the relation between the time variation of the image feature vector and the movement velocity of an object relative to the vision system placed at the end effector of the robot, when a punctual image feature is considered.

$$\dot{\xi} = \mathbf{J}_{img} \begin{bmatrix} {}^{C}\mathbf{R}_{W} & 0 \\ 0 & {}^{C}\mathbf{R}_{W} \end{bmatrix} \mathbf{J}_{G}\dot{\mathbf{q}} + \mathbf{J}_{O}\dot{\mathbf{P}} \tag{5}$$

where ${}^{C}\mathbf{R}_{W}$ is the rotation matrix of the coordinate system attached to the robot's base relative to the coordinate system attached to the vision camera; $\mathbf{J}_{G}$ is the geometric Jacobian of the robot (Sciavicco and Siciliano, 2001); and $\mathbf{J}_{img}$ and $\mathbf{J}_{O}$ are the image and the object Jacobians respectively, with:

$$\mathbf{J}_{img} = \begin{bmatrix} \frac{\lambda}{Z_C} & 0 & \frac{u}{Z_C} & -\frac{uv}{f} & \frac{\lambda^2+u^2}{\lambda} & v \\ 0 & \frac{\lambda}{Z_C} & \frac{v}{Z_C} & -\frac{\lambda^2+v^2}{\lambda} & \frac{uv}{\lambda} & -u \end{bmatrix}; \ \mathbf{J}_O = -\frac{\lambda}{Z_C} \begin{bmatrix} 1 & 0 & -\frac{X_C}{Z_C} \\ 0 & 1 & -\frac{Y_C}{Z_C} \end{bmatrix} {}^C\mathbf{R}_W$$

3. Open software design

3.1 Users

The software system developed in this chapter is expected to be useful in control system teaching, human resources training, research, as well as in industrial area. Users in all these areas can be classified in four different levels, depending on how they would use the software system.

- *Level 1*: those users who do not need to make any changes in the software system, for example: undergraduate students.

- *Level 2*: those users who need to evaluate the performance of new control algorithms. They would need to modify just the implemented control law, using the rest of the system without any change. Those users have to have minimum knowledge about data structures and the system operation in order to make appropriate modifications. For example: postgraduate students, researchers.

- *Level 3*: those users who want to make they own control software implementation, using only the sensors' data acquisition program.

- *Level 4*: those users who need to add one or more sensors or actuators in the system. Those users have to be knowledgeable about data structures and the system operation.

3.2 Operational requirements

Based on the main objectives of this development, the operational requirements of the software system are:

- The control software for the industrial robot manipulator Bosch SR-800 must allow implementing and evaluating different control algorithms, using the information from the force sensor, position sensors, and visual sensor. All relevant data of the experiments have to be saved for later analysis.
- The software developed must be flexible and with an open architecture in order to facilitate the incorporation of new components, such as sensors, actuators, teleoperation devices, etc.

3.3 Reuse-based design

In many engineering disciplines, like mechanical of electrical engineering, the design process is based on the reuse of the components. In the last decades, software engineering

has directed its efforts to imitate these techniques by encapsulating software units for its later reuse (Sommerville, 2000). With this aim, object oriented architecture is developed to handle different devices and hardware components, such as sensors, actuators, teleoperation devices. Therefore, data and inner tasks of each device are encapsulated, running in independents threads. This way, the software modules designed for each device can be reused for the inclusion of some new hardware component.

3.4 Operating system and programming environment

All the software development was made under platform QNX (Krten, 1999). This operating system has been selected because it is one of the best real time operating systems with high stability and robustness of operation. Additionally, QNX support multi-processors systems and several benefits can be obtained from the memory management unit (MMU) protection. The programming language chosen is C++, and the user interface has been implemented by using the Photon microGui.

Different objects in the software are implemented in classes, which are initialized at the beginning of the program but they do not star working until their activation function is called. In the particular cases of objects related to the sensors and the actuators, each one of them has an associated function that runs in a different thread, with a suitable sample time for each device.

3.5 Design of the software structure

The software structure is designed with independent modules for the user interface, the hardware devices, and the control algorithm. Figure 5 shows a block diagram of the software structure. The different tasks are divided into two processes or programs that communicate each other and work cooperatively. Communication task between the software and the hardware devices, and the synchronization of the control sample time are carried out by the so called *Critic Time Program*; whereas the control algorithm runs in the so called *Control Program*.

The function that implements the control algorithm can be easily modified to allow evaluating different control strategies with a minimum effort. This function is called at each control sample instant, which is defined by the user through the user interface.

In the following sections, main characteristics of both the *Critic Time Program* and the *Control Program* of the software are briefly described.

3.5.1 Critic Time Program

The *Time Critic Program* is responsible for communicating with the sensors and the actuators through the data acquisition and control hardware, updating the sensors' data in the shared memory block, and it is also responsible for synchronizing the *Control Program* for the correct running of the control algorithm at each sample instant.

This program has four different classes,

- Motor: this object is responsible for applying the control actions obtained by the control algorithm to the motors of the industrial manipulator through the D/A converter of the data acquisition and control hardware.

- Vision system: this object uses the TCP/IP connection functions to receive the visual information from the vision PC. This vision PC process the image obtained through the camera and sends the image features to the Critic Time Program via the TCP/IP connection.
- Position: this object is responsible for obtaining the position data from the internal encoders of each joint of the industrial robot. The data acquisition hardware is used to carry out this task.
- Force: this object is responsible for obtaining the force data from the force sensor FS6-120. The serial port RS232 of the control PC is used.

Additionally, a timer is used in this program to determine the sample instant of the control algorithm; and a graphic user interface is also implemented. Through this interface, users have a set of graphic controls that allow them to select the desired sensors and set their parameters, set the sample period, and start or stop the experiment.

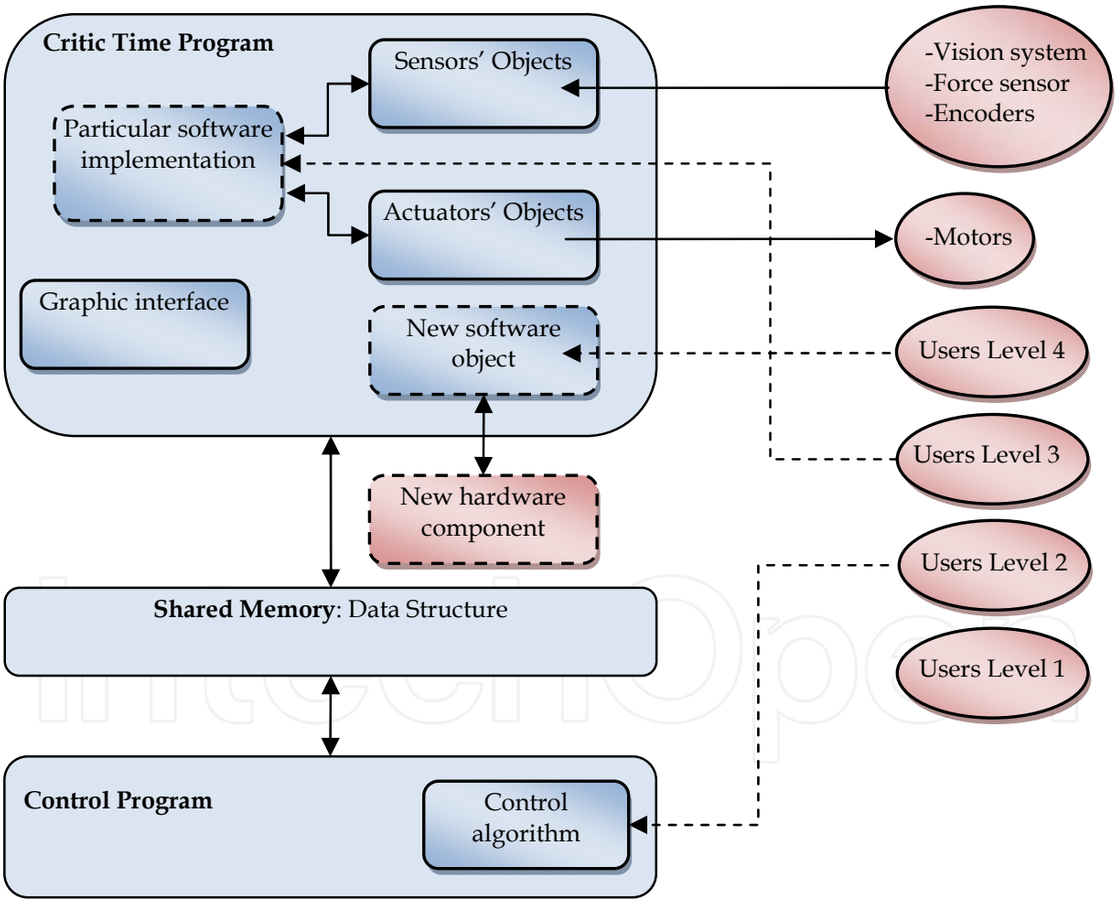


Fig. 5. Block diagram of the developed software system

3.5.2 Control Program

As explained above, the control algorithm runs in this program. Since the *Control Program* may be modified by users of *Level 2*, who may have not a large experience in software

development, some issues are commented. These issues should be taken into account to develop a program that efficiently uses the available hardware resources.

- Determine the correct number of threads of the program, according to available PC hardware.
- Avoid high time demanding operations. Perform I/O operations on files and communication devices asynchronously.
- Do not use global variables. The reuse-based design using object oriented programming is desirable.
- Use shared memory blocks for the data interchange between different processes.
- Use events for the system synchronization.
- If an on-line data writing to a hard disk device is needed, use a double buffer structure and an asynchronous writing.
- Determine and set the correct priority of each thread according to its tasks.

4. Implemented control laws

The open software system developed has been tested by the implementation of two different control strategies. First, a classical PD position controller was implemented, based in the robot position information obtained from the internal encoders of the robot. Then, a passivity based visual controller was implemented. This way, the performance of the software system is evaluated not only when internal sensors are used, but also when a vision camera placed at the end effector of the robot is used as sensor of the control system. In addition, it allows showing the possibility of a fast and easy control law interchange.
Throughout this Section, a brief description of the control laws and some experimental results will be presented.

4.1 PD controller

The PD controller is a typical control algorithm used in robotics teaching. With the proposed open software structure, teaching duties relative to the laboratory experimentation can be fast and easy, bringing more time to the theoretical classes. Next, a brief explanation of the PD controller is presented.
The PD position controller is defined as,

$$\boldsymbol{\tau} = \mathbf{K}_p \tilde{\mathbf{q}} + \mathbf{K}_v \dot{\tilde{\mathbf{q}}} \tag{6}$$

where $\mathbf{K}_p = diag\{k_{pi}\}$ and $\mathbf{K}_v = diag\{k_{vi}\}$ are positive definite gain matrices; $\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q}$ is the joint position error; and $\dot{\tilde{\mathbf{q}}} = -\dot{\mathbf{q}}$ since a position problem is considered . A block diagram of the control system is shown in Fig. 6.
By equating the control law (6) with the robot's dynamic model (2), the close loop equation is obtained,

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} = \mathbf{K}_p \tilde{\mathbf{q}} - \mathbf{K}_v \dot{\mathbf{q}} \tag{7}$$

Considering the following Lyapunov candidate function and its time derivative (Slotine and Li, 2001; Khalil, 2001),

$$V = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}} + \frac{1}{2}\tilde{\mathbf{q}}^T\mathbf{K}_p\tilde{\mathbf{q}}$$
$$\dot{V} = -\dot{\mathbf{q}}^T\mathbf{K}_v\dot{\mathbf{q}} \leq 0 \tag{8}$$

and recalling La Salle theorem (Slotine and Li, 2001; Khalil, 2001), the asymptotic stability of the control system can be proven.

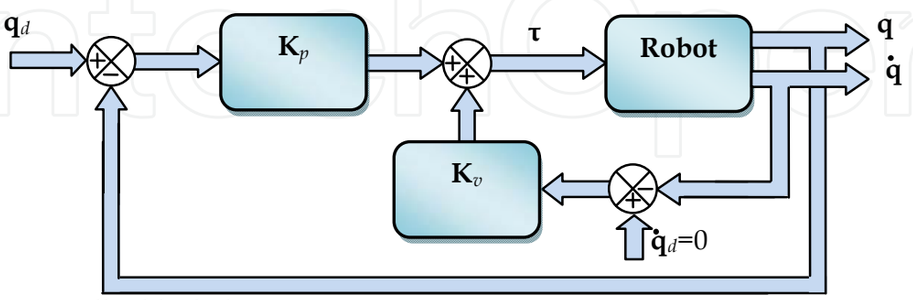


Fig. 6. PD controller block diagram

4.2 Passivity based visual controller

The proposed open software structure can also be used for the experimentation of new advanced control algorithms, such as passivity based visual servoing. This way, researchers can find in the proposed software system a useful experimentation platform, saving time in the implementation, focusing their efforts on the controllers design. Next, a brief explanation of the passivity based visual controller is presented.

Passivity is an important property between input and output of a system that has been widely used in the stability analysis of non-lineal systems (Hill and Moylan, 1976; Lin, 1995; Willems, 1972a; Willems, 1972b) and the stability analysis of interconnected systems, especially in cascade structures (Vidyasagar, 1979; Byrnes et al., 1991; Ortega et al., 1995). The concept of passivity shows, in an intuitive way, that a passive system cannot provide more energy than the energy received, and it allows to prove that a non linear passive system can be stabilized by a simple negative output feedback $v = -k\,y$, with $k > 0$ (see Fig. 7). Therefore, passivity is a useful property for the non linear systems analysis and design, representing a good alternative to the Lyapunov method.

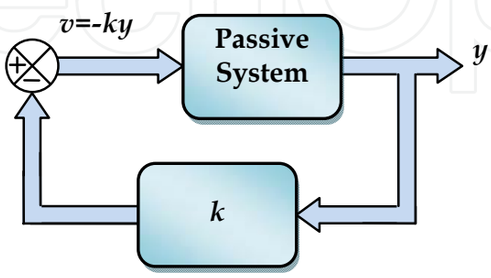


Fig. 7. Stabilized passive system

Some important definitions about the passive systems theory are (Ortega et al., 1998; van der Schaft, 2000),

Definition 1 – The mapping $H : L_{2e} \to L_{2e}$ is passive if there exist some constant β such that,

$$\langle Hx, x \rangle_T \geq \beta \quad \forall x \in L_2; \quad \forall T \in [0, \infty]$$

Definition 2 – The mapping $H : L_{2e} \to L_{2e}$ is strictly passive if there exist some constants $\delta > 0$ and β such that,

$$\langle Hx, x \rangle_T \geq \beta + \delta \|x\|_{2,T}^2 \quad \forall x \in L_2; \quad \forall T \in [0, \infty]$$

Definition 3 – The mapping $H : L_{2e} \to L_{2e}$ is strictly input passive if there exist some constants δ and β such that,

$$\langle Hx, x \rangle_T \geq \beta + \delta \|x\|_{2,T}^2 \quad \forall x \in L_2; \quad \forall T \in [0, \infty]$$

Definition 4 – The mapping $H : L_{2e} \to L_{2e}$ is strictly output passive if there exist some constants δ and β such that,

$$\langle Hx, x \rangle_T \geq \beta + \delta \|Hx\|_{2,T}^2 \quad \forall x \in L_2; \quad \forall T \in [0, \infty]$$

4.2.1 Passivity property of the vision system

Considering a static object $\dot{\mathbf{P}} = \mathbf{0}$, equation (5) can be written as,

$$\dot{\xi} = \mathbf{J}\dot{\mathbf{q}} \tag{9}$$

where $\mathbf{J} = \mathbf{J}_{img} \begin{bmatrix} {}^{C}\mathbf{R}_{W} & 0 \\ 0 & {}^{C}\mathbf{R}_{W} \end{bmatrix} \mathbf{J}_{G}$; being $\mathbf{J}_{img}$ the image Jacobian matrix, and $\mathbf{J}_{G}$ the robot geometric Jacobian (Kelly et al., 2000).

Taking the energy function $V_{\xi} = \frac{1}{2} \xi^{\mathrm{T}} \xi$ and making its time derivative (Fujita et al., 2007),

$$\dot{V}_{\xi} = \xi^{\mathrm{T}} \dot{\xi} = \xi^{\mathrm{T}} \mathbf{J} \dot{\mathbf{q}} \tag{10}$$

and integrating in $[0, T]$

$$\int_0^T \dot{V}_{\xi} \, dt = \int_0^T \xi^{\mathrm{T}} \mathbf{J} \dot{\mathbf{q}} \, dt = \int_0^T \mathbf{v}_{\xi}^{\mathrm{T}} \dot{\mathbf{q}} \, dt = V_{\xi}(T) - V_{\xi}(0) \geq -V_{\xi}(0) \tag{11}$$

where $\mathbf{v}_{\xi} = \mathbf{J}^{\mathrm{T}} \xi$.

Therefore, it can be concluded that the mapping $\mathbf{v}_{\xi} \to \dot{\mathbf{q}}$ is passive.

4.2.2 Control system design

Considering now the variable $\tilde{\xi}(t) = \xi(t) - \xi_{\mathrm{d}}$ instead of $\xi(t)$ in order to contemplate the regulation problem, and also considering perfect velocity tracking ($\dot{\mathbf{q}} \equiv \mathbf{u}$), it is possible to prove that the passivity property of the vision system is preserved, that is,

$$\int_0^T \mathbf{u}^{\mathrm{T}} \mathbf{v}_{\tilde{\xi}}\, dt \geq -\beta \quad \forall T\,;\ \text{then}\ \mathbf{u} \rightarrow \mathbf{v}_{\tilde{\xi}}\ \text{is passive.} \tag{12}$$

where $\mathbf{v}_{\tilde{\xi}} = \mathbf{J}^{\mathrm{T}}\tilde{\boldsymbol{\xi}}$ and $\beta = V_{\tilde{\xi}}(0)$ with $V_{\tilde{\xi}} = \frac{1}{2}\tilde{\boldsymbol{\xi}}^{\mathrm{T}}\tilde{\boldsymbol{\xi}}$.

Then, the following control law is proposed, according to the general structure of Fig. 7,

$$\begin{aligned} \mathbf{u} &= -\mathbf{K}\mathbf{v}_{\tilde{\xi}} \\ \mathbf{u} &= -\mathbf{K}\mathbf{J}^{\mathrm{T}}\tilde{\boldsymbol{\xi}} \end{aligned} \tag{13}$$

where $\mathbf{K}$ is a symmetric and positive definite gain matrix. The control structure is shown in Fig. 8.

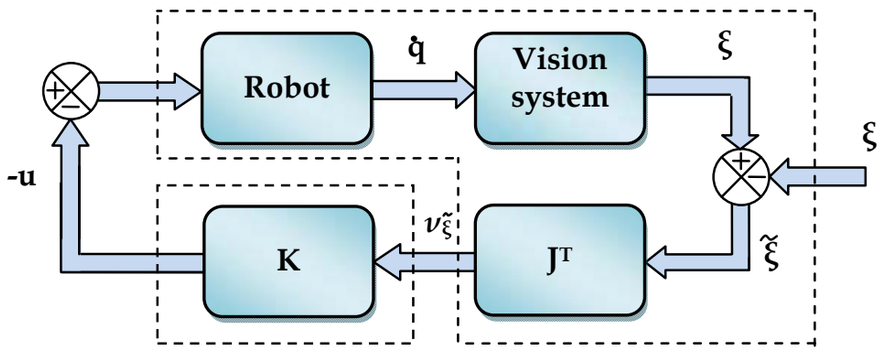


Fig. 8. Block diagram of the passivity based control approach

4.2.2 Control system analysis

From (12), and replacing the control law expression (13),

$$\int_0^T \mathbf{v}_{\tilde{\xi}}^{\mathrm{T}}\mathbf{u}\, dt = -\int_0^T \mathbf{v}_{\tilde{\xi}}^{\mathrm{T}}\mathbf{K}\mathbf{v}_{\tilde{\xi}}\, dt \leq -\lambda_{\min}(\mathbf{K})\int_0^T \left\|\mathbf{v}_{\tilde{\xi}}\right\|^2 dt \tag{14}$$

Or,

$$\int_0^T \mathbf{v}_{\tilde{\xi}}^{\mathrm{T}}(-\mathbf{u})\, dt \geq \lambda_{\min}(\mathbf{K})\int_0^T \left\|\mathbf{v}_{\tilde{\xi}}\right\|^2 dt \tag{15}$$

where $\lambda_{\min}(\mathbf{K})$ represents the minimum eigenvalue of matrix $\mathbf{K}$. Therefore, from Definition 3, the controller is strictly input passive from $\mathbf{v}_{\tilde{\xi}} \rightarrow -\mathbf{u}$. This way, the closed loop system of Fig. 8 is built by the interconnections of passive subsystems.

By adding equations (12) and (15), the following inequality can be obtained

$$\begin{aligned} \lambda_{\min}(\mathbf{K})\int_0^T \left\|\mathbf{v}_{\tilde{\xi}}\right\|^2 dt - \beta \leq 0 \\ \int_0^T \left\|\mathbf{v}_{\tilde{\xi}}\right\|^2 dt \leq \frac{\beta}{\lambda_{\min}(\mathbf{K})} \end{aligned} \tag{16}$$

which implies that $\dot{\mathbf{v}}_{\tilde{\xi}} \in L_2$. Then, for $\dot{\mathbf{v}}_{\tilde{\xi}} \in L_\infty$, the Barbalat's lemma allows concluding that $\mathbf{v}_{\tilde{\xi}}(t) \rightarrow 0$, and therefore $\tilde{\boldsymbol{\xi}} \rightarrow \mathbf{0}$ with $t \rightarrow \infty$, achieving the control objective.

4.2.3 Robustness to the object movement: L_2-gain performance design

In this section, the possibility of moving objects existence is considered and the control system's performance for tracking tasks is evaluated. With this aim, the object's velocity is considered as an external disturbance of the control system and a robust controller with L_2-gain performance criteria is designed (Fujita et al., 2007).

The system $\mathbf{w} \to \tilde{\boldsymbol{\xi}}$ would have finite L_2-gain if (van der Schaft, 2000),

$$\int_0^T \left\| \tilde{\boldsymbol{\xi}} \right\|^2 dt \le \gamma^2 \int_0^T \left\| \mathbf{w} \right\|^2 dt + \delta \quad ; \quad \forall T > 0 \tag{17}$$

being $\mathbf{w} = \mathbf{J}_O \dot{\mathbf{P}}$ the object's velocity on the image plane, considered as an external disturbance; $\gamma > 0$; and $\delta > 0$. In this context, γ represents an indicator of the system's tracking performance. The proposed control system will have finite L_2-gain if

$$\dot{V}_{\tilde{\xi}} \le \frac{1}{2}\left(\gamma^2 \left\| \mathbf{w} \right\|^2 - \left\| \tilde{\boldsymbol{\xi}} \right\|^2 \right) \tag{18}$$

As can be easily verified by integrating (18) in $[0, T]$. In order to find a gain matrix $\mathbf{K}$ that fulfils the L_2-gain performance criteria (18), it is considered again the positive definite function $V_{\tilde{\xi}}$ and its time derivative,

$$\begin{aligned} V_{\tilde{\xi}} &= \frac{1}{2} \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \tilde{\boldsymbol{\xi}} \\ \dot{V}_{\tilde{\xi}} &= \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \dot{\tilde{\boldsymbol{\xi}}} = \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \left(\mathbf{J}\dot{\mathbf{q}} + \mathbf{J}_O \dot{\mathbf{P}} \right) \end{aligned} \tag{19}$$

Considering again perfect velocity tracking ($\dot{\mathbf{q}} \equiv \mathbf{u}$), the control law (13) is introduced in (19),

$$\dot{V}_{\tilde{\xi}} = \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \dot{\tilde{\boldsymbol{\xi}}} = \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \mathbf{J}\mathbf{K}\mathbf{J}^{\mathrm{T}} \tilde{\boldsymbol{\xi}} + \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \mathbf{w} \tag{20}$$

and imposing L_2-gain performance condition (18) to (20), the following inequality is obtained,

$$\dot{V}_{\tilde{\xi}} = \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \dot{\tilde{\boldsymbol{\xi}}} = \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \mathbf{J}\mathbf{K}\mathbf{J}^{\mathrm{T}} \tilde{\boldsymbol{\xi}} + \tilde{\boldsymbol{\xi}}^{\mathrm{T}} \mathbf{w} \le \frac{1}{2}\left(\gamma^2 \left\| \mathbf{w} \right\|^2 - \left\| \tilde{\boldsymbol{\xi}} \right\|^2 \right) \tag{21}$$

Reorganizing (21), the following matrix inequality is obtained,

$$\begin{bmatrix} \tilde{\boldsymbol{\xi}}^{\mathrm{T}} & \mathbf{w}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} -\mathbf{J}\mathbf{K}\mathbf{J}^{T} + \frac{\mathbf{I}}{2} & \frac{\mathbf{I}}{2} \\ \frac{\mathbf{I}}{2} & -\frac{\mathbf{I}}{2}\gamma^2 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\xi}} \\ \mathbf{w} \end{bmatrix} \le 0 \tag{22}$$

The problem now is to find a symmetric and positive definite matrix $\mathbf{K}$ and a value for γ, such that the matrix inequality (22) is fulfilled. With this aim, the LMI technique (Boyd et al.,

1994) is used by restricting the Jacobian matrix $\mathbf{J}$ to a convex set. The only restriction imposed to the gain matrix $\mathbf{K}$ is that it must be symmetric and positive definite (in order to fulfill the passivity property of the controller (15)), and the condition of be diagonal is not imposed allowing the controller to incorporate dynamics coupling, obtaining better performances.

Now, the problem that immediately rises in the selection of the gain matrix $\mathbf{K}$ is that, if a small value for γ is adopted for a good performance in moving objects tracking, actuators could be saturated in presence of large image features errors. On the other hand, if a large value for γ is adopted, the saturation of the actuators would be avoided to de detriment of the tracking performance. The proposed solution to this problem lies in the use of a variable gain matrix, as a function of the image features error. With this aim, two different gain matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ are found by solving the matrix inequality (22) ($\mathbf{K}_1$ for small features errors and $\mathbf{K}_2$ for large features errors), and the gain matrix $\mathbf{K}$ is obtained as,

$$\mathbf{K} = (1-\alpha)\mathbf{K}_1 + \alpha\mathbf{K}_2 \tag{23}$$

where $\alpha = \dfrac{\left\|\tilde{\boldsymbol{\xi}}\right\|}{\left\|\tilde{\boldsymbol{\xi}}\right\|_{\max}}$; and being $\left\|\tilde{\boldsymbol{\xi}}\right\|_{\max}$ the maximum image features error. This way, matrix $\mathbf{K}$ always fulfils the performance condition γ, accepting a large value for large image features errors and adopting a smaller value for small image features errors, according to the design specifications.

4.3 Experimental results

Both the PD controller and the passivity based visual controller explained above were implemented in the industrial robot manipulator Bosch SR-800 shown in Fig. 9, with the open software developed. For the first experiment, as well as for the second one, it could be confirmed the fast implementation of the control algorithms.

Fig. 9. Industrial robot manipulator Bosch SR-800, at the National University of San Juan, Argentina

4.3.1 Experimental results for the PD controller

Some experiments are carried out with the classical PD controller, considering only first and second joint of the robot. The controller is implemented in the Control Program and runs with a sample time of 1 msec. In the first experiment, the end effector of the robots must achieve the desired position $(50, 30)$ (expressed in centimetres) on the Cartesian space, which means that the desired joint positions are $q_1 = 1.194$ rad and $q_2 = -1.52$ rad . On the other hand, in the second experiment, the end effector of the robots must achieve the desired position $(50, -30)$ (expressed in centimetres) on the Cartesian space, which means that the desired joint positions are $q_1 = 0.113$ rad and $q_2 = -1.52$ rad . In both experiments, the following gain matrices were used,

$$\mathbf{K}_p = \begin{bmatrix} 40.85 & 0 \\ 0 & 46.39 \end{bmatrix}$$

$$\mathbf{K}_v = \begin{bmatrix} 12.74 & 0 \\ 0 & 13.62 \end{bmatrix}$$

Figures 10, 11 and 12 show the results for the first experiment. Figures 10 and 11 show the time evolution of the joint positions; and Fig. 12 shows the trajectory described by the end effector on the Cartesian space. Figures 13, 14 and 15 show the results for the second experiment. Figures 13 and 14 show the time evolution of the joint positions; and Fig. 15 shows the trajectory described by the end effector on the Cartesian space.

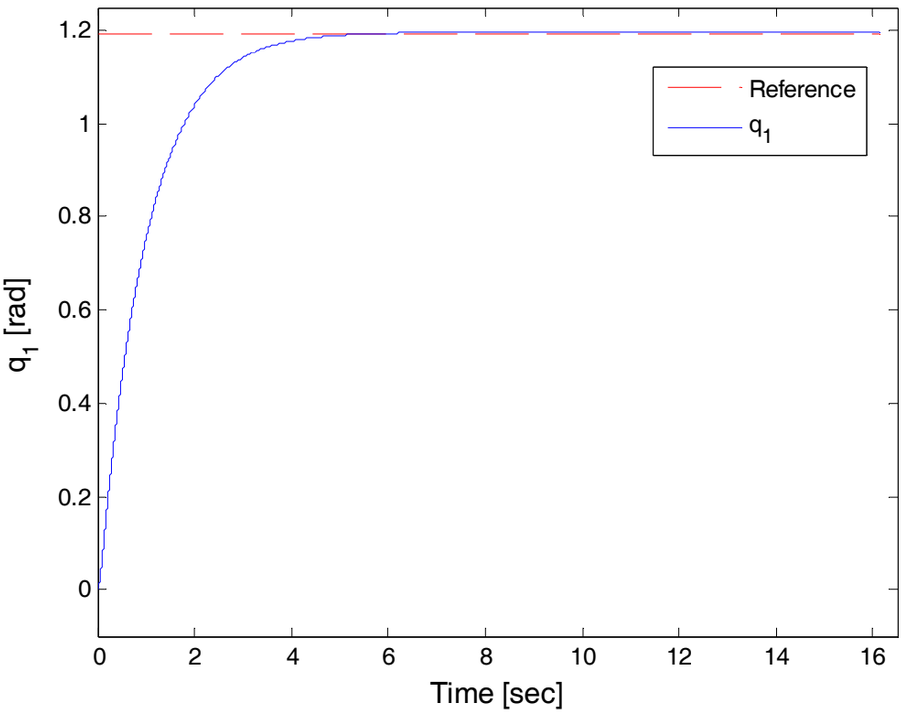


Fig. 10. Time evolution of q_1

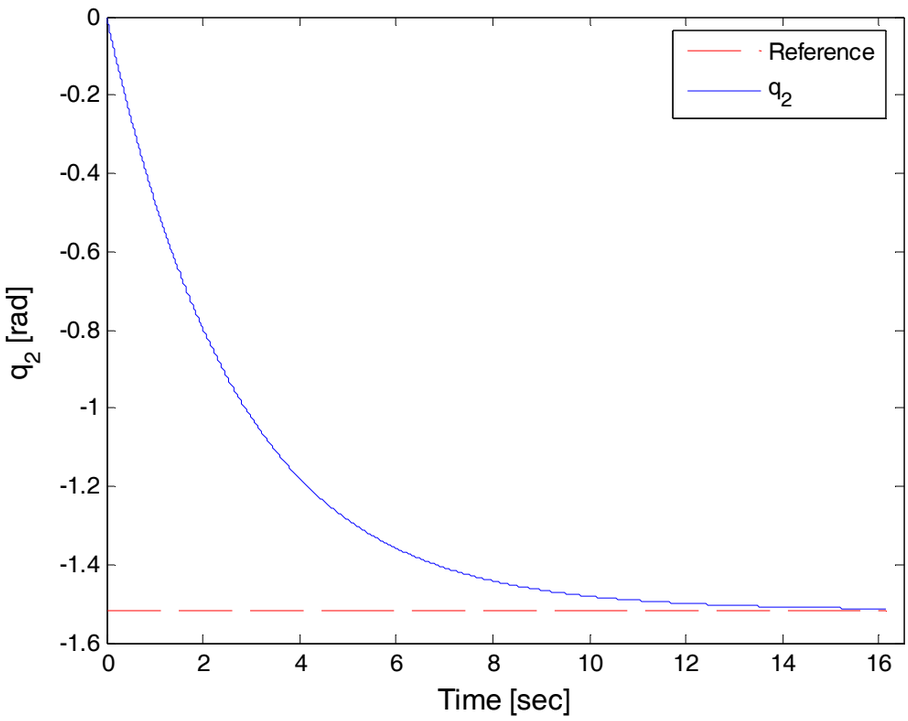


Fig. 11. Time evolution of q_2

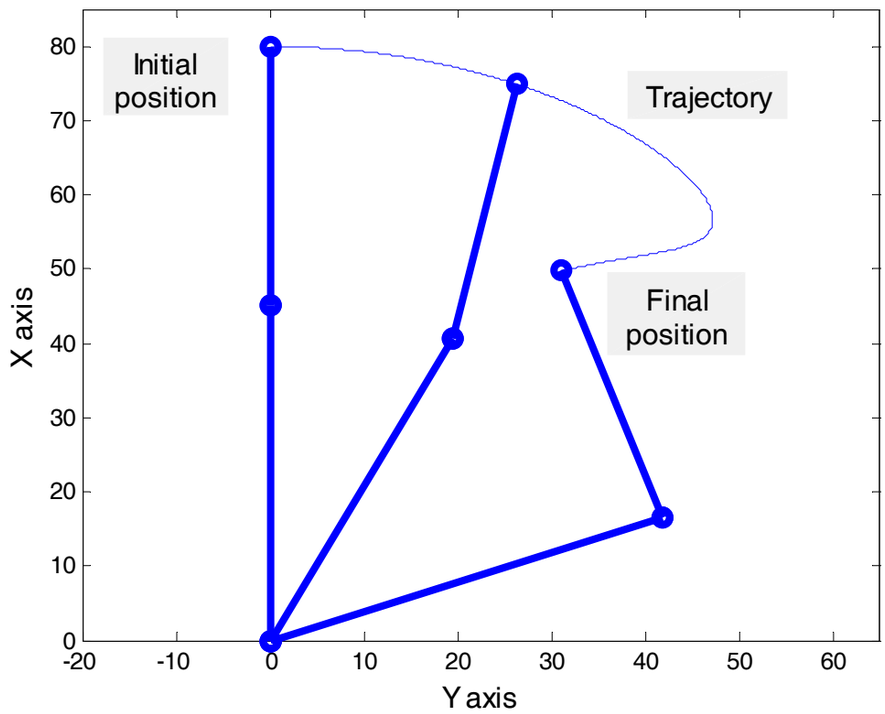


Fig. 12. Trajectory described on the Cartesian space

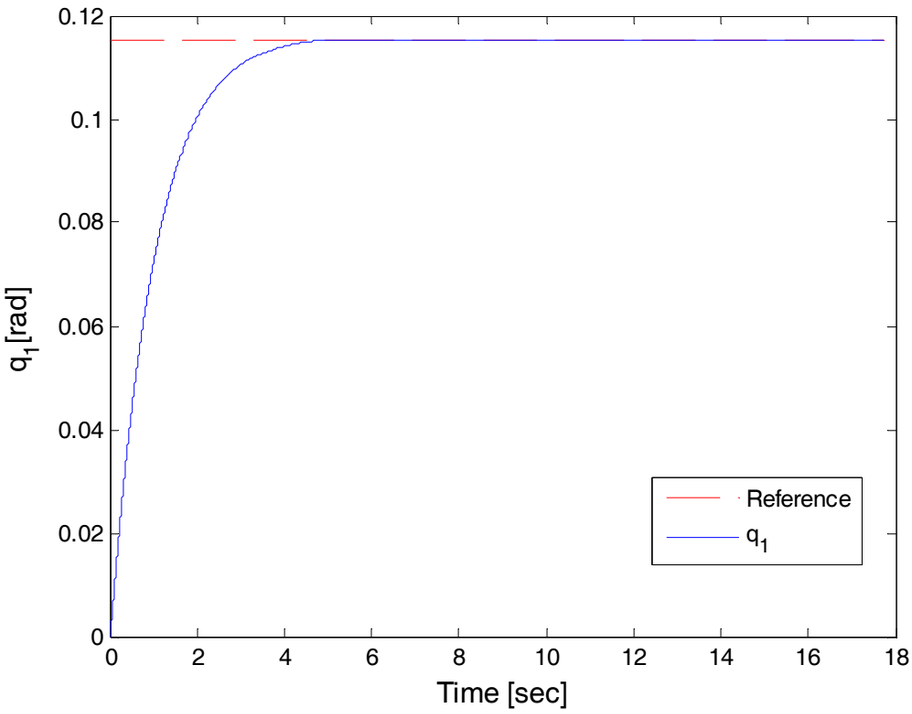


Fig. 13. Time evolution of q_1

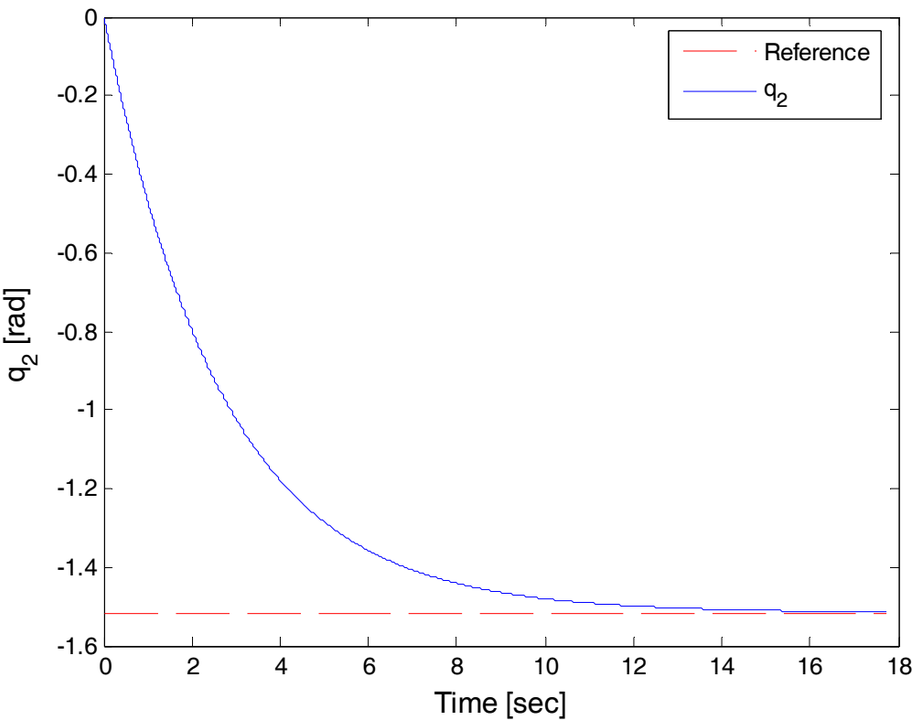


Fig. 14. Time evolution of q_2

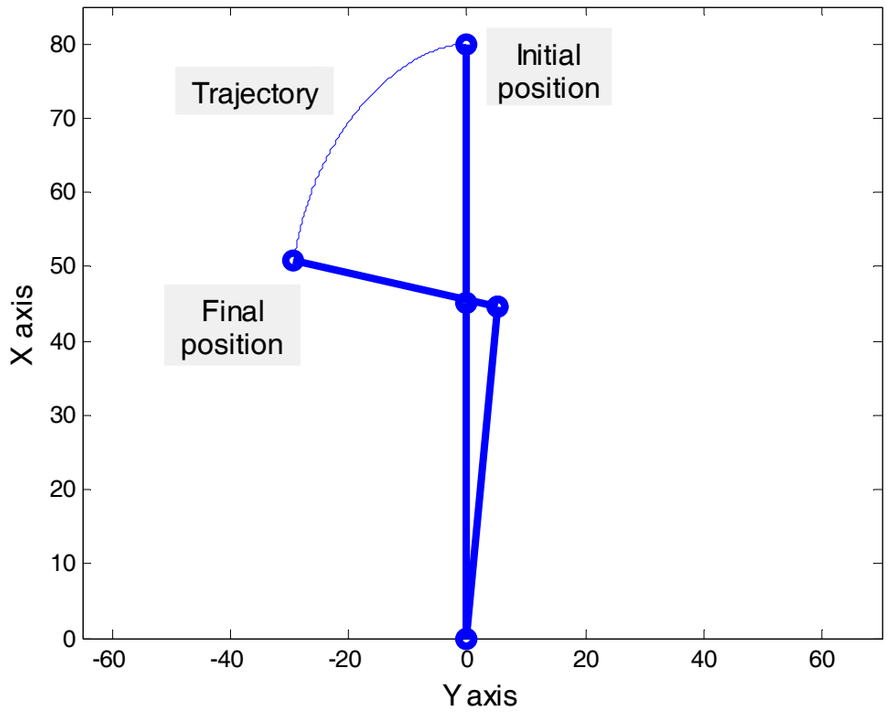


Fig. 15. Trajectory described on the Cartesian space

4.3.2 Experimental results for the visual controller

Third experiment is carried out with the passivity based visual controller, considering only first and second joint of the robot. The controller is implemented in the Control Program and runs with a sample time of 1 msec. for the controller and 33 msec. for the image processing. The gain matrices, obtained with the LMI-tool (El Ghaoui et al., 1995) are,

$$\mathbf{K}_1 = 10^{-5}\begin{bmatrix} 0.1443 & -0.1443 \\ -0.1443 & 0.4096 \end{bmatrix} \text{ with } \gamma = 3.9$$

$$\mathbf{K}_2 = 10^{-4}\begin{bmatrix} 0.0496 & -0.0496 \\ -0.0496 & 0.1399 \end{bmatrix} \text{ with } \gamma = 0.9$$

The experiment starts with an initial vector of image features $\boldsymbol{\xi}(0) = \begin{bmatrix} -48 & -65 \end{bmatrix}$ pixels and the first reference on the image plane is chosen as $\boldsymbol{\xi}_{d1} = \begin{bmatrix} 0 & 2 \end{bmatrix}$ pixels, and then the reference changes to $\boldsymbol{\xi}_{d2} = \begin{bmatrix} -72 & 64 \end{bmatrix}$ pixels. At instant $t = 15$ sec. the object starts moving.

Figures 16 and 17 show the time evolution of the image features ξ_1 and ξ_2 respectively, being ξ_1 and ξ_2 the components of the vector $\boldsymbol{\xi}$. The time evolution of the features error norm can be seen in Fig. 18. In this last plot, it can be seen that the image error is below 2 pixels when the object is not moving ($t < 15$ sec); and with a moving object, the features error is below 10 pixels. Figure 19 shows the control actions for q_1 and q_2. Finally, Fig. 20 shows the evolution of the image features on the image plane.

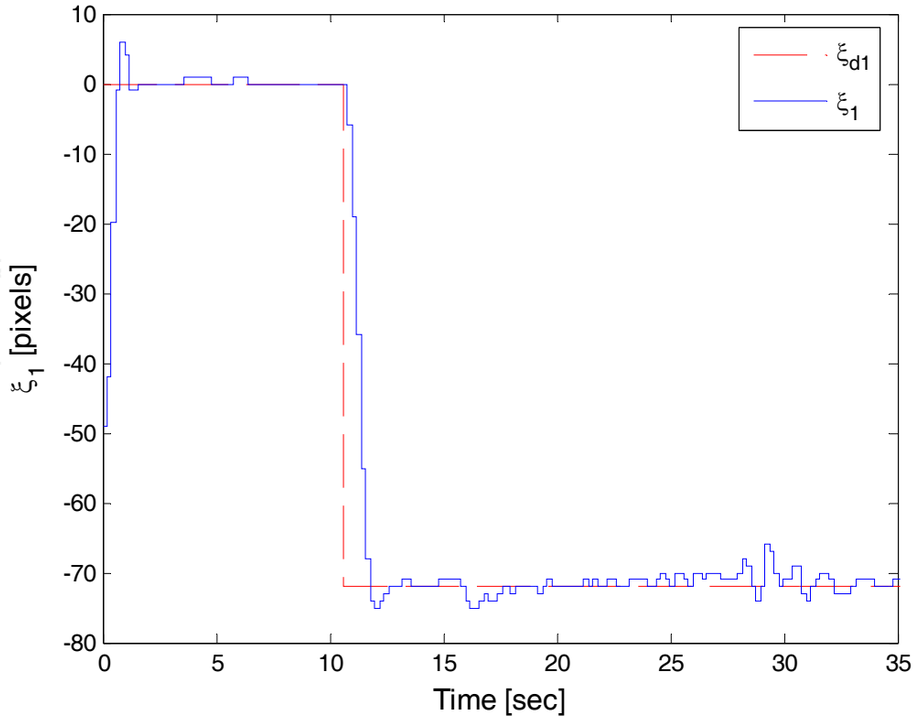


Fig. 16. Time evolution of the image feature ξ_1

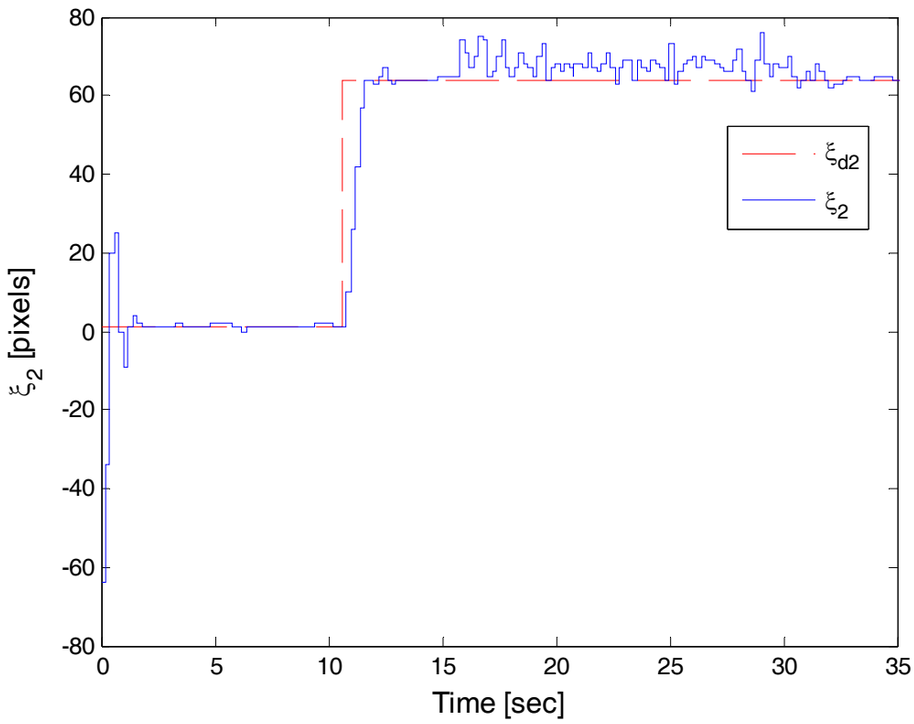


Fig. 17. Time evolution of the image feature ξ_2

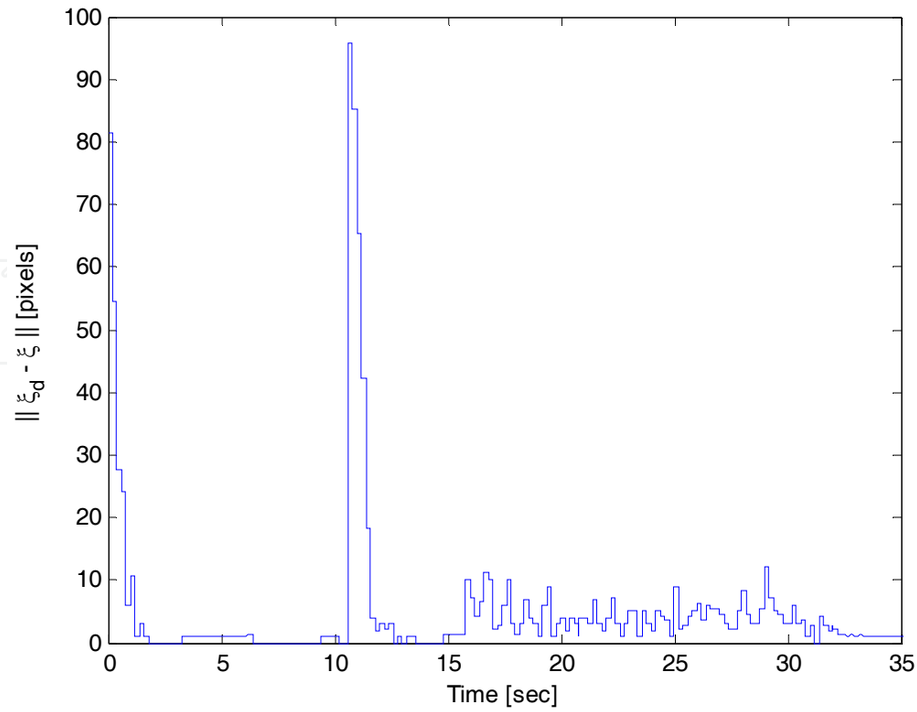


Fig. 18. Time evolution of the image features error norm $\left\|\tilde{\mathbf{s}}\right\|$

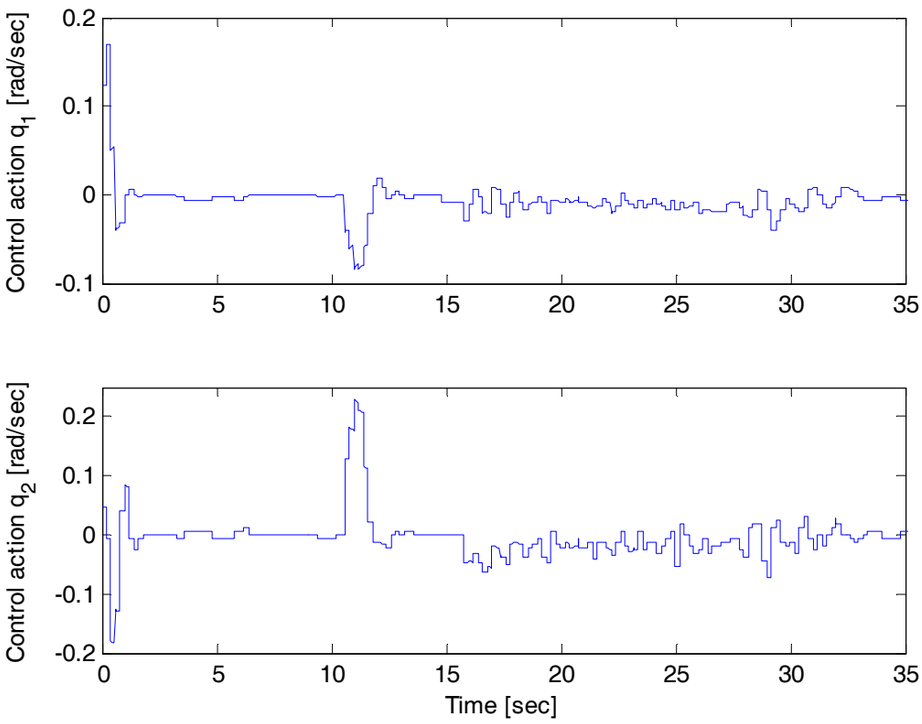


Fig. 19. Control actions for q_1 and q_2

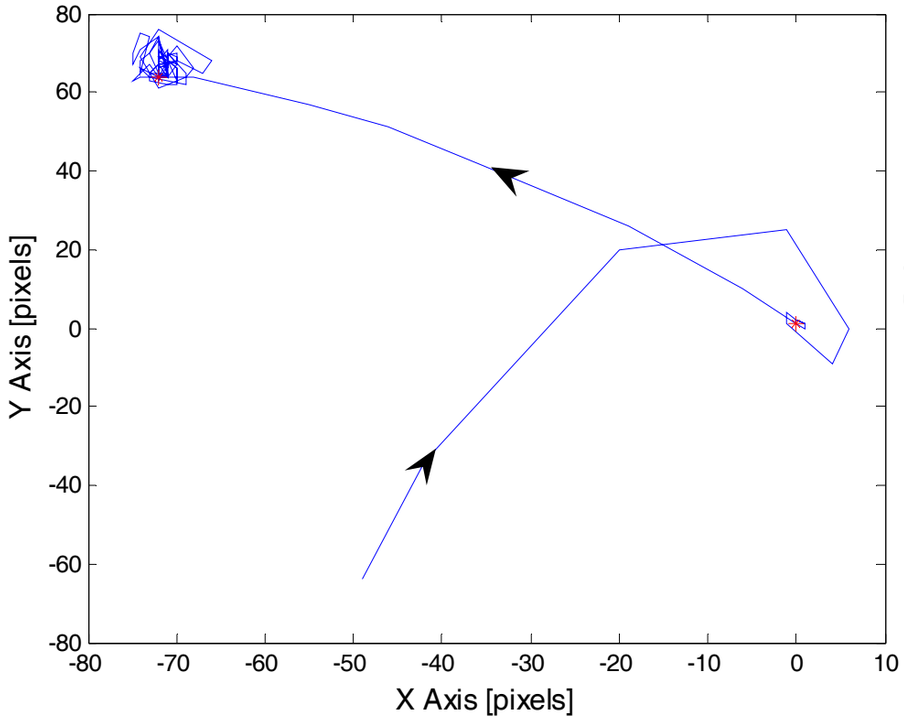


Fig. 20. Image features trajectory on the image plane

5. Conclusions

In this chapter, the design, implementation and experimentation of an open software structure for industrial robot manipulators have been presented. The developed software allows the users to save time and efforts in the implementation and performance evaluation of new control algorithms, as well as in the addition of new hardware components, i.e. sensors or actuators. Therefore, the developed software is useful for research in the field of robotics and human resource training, with potential impact in industry.

The software system has been split into two different programs that communicate each other, clearly dividing different tasks of the control system. This way, a modular reuse based system is obtained. First program (*Critic Time Program*) is responsible for communicating with the sensors and the actuators through the data acquisition and control hardware, updating the sensors' data in the shared memory block, and it is also responsible for synchronization of the two programs. Each one of the hardware devices is handled with a different object, obtaining the desirable encapsulation for the data and methods associated to each device. Second program (*Control Program*) is responsible for running the control algorithm and updating the control actions in the shared memory block.

Additionally, the proposed open software structure has been evaluated with two different control algorithms: first, a classical PD controller using the internal position sensors of the robot; and second, a passivity based visual controller using a vision system placed at the end effector of the robot. Both, the classical PD controller and the visual controller were successfully implemented in the proposed software structure, showing that the main objectives of the work presented in this chapter have been achieved.

6. Acknowledgment

Authors thank to the National Council of Scientific and Technical Research of Argentina (CONICET) for partially supporting this research.

7. References

Boyd, S.; El Ghaoui, L.; Feron, E. and Balakrishnan, V. (1994). *Linear Matrix Inequalities in Systems and Control Theory*, Society for Industrial Mathematics, ISBN: 0-89871-334-X, Philadelphia, PA, USA.

El Ghaoui, L.; Nikoukhah, R. and Delebecque, F. (1995). LMITOOL: a Package for LMI Optimization, *Proceedings IEEE Conference on Decision and Control*, pp. 3096-3101, ISBN: 0-7803-2685-7, New Orleans, LA, USA, December 1995.

Frederick, M. P. and Albus, J. S. (1997). Open architecture controllers, *IEEE Spectrum*, Vol. 34, Nº 6, (June, 1997) 60-64, ISSN: 0018-9235.

Fujita, M.; Kawai, H. and Spong, M. W. (2007). Passivity-based Dynamic Visual Feedback Control for Three Dimensional Target Tracking: Stability and L_2-gain Performance Analysis. *IEEE Transactions on Control Systems Technology*, Vol. 15, Nº 1, (January 2007) 40-52, ISSN: 1063-6536.

Hill, D. and Moylan, P. (1976). Stability results for nonlinear feedback systems. *Automatica*, Vol. 13, Nº 4, (July 1976) 377-382. ISSN: 0005-1098.

Hutchinson, S.; Hager, G. and Corke, P. (1996). A tutorial on visual servo control. IEEE Transactions on Robotics and Automation, Vol. 12, Nº 5, (October 1996) 651-670, ISSN: 1042-296X.

Jang, W. and Bien, Z. (1991). Feature-based visual servoing of an eye-in-hand robot with improved tracking performance, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2254-2260, ISBN: 0-8186-2163-X, Sacramento, USA, April 1991.

Kelly, R.; Carelli, R.; Nasisi, O.; Kuchen, B. and Reyes, F. (2000). Stable Visual Servoing of Camera-in-Hand Robotic Systems. *IEEE Transactions on Mechatronics*, Vol. 5, Nº 1, (March 2000) 39–48, ISSN: 1083-4435.

Khalil, H. K. (2001). *Non-linear Systems*, Prentice-Hall, ISBN: 978-0130673893, New Jersey, USA.

Krten, R. (1999), *Getting Started with QNX Neutrino 2: A Guide for Realtime Programmers*, PARSE Software Devices, ISBN: 978-0968250112, Ottawa, Canada.

Lin, W. (1995). Feedback Stabilization of General Nonlinear Control System: A Passive System Approach. *Systems & Control Letter*, Vol. 25, Nº 1, (May 1995) 41-52, ISSN: 0167-6911.

Ortega, R.; Loria, A.; Kelly, R. and Praly, L. (1995). On passivity based output feedback global stabilization of Euler-Lagrange systems. *International Journal of Robust and Nonlinear Control*, Vol. 5, Nº 4, 313-323, ISSN: 1049-8923.

Ortega, R.; Loria, A.; Nicklasson, P. J. and Sira-Ramirez, H. (1998). *Passivity--based control of Euler-Lagrange systems: Mechanical, Electrical and Electromechanical Applications*, Springer-Verlag, ISBN: 978-1852330163, Berlin.

Sawada, C. and Akira, O. (1997). Open controller architecture OSEC-II: architecture overview and prototype system, *Proceedings of International Conference of Emerging Technologies and Factory Automation*, pp. 543-550, ISBN: 0-7803-4192-9, Los Angeles, CA, USA, September 1997.

Sciavicco, L. and Siciliano, B. (2001). *Modelling and Control of Robot Manipulators*, Springer-Verlag, ISBN: 978-1852332211, London, Great Britain.

Slotine, J and Li, W. (1991). *Applied non linear control*, Prentice-Hall, ISBN: 978-0130408907, New Jersey, USA.

Sommerville, I. (2000). *Software Engineering, Pearson Education*, ISBN: 978-0201398151, USA.

Spong, M. and Vidyasagar, M. (1989). *Robot dynamics and control*, John Wiley & Sons, ISBN: 978-0471612438.

United Nations Economic Commission for Europe (UNECE) and International Federation of Robotics (IFR). (2005). *World Robotics – Statistics, Market Analysis, Forecasts, Case Studies and Probability of Robot Investment*, International Federation of Robotics and United Nations Publication, ISBN: 92-1-1011000-05, Geneva, Switzerland.

van der Schaft, A. (2000), L_2*-Gain and Passivity Techniques in Nonlinear Control*, Springer-Verlag, ISBN: 978-1852330736, London, Great Britain.

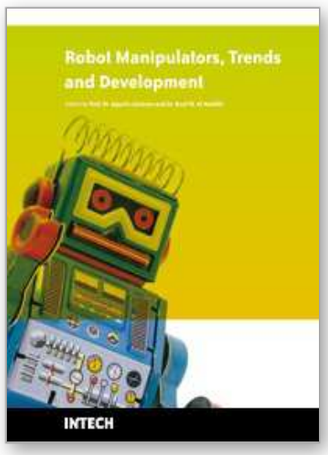
Vidyasagar M. (1979). New passivity-type criteria for large-scale interconnected systems. *IEEE Transactions on Automatic Control*, Vol. 24, Nº 4, (August 1979) 575-579, ISSN: 0018-9286.

Weiss, L. E.; Sanderson, A. and Neuman, P. (1987). Dynamic Sensor-based Control of Robots With Visual Feedback. *IEEE Journal of Robotics and Automation*, Vol. 3, Nº 9, (October 1987) 404-417, ISSN: 0882-4967.

Willems J. C. (1972a). Dissipative dynamical systems part I: General theory. *Archive for Rational Mechanics and Analysis*, Vol. 45, Nº 5, (January 1972) 325-351, ISSN 0003-9527.

Willems J. C. (1972b). Dissipative dynamical systems part II: Linear systems with quadratic supply rates. *Archive for Rational Mechanics and Analysis*, Vol. 45, Nº 5, (January 1972) 352-393, ISSN 0003-9527.

William, E. F. (1994). What is an open architecture robot controller?, *Proceedings of IEEE International Symposium on Intelligent Control*, pp. 27-32, ISBN: 0-7803-1990-7, Columbus, Ohio, USA, August, 1994.



Robot Manipulators Trends and Development

Edited by Agustin Jimenez and Basil M Al Hadithi

ISBN 978-953-307-073-5
Hard cover, 666 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

This book presents the most recent research advances in robot manipulators. It offers a complete survey to the kinematic and dynamic modelling, simulation, computer vision, software engineering, optimization and design of control algorithms applied for robotic systems. It is devoted for a large scale of applications, such as manufacturing, manipulation, medicine and automation. Several control methods are included such as optimal, adaptive, robust, force, fuzzy and neural network control strategies. The trajectory planning is discussed in details for point-to-point and path motions control. The results in obtained in this book are expected to be of great interest for researchers, engineers, scientists and students, in engineering studies and industrial sectors related to robot modelling, design, control, and application. The book also details theoretical, mathematical and practical requirements for mathematicians and control engineers. It surveys recent techniques in modelling, computer simulation and implementation of advanced and intelligent controllers.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen