Java Software Platform for the Development of Advanced Robotic Virtual Laboratories

CARLOS A. JARA, FRANCISCO A. CANDELAS, JORGE POMARES, FERNANDO TORRES

University of Alicante, Department of Physics, System Engineering and Signal Theory, Carretera San Vicente del Raspeig s/n, 03690 Alicante, Spain

Received 16 August 2010; accepted 2 March 2011

ABSTRACT: This article presents an interactive Java software platform which enables any user to easily create advanced virtual laboratories (VLs) for Robotics. This novel tool provides both support for developing applications with full 3D interactive graphical interface and a complete functional framework for modelling and simulation of arbitrary serial-link manipulators. In addition, its software architecture contains a high number of functionalities included as high-level tools, with the advantage of allowing any user to easily develop complex interactive robotic simulations with a minimum of programming. In order to show the features of the platform, the article describes, step-by-step, the implementation methodology of a complete VL for Robotics education using the presented approach. Finally, some educational results about the experience of implementing this approach are reported. © 2011 Wiley Periodicals, Inc. Comput Appl Eng Educ; View this article online at wileyonlinelibrary.com/journal/cae; DOI 10.1002/cae.20542

Keywords: distance teaching; learning environments; Robotics; simulations; virtual laboratories

INTRODUCTION

Because of the explosive growth of the robotic systems in the last few years, engineering students are required to know and learn their rapidly expanding development. However, theoretical lessons do not provide enough knowledge to students, especially in the field of Robotics since this discipline integrates the knowledge of several scientific subjects such as mechanics, electronics and control systems [1]. Education in technical disciplines inevitably involves hands-on experiments, so the laboratory work is also needed to improve student learning with practical issues. Nevertheless, many problems exist in giving students sufficient and appropriate educational resources for Robotics. These include expensive equipment, limited time, and complicated experimental set-ups. As a solution to these problems, virtual laboratories (VLs) represent distributed environments which are intended to perform the interactive simulation based on a mathematical model of a real system [2]. By means of them, students can learn through the Internet in a practical way and thus become aware of physical phenomena that are difficult to explain from just a theoretical point of view. In particular, interactive VLs are effective pedagogical resources in engineering education, well suited for web-based and distance education [3]. Their interactivity allows students to simultaneously visualise the response of the simulation to any change introduced, and this immediate observation of the change is what really helps students to develop useful practical insight into robotic systems theory [4]. Moreover, the interactivity of VLs encourages students to play a more active role in the e-learning process and provides realistic hands-on experiences [5].

Robotics systems have highly complex behaviours. For this reason, with this kind of systems, a reliable mathematical model is indispensable to design a VL which enables users solving real problems by means of simulation. In this way, appropriate software platforms are needed to define the mathematical models and to design useful and comprehensive VL for Robotics. Nowadays, there are several simulation tools devoted to robotic systems. Among them, several graphical software environments designed for professional applications such as RoboWorks [6], Robot Assist [7], Easy-ROB3D [8], RobCAD [9], DELMIA [10] and GRASP [11] have been created in the form of stand-alone business packages for well-defined problems. Other open source packages are in the form of toolboxes such as RoboOp [12] (C++ library), SimMechanics [13], RobotiCad [14] and Robotics Toolbox [15] (Matlab-Simulink libraries). However, these tools are not suitable for developing robotic VLs because, on the one hand, they do not provide a powerful graphical support for creating an interactive graphical interface and, on the other hand, users must have programming skills in order to develop the simulation. In this way, currently science students and educators

Correspondence to C. A. Jara (cajb@dfists.ua.es).

have to spend time and effort for developing a robotic VL. This supposes that up to now, there is not yet a specific tool to create interactive VLs for Robotics.

Currently, there are several interactive and simulation tools created for Robotics education. Among them, it is worth mentioning the applications presented in Refs. [16–18], which allow students to experiment with only kinematics, trajectory planning and programming concepts of different robotic systems. There are other platforms which enable users to experiment with advanced robotic concepts such as dynamics [19]. However, these approaches are specific applications and do not permit to change or develop other robotic systems under the same platform. In contrast, the approach presented in this article provides a complete functional framework for modelling and simulation of any robotic system. The main novel feature of this approach is that its software architecture contains a high number of functionalities in the same platform which are included as high-level tools, with the advantage of allowing users to easily create robotic applications with a minimum of programming. These tools provide several advanced features for serial-link robots. They cover functions for solving problems such as kinematics, trajectory planning, programming, dynamics, object interaction, world modelling, importation of 3D model files and so on. Moreover, the tasks of creating the user interface and developing the virtual environment of the simulation are quite simplified. Therefore, the software platform presented in this article is a specific tool so that any user will be able to develop advanced interactive VLs for Robotics. This represents an approach which has not been implemented before in previous research projects.

Another meaningful problem of the previously commented tools is the platform dependency. Some C++ tools are not portable for all the operating systems. The tool presented in this article is based on Java [20], a modern programming language which is platform independent. User requires only installing the corresponding Java plug-ins in order to execute Java applications. Moreover, Java allows users to share its computational resources in distributed environments.

The article is structured as follows: The second section shows the software architecture of the tool. Third section explains the implementation methodology of a complete VL of a Scara robot using the approach presented in this article. Afterwards, some advanced features are described with an experimental example. Next, in the fifth section, the experience of implementing this approach is explained and some educational results are presented. Finally, the most important conclusions are discussed in the sixth section.

SOFTWARE ARCHITECTURE

There are two main blocks that represent the functional core of the software platform (Fig. 1): an object-oriented Java library (EjsRL from this moment) which allows users to model arbitrary serial-link robots; and Easy Java Simulations (EJS), a powerful application for easily developing simulations with a higher graphical interface capacity [21]. The combination of both software blocks makes possible to easily and quickly create advanced robotic VLs (Fig. 1).

EJS is an open-source application developed in Java, specifically designed for the creation of interactive dynamic VL [22]. EJS has been created for people who do not need complex

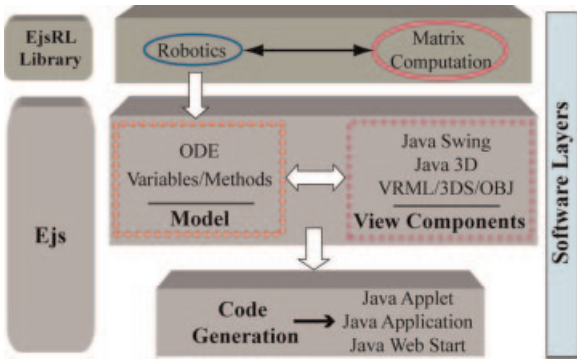

Figure 1 Architecture of the software platform. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

programming skills. Users can easily and quickly create interactive simulations. They only need to provide the most relevant core of the simulation algorithm and EJS automatically generates all the Java code needed to create a complete interactive VL. In addition, EJS creates all the necessary files in order to run the final applications in three different ways: as an applet embedded in a web page, as a stand-alone Java application and as a Java Web Start application.

EjsRL is a Java library specifically designed for EJS, which provides a complete functional framework that enables EJS to model, design and execute advanced Robotics applications within its environment. For a complete description of all the classes and packages, readers can see the online documentation available in the web page dedicated to the platform presented here: www.aurova.ua.es/rcv. There are two important modules which define the high-level API of the EjsRL: Robotics and Matrix Computation. In relation to these modules, their functional structures are based on the following criteria:

- The Matrix Computation classes perform the mathematical computations required for solving all the Robotics algorithms. They cover the fundamental operations of numerical linear algebra, such as matrix and vector arithmetic. Besides, this package incorporates methods to perform the essential mathematical computations for Robotics, for example the axis rotation XYZ and the transformations among Euler, Roll-Pitch-Yaw and Matrix representations.
- The Robotics module covers the fundamental engine for arbitrary serial-link Robotics computation such as kinematics (forward and inverse), trajectory planning and dynamics (forward and inverse). Most of the methods implemented are based on the well-known standard Robotics algorithms. In addition, an interpreter of Java code has been included to allow users the compilation and execution of programming routines for virtual robots in this language.

The software design is based on a hierarchical coordination between EJS and EjsRL. Each of them is divided into subsystems which must communicate and interchange data in order to develop an advanced VL (Fig. 1). In general terms, on the one hand, EJS provides a complete 3D graphical interface support and a model construction tools in order to easily build advanced VLs. On the other hand, EjsRL provides all the

Robotics algorithms which are necessary to simulate the system behaviour within an EJS application.

Within the EJS environment, a specific VL includes the definitions of the model and the graphical interactive user interface (called view) as it is shown in Figure 1. In the model, users can declare different types of variables (int, double, String, Object) in order to describe the real system. Specifically, the Java Object variable can be used as an abstract wrapper to call external objects from other Java libraries. The approach presented in this article utilises this last type to create 'robot' objects from EjsRL to use them into the EJS environment. Moreover, users can define the model by means of a built-in editor of Ordinary Differential Equations (ODEs) to establish how these variables change in time or under user interaction. EJS offers different standard algorithms (Euler, Euler–Richardson, Runge–Kutta, Runge–Kutta–Fehlberg, etc.) to numerically solve these equations. In relation to the view or user interface, EJS provides a set of standard Java components (as Fig. 1 shows) in order to build the interface in a simple drag-and-drop way. Moreover, new features for 3D modelling based on Java 3D capabilities have been added to the last version of EJS in order to improve its interface elements [22]. These features allow users easily and quickly creating very realistic applications and complex VL with sophisticated animations. Among them, it is worth mentioning texture loading or texture mapping, which permits to add surface textures from image files of 3D models, and importation of complex geometric shapes from external graphical files such as VRML (Virtual Reality Modelling Language), OBJ (Wavefront) or 3DS (3D Studio).

All the graphical components of EJS have certain properties that the user can connect with the model variables and set a link between the model and the view. This property can be used to establish a connection between a Java object created with EjsRL, which contains embedded all the Robotics algorithms and its methods, and the graphical components which simulate the system behaviour. Therefore, creating a VL that simulates a robot is greatly simplified.

DEVELOPMENT OF AN ADVANCED ROBOTIC VIRTUAL LAB

This section describes the necessary steps to design and develop a complete robotic VL using the presented platform. It will be also emphasised as these steps are carried out in a simple way. The example application will be composed of an industrial Scara robot with 4 degrees of freedom (DOF) and its virtual workspace. Next, an outline of the implementation methodology of all the Robotics algorithms (kinematics, path planning, dynamics, programming and object interaction) will be explained.

EJS Set-Up

As mentioned in the second section, a VL inside of EJS is composed of two interrelated parts: (1) the simulation of the mathematical model that describes the system; and (2) the interactive user-to-model interface or view. Figure 2 shows these parts, which have to be defined by VLs developers, as well as the area to insert the Java code to create the VL model. The first step in order to create a robotic VL is to insert the library EjsRL as an external resource (Fig. 2). In this way, all the Robotics algorithms of this library can be used within EJS' environment to define the robotic model.

Creating the Virtual Robot and Its Environment

The second step in order to develop a robotic VL is to create a specific robot in the model section of EJS environment. This action implicates to define the variables and to program a robot object specifying a minimum code.

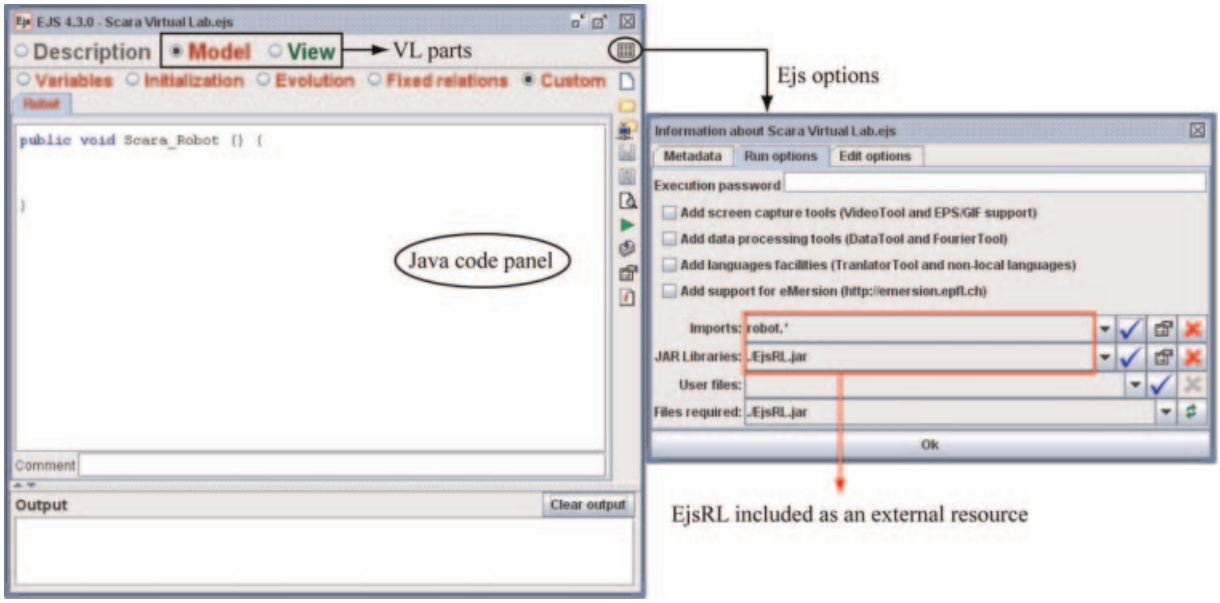

Figure 2 Set-up of EJS' environment. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

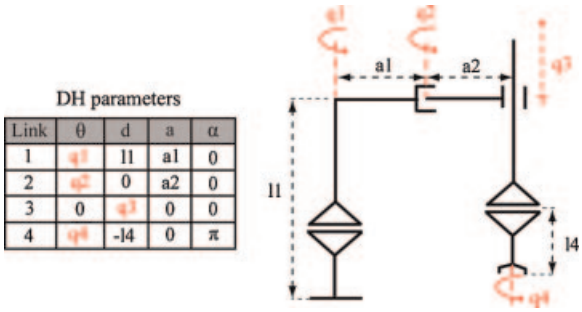

Figure 3 DH parameters and skeleton representation of the Scara robot. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

For creating an arbitrary robot arm object, users only have to know its Denavit–Hartenberg (DH) parameters [23], the type of joints (rotational or prismatic) and its range of motion. The DH parameters of the system proposed with the skeleton representation of the robot manipulator can be seen in Figure 3.

A Java object variable defined in EJS has to be initialised using the robot constructor of the Robotics module of EjsRL. Figure 4 shows the Java code which must be inserted in the model part of EJS (Fig. 2, 'Java code panel') and the variables that must be created to program an industrial Scara robot arm of 4 DOF. The parameter *type* indicates the kind of each joint (rotational 'R', prismatic 'P') for the robot constructor. This Java code creates a robot object which includes the needed functions or methods to implement all the Robotics algorithms (kinematics, path planning, dynamics, programming and object interaction).

Once the robot object has been programmed, VL developers can build the graphical interface or view of the virtual robot and its environment. As stated, the new 3D framework of EJS provides a wide variety of different elements, based on the Java 3D classes [22]. Users can drag any of these view elements into the view panel, in order to build the simulation view. The result is a tree-like structure as the one shown in Figure 5, which is the view containing the user interface of the robot arm and its workspace. The component *drawingPanel3D* is the 3D environment where the robot and its workspace will be displayed in the simulation. Here, the 3D links of the manipulator are defined by means of 3DS components, which allow users to import models from existent 3DS files (see Fig. 5). As mentioned before, all the interface components of EJS have certain properties which are used for the VL's simulation. Figure 5 also shows the properties of the 3DS component. The *Position* and *Transform* fields will be used to move the robot since they will be connected with the results which the robot object computes.

In the example considered, the main view elements which compose the 3D robot workspace are cubes with texture mapping and VRML models. In order to give a realistic appearance to the elements, it is only required to edit their properties and to associate the file path of the images and the 3D models, respectively. These properties allow users to develop high-performance 3D virtual robotic environments.

Kinematics Simulation

This section explains how users can easily implement the kinematics simulation of an arbitrary serial link manipulator of n DOF by means of the software platform presented in this article.

Robot kinematics deals with the analytical study of the motion of a manipulator. There are two well-known problems: the forward and inverse kinematics problems [24]. The Robotics classes of EjsRL implement both algorithms in order to give motion to the 3D links of a specific manipulator defined in an EJS view.

The implementation of the forward kinematics problem is based on a sequential multiplication of the homogeneous transformations ${}^0A_1 \ldots {}^{n-1}A_n$ that describe the spatial relation between the joint values and the spatial location of the end effector T [24]:

$$T(q_1, \ldots, q_n) = {}^0A_1(q_1) \cdot {}^1A_2(q_2), \ldots, {}^{n-1}A_n(q_n) \qquad (1)$$

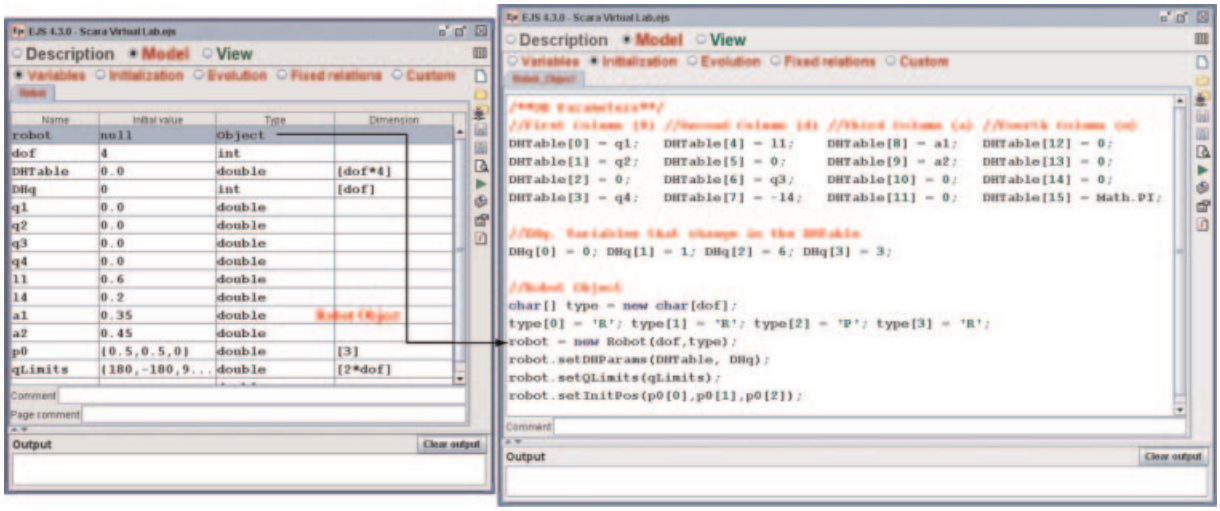

Figure 4 Java code to be included in the VL model in order to create a 4 DOF Scara robot. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

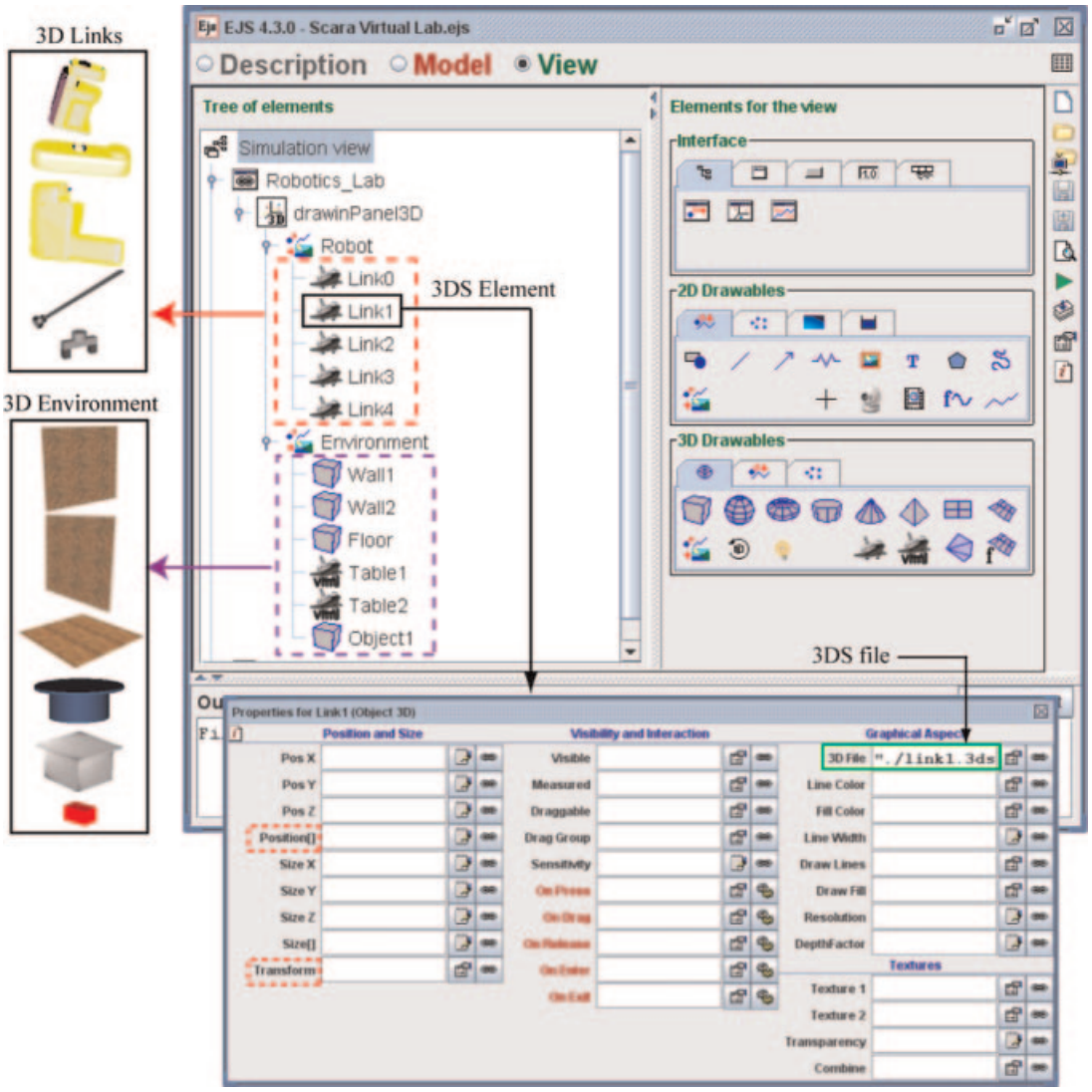

Figure 5 Design of the view interface of the robot arm proposed and its workspace. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Figure 6 shows the Java code to solve the forward kinematics computation of the Scara robot. The joint values are obtained from an interactive user interface developed in the view by means standard Java Swing components such as a dialog panel and sliders (see Fig. 6). The joint variables *q1*, . . . ,*q4* are connected with the sliders' elements which call the *forwardKinematics* method for updating the *DHTable* array of the model. Afterwards, the homogeneous transformations of each link are computed using the function *FKinematics* of the Robotics module of EjsRL. These matrix objects (*A01*, . . . ,*A04*) are prepared to directly be inserted in the property *Transform* (see Fig. 5) of the 3DS view elements to move them according to this robotic algorithm.

The library EjsRL implements a numerical method for solving the inverse kinematics problem. This algorithm is an iterative procedure where the initial position affects both the search time and the solution found. In addition, some solutions could not be possible if the joint values computed describe an end-point out of reach of the manipulator. Figure 7 shows the Java code for programming the inverse kinematics of the proposed robot. The method *IKinematics* receives the current joint values (variable *q_current*), and the position and orientation of the end effector from an interactive user interface where Cartesian variables *X*, *Y*, *Z*, Roll are connected with the sliders' elements. The method also checks that the solution proposed is inside the area of reach of the manipulator using the values of the variable *qLimits* (see Creating the Virtual Robot and Its Environment Section, Fig. 4). Finally, the robot is moved to the suitable position using the function for the forward kinematics explained before.

Thereby, the Robotics classes of EjsRL provide several methods to compute the inverse kinematics in closed form of some standard robots to obtain a best response time. EjsRL supports the inverse kinematics solution of some standard robots such as the 3R (3 rotational DOF), the 6R (6 rotational DOF), the Scorbot robot and the Scara robot.

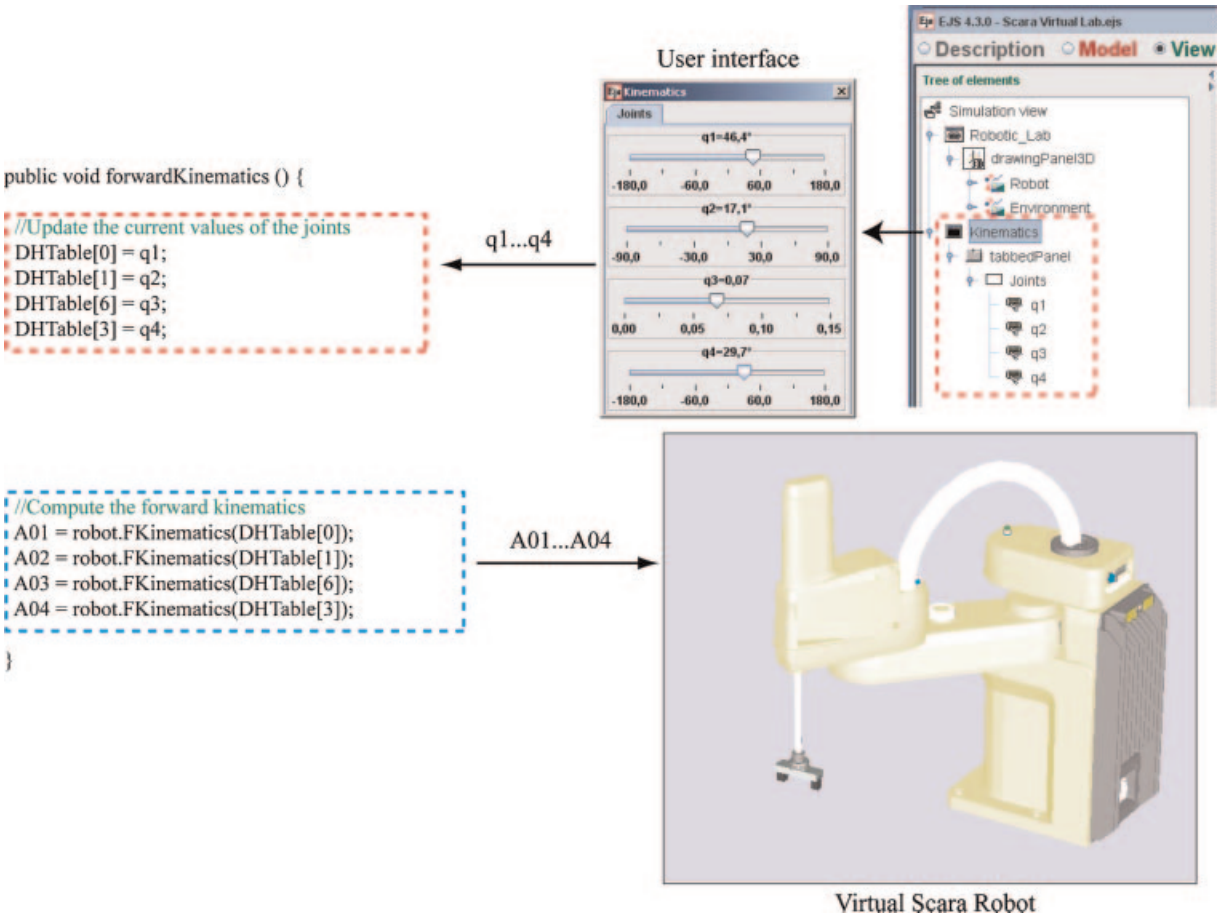

Figure 6 Implementation of the forward kinematics simulation for the Scara robot. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Path Planning Algorithms

The path planning problem consists of generating a time sequence of the values attained by a polynomial function interpolating the desired trajectory. The presented tool allows users to easily perform the simulation of path planning trajectories for *n-axis* robot arms. On the one hand, the built-in ODEs editor implemented in EJS is employed to generate the position and velocity. This feature allows users to write differential equations to model the robot movement. On the other hand, the Robotics classes of EjsRL contain a path planning construction module which computes the acceleration parameters of several trajectories from their imposed constrains. This module implements a lot of methods for the simulation of the robot motion such as splines (third, fourth and fifth order), cubic interpolators, synchronous, asynchronous and linear trajectories.

In this way, two simple steps are necessary in order to implement the simulation of a path planning algorithm in the VL proposed: (1) To write the differential equations of the basic motion of a multi-body system. Figure 8 shows these equations in the ODEs editor of EJS, which compute the sequence values of the position (q) and velocity (vpp) of all the robot joints from the acceleration of the trajectory (app); (2) To compute the acceleration of the path planning algorithm proposed using one of the functions provided by the Robotics module. The trajectory planning module returns the acceleration parameters of several trajectories which will be used in the motion equations. Figure 8 shows the Java code to program a 4-3-4 polynomial interpolator [24] in the VL proposed. This code computes the acceleration array for the differential equations of motion (app).

The generated joint values are given to the kinematics model to simulate the robot movement. Figure 9 shows the movement of this path planning algorithm in the Scara robot. Several plot controls of EJS can be used to visualise the more interesting variables of the trajectory (Fig. 9).

Dynamics Solution

Robot dynamics is concerned with the equations of motion for *n-axis* robot manipulators which reflect the relationship between the forces acting on a robot mechanism and the accelerations produced [24]:

$$\tau = M(q) \cdot \ddot{q} + C(q, \dot{q}) + G(q) \tag{2}$$

In this formulation, M is the symmetric joint-space inertia matrix, C describes Coriolis and centripetal effects and G is the gravity loading. The solution to this equation is used to determine the torques in a particular spatial motion, that is the

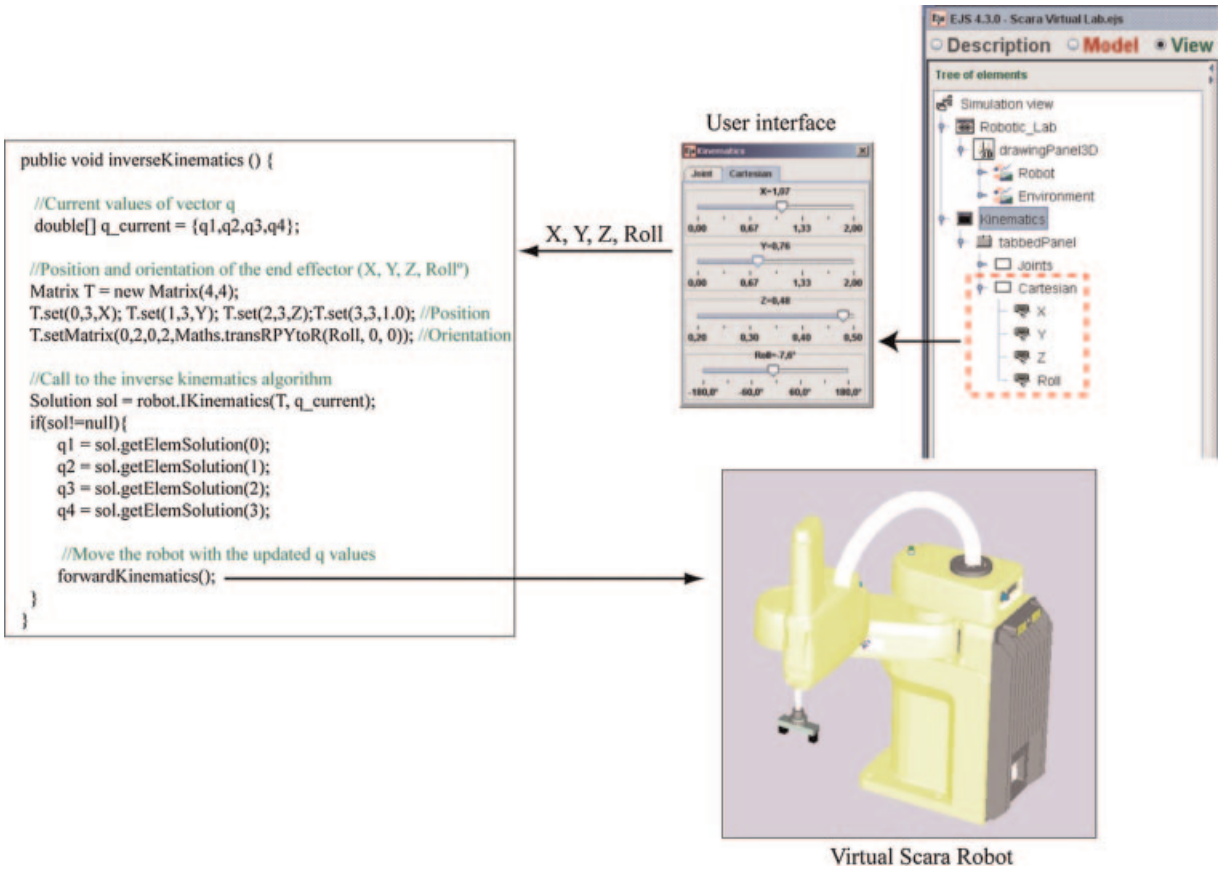

Figure 7 Implementation of the inverse kinematics simulation for the Scara robot. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

inverse dynamics problem, and to find the spatial motion produced by a particular set of torques or the forward dynamics problem. The Robotics module of EjsRL implements classes and numerical methods to solve both dynamics problems.

Figure 10 shows the Java code to implement the inverse dynamics simulation with an external force (object weight) in the proposed robotic VL. This method utilises the recursive Newton–Euler algorithm [24] for computing the torque for each joint, given the velocity and acceleration of the robot in the joint space. Mass, inertias and viscous friction properties must be defined as variables in order to solve this robotic algorithm (Fig. 10). The array of variables *vpp* and *app* belong to the velocity and acceleration of the path planning (see Path Planning Algorithms Section).

The Robotics package also provides methods to compute separately the matrices M, C and G of the equation of motion (2). Thus, users can obtain, in real-time, all the values of the dynamics parameters involved in the equation. In Figure 11, it can be seen how both these matrices and the torque joint value change during the robot movement.

Virtual Programming

The Robotics package incorporates high-level functions for virtual programming issues which permit final users of the VL to program, compile and execute complex tasks in the robotic VL. The actions are described using the Java language, whereby final users can create variables, mathematical operations and objects in this language. The programming module implements specific classes and methods for both robot simulation and object interaction. For a detailed description of the available classes and methods see Table 1. The Robotics module of EjsRL provides a function which calls the Java compiler and generates the binary code that is executed in the virtual robotic world. Thus, the compilation and execution of this Java code only requires the Java Development Kit (see Appendix A).

Figure 12 shows the implementation of virtual programming features in the VL proposed. It is only necessary to create an interactive user interface in the EJS view (like a dialog panel including a text field and a button), to insert Java code, and to carry out the compilation and execution of this Java code by using the functions provided by the Robotics module of EjsRL (see Fig. 12).

The common structure that users must follow to program a specific task is the following:

- Declaration of the positions (joint or Cartesian) and time in form of variables and arrays of the 'double' type.
- Creation of the positions objects by means of classes *posJ* and *posC*. The parameters for these objects are the variables and arrays initialised before.
- Insertion of the virtual objects located in the workspace with the same name given in the EJS view.

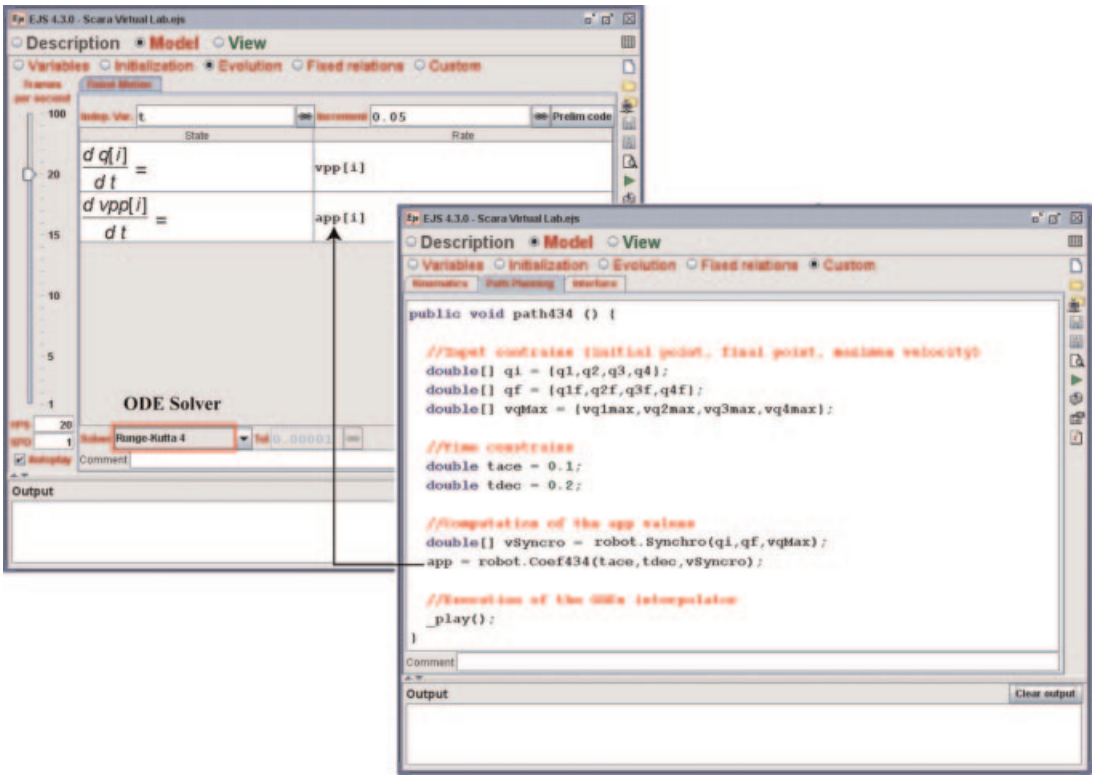

Figure 8 ODEs of basic robot motion and Java code to program a 4-3-4 polynomial trajectory. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

- Declaration of the movement commands. Users can employ the methods *moveJ* and *moveC* for joint and Cartesian movements. The parameters for these methods are the objects *posJ* and *posC* created before.

Figure 13 shows a programming experiment which consists of doing a pick-and-place operation of a virtual object located in the table. As mentioned before, there are four different parts in the program's structure: (1) declarations of the positions; (2)

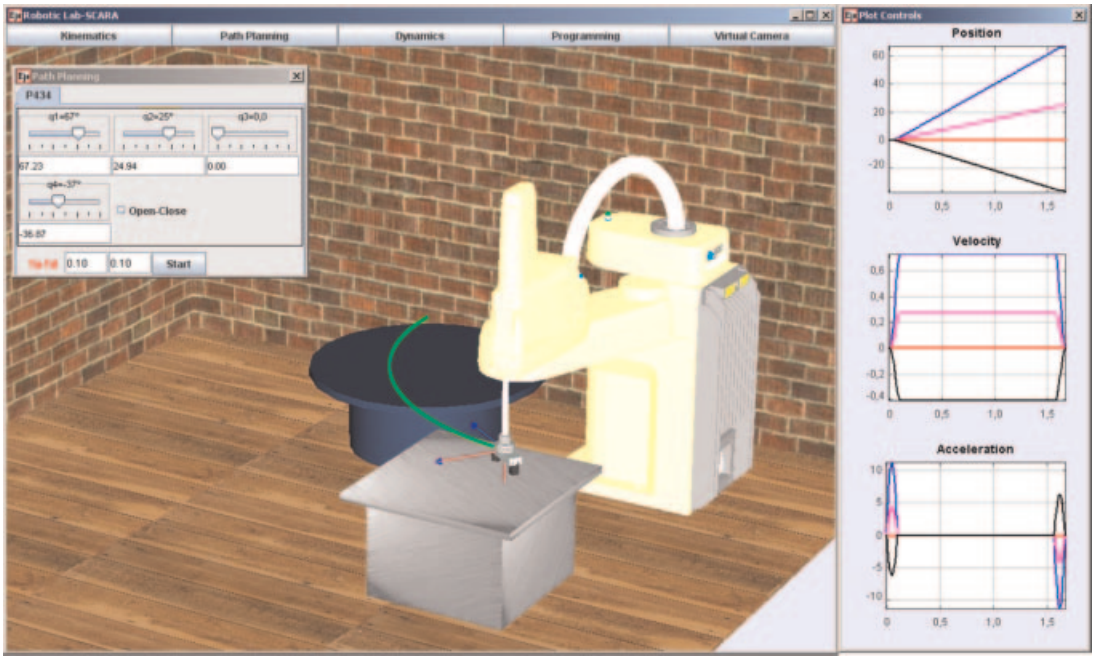

Figure 9 Simulation of the 4-3-4 polynomial path planning algorithm in the VL proposed. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

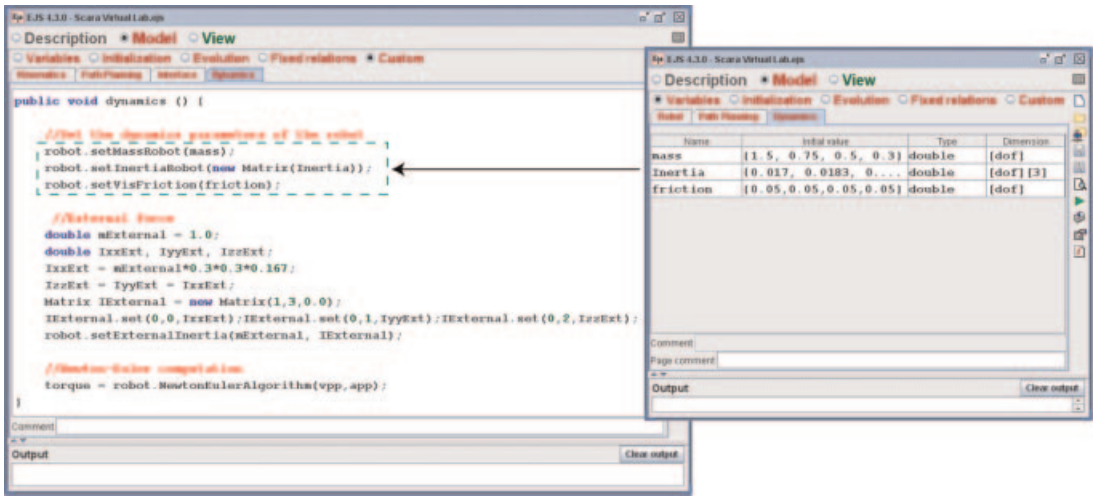

Figure 10 Implementation of the inverse dynamics simulation with an external force. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

creation of the objects *posJ*; (3) insertion of the virtual objects; and (4) definition of the order movements. In these last methods, users can specify the trajectory that they want to use. In the example proposed, the task is performed by means of synchronous trajectories (parameter 'Syn' in the method *moveJ*). Figure 13 also shows the states of the virtual robot during the execution of the pick-and-place experiment. The image sequence represents each of the joint positions programmed in the code.

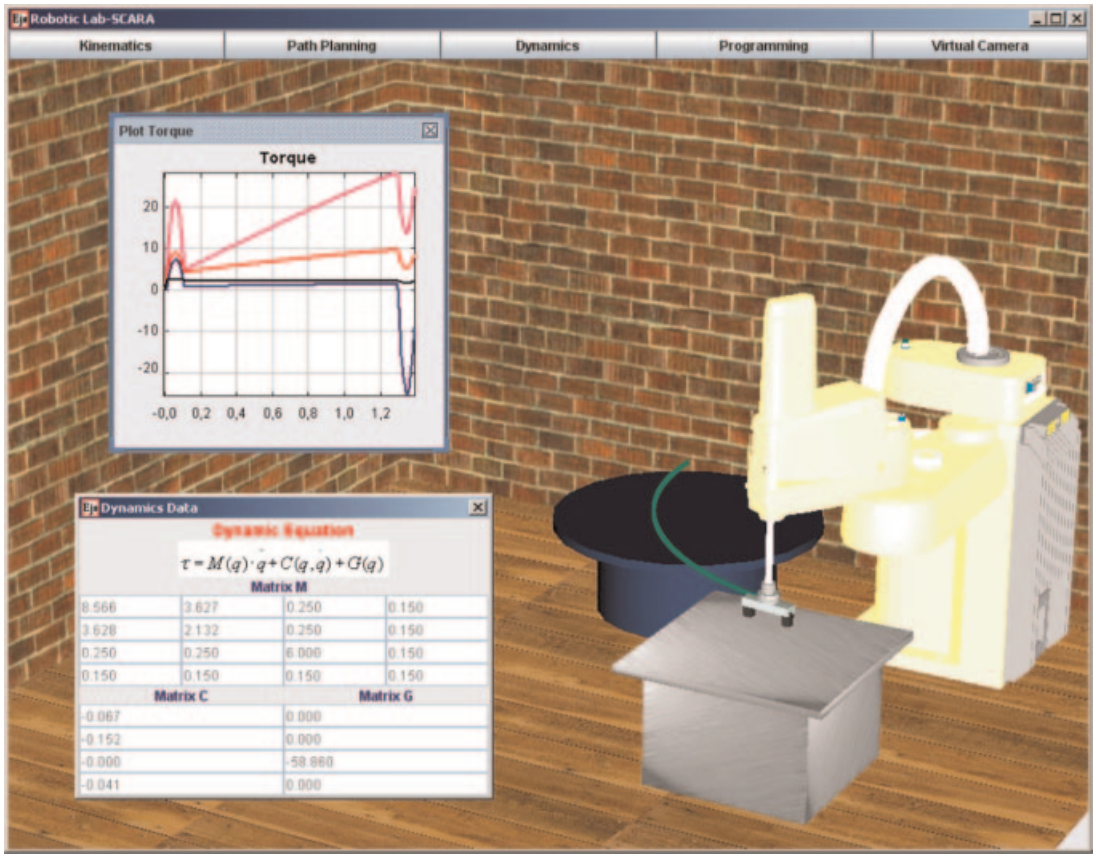

Figure 11 Simulation of the robot dynamics. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Table 1 Classes and Methods of the Programming Module of the Robotics Package

Classes/methods	Description
posJ (double[] position)	Class for define joint positions *Input parameter*: array of joint values
posC (double[] position)	Class for define Cartesian positions *Input parameter*: array of Cartesian values
moveJ (String traj, posJ p, double time)	Method for robot joint movements *Input parameters*: type of trajectory, position, time
moveC (String traj, posC p, double time)	Method for robot Cartesian movements *Input parameters*: type of trajectory, position, time
insertObj (String name)	Method to insert the objects of the virtual workspace *Input parameter*: object name
double[] getPosObj(String name)	Method for getting the position of an object *Input parameter*: object name *Output parameter*: array of object position
double[] getPosRobot()	Method for getting the robot position *Output parameter*: array robot position
int getNumberObj()	Method for getting the number of objects of the environment *Output parameter*: number of objects

ADVANCED FEATURES

The Robotics package of EjsRL contains some functions for the development of image processing algorithms within EJS' environment. These functions implement image feature extraction algorithms in order to add sensor features in the robotic VL.

In order to show this advanced property, authors have implemented a computer vision algorithm in the virtual robotic environment previously created. The aim is to perform an eye-in-hand (EIH) visual servoing, a complex robotic algorithm. A visual servoing is an approach to guide a robot by using visual information. The two main types of visual servoing techniques are position-based and image-based [25]. The first one uses 3-D visually derived information when making motion control decisions. The second one performs the task by using information obtained directly from the image. In this section an image-based visual servoing system is implemented using EJS and EjsRL.

A visual servoing task can be described by an image function, $\mathbf{e}_t$, which must be regulated to 0:

$$e_t = s - s^* \tag{3}$$

where $\mathbf{s} = (f_1, f_2, \ldots f_M)$ is a *Mx1* vector containing M visual features observed at the current state, while $\mathbf{s}^* = (f_1{}^*, f_2{}^*, \ldots f_M{}^*)$ denotes the visual features values at the desired state, that is the image features observed at the desired robot location. In Figure 14a, the eye-in-hand camera system is shown within the virtual robotic environment where a virtual camera is located at the end-effector of the robot. In Figure 14b, an example of a visual servoing task is represented. This image shows the initial and desired image features from the camera point of view.

In a visual-servo control algorithm, the interaction matrix, $\mathbf{L}_s$, relates the variations in the image $(\dot{s})$ with the variation in the velocity of the camera respect to the scene $(\dot{r})$, that is:

$$\dot{s} = L_s \cdot \dot{r} \tag{4}$$

By imposing an exponential decrease of $e_t\,(\dot{e}_t = -\lambda_1 e_t)$, where $\boldsymbol{\lambda_1}$ is a positive control gain, it is possible to obtain the following control action for a classical image-based visual servoing:

$$v_c = -\lambda_1 \hat{L}_s^{+}(s - s^*) \tag{5}$$

where $\hat{L}_s^{+}$ is an approximation of the pseudo-inverse of the interaction matrix $\mathbf{L}_s$ [25].

In order to implement this advanced experiment, first it is necessary to obtain a view projection from the end effector of the robot. For that end, EJS has an option which allows users to obtain the projection of the virtual camera included in the 3D robotic environment. Figure 15 shows the projection of the EIH virtual camera in the window 'Virtual Camera'. Secondly, this projection must be processed in order to extract the corner features of the object. Figure 15 also shows the Java code which computes corner detection in the virtual camera's image using the SUSAN algorithm [26], a method implemented in the EjsRL library. The classes *ImageObject* and *ImageFunction* belong to EjsRL library and they are used to develop the necessary image processing algorithms. In the window 'Virtual Image', it can be seen the point features detected of the virtual object (see Fig. 15).

Figure 16 shows the resulting final application, which implements a visual servoing algorithm that uses points as visual features. In the upper part of Figure 16, the window 'Virtual Camera' projects the virtual workspace of the EIH virtual camera. In the window 'Virtual Image', it can be seen the corner features detected using the SUSAN algorithm. The application also shows the 3D trajectory performed by the virtual robot to reach the final desired features $\mathbf{s}^*$ (see Eqs. 3 and 5). Finally, the evolution of the velocity module is shown in order to validate the correct convergence of the visual servo task since the velocity described in (5) must be 0 at the end of the trajectory (the image error (3) is regulated to 0).

EDUCATIONAL USE OF THE SOFTWARE PLATFORM

Developing a VL helps students connect the mathematical model to the real system which they are trying to simulate [2]. In addition, designing and performing simulation experiments one-self magnifies student confidence in managing modelling techniques and makes them active players in the learning process, which increases motivation to further learning [4]. For

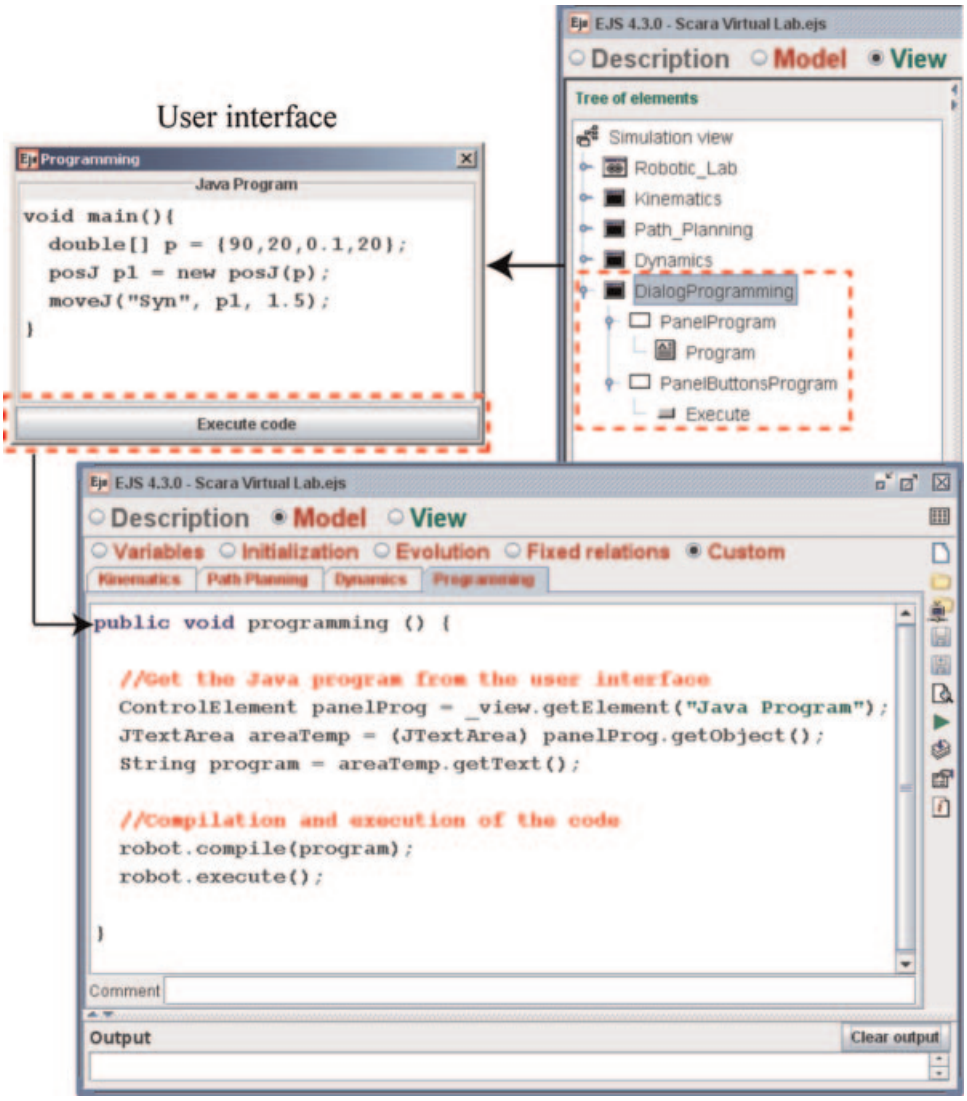

Figure 12 Implementation of virtual programming features in the VL. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

these reasons, authors of this article have been used this approach to allow both undergraduate/postgraduate students and teachers developing their own robotic VLs.

Currently, the software platform presented is being used in the course *Robots and Sensorial Systems* in the Computer Science Engineering degree at the University of Alicante. This course is organised so that robotic concepts are presented in the theoretical lessons in a parallel manner to the creation of a robotic VL by using EJS and EjsRL to make these concepts more concrete. Moreover, the software platform is being used by Computer Science students for their Final Project to fulfil their major requirements, by PhD students for their research experiments and by teachers for developing learning material.

Course Organisation

Computer science engineers typically have no background in Robotics. As commented, to solve this handicap, the course is structured so that students develop the robotic VL in a parallel way that they learn Robotics theory. In particular, the theoretical classes of the courses are organised as follows:

(1) Mathematical tools for spatial transformation in Robotics.
(2) Manipulator kinematics.
(3) Robot dynamics.
(4) Robot programming languages.

The practical experiments employ the software platform presented here. The educational methodology proposed for practises is based on the following criteria:

(1) Students have to choose a specific real robotic system and to compute its DH parameters. Moreover, the 3D links of the robot must be modelled by means of CAD

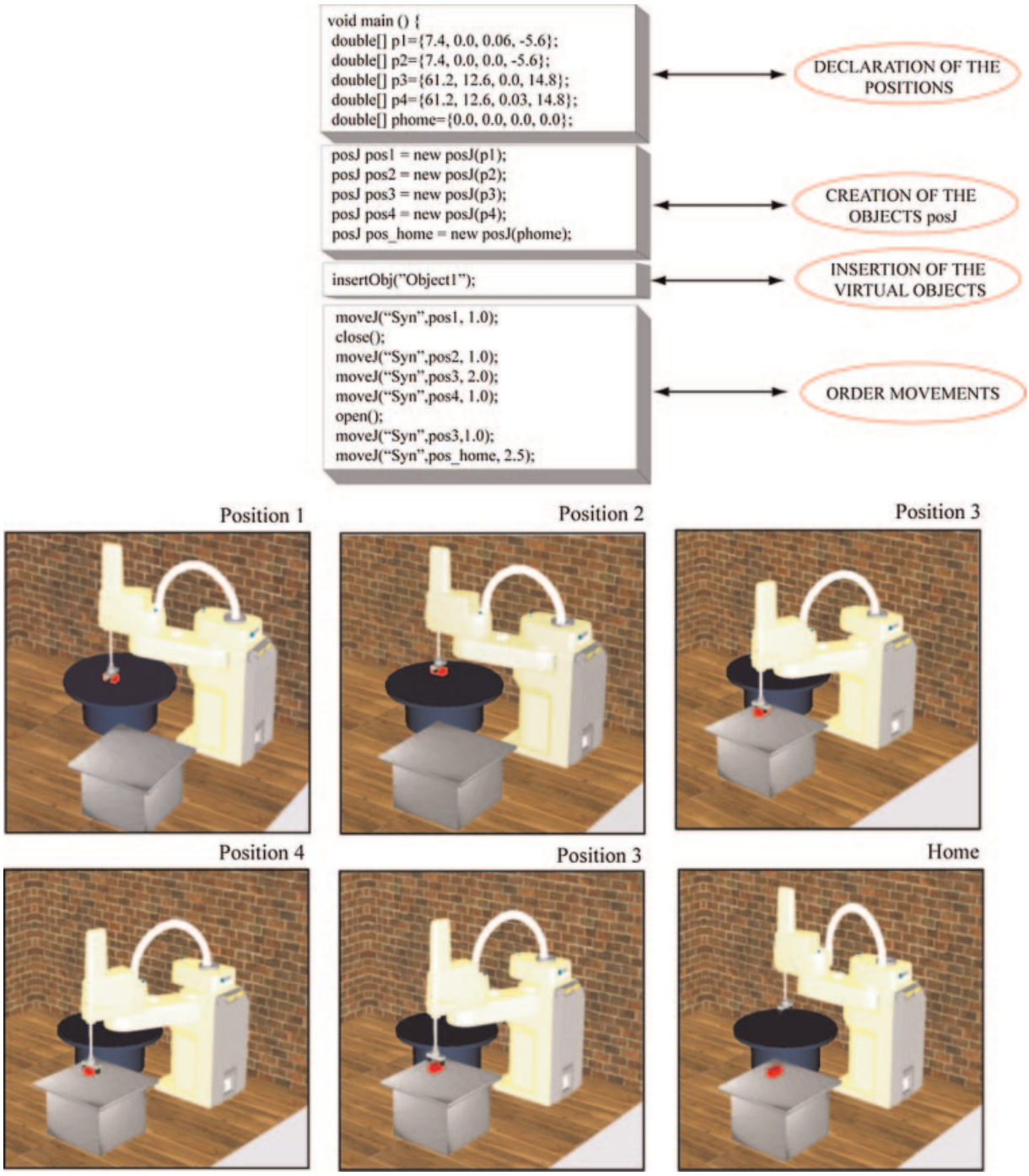

Figure 13 Java code of the pick-and-place experiment and states of the virtual robot during the execution of this programming experiment. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

design software. These 3D solids are exported to VRML/3DS files for the EJS' environment.

(2) Students are required to include the kinematics simulation in the VL.
(3) Students must develop several path planning algorithms for the robot chosen.
(4) Students must implement dynamics and programming features in the VL.

Evaluation of the Teaching Technique and Educational Results

In order to evaluate the teaching technique, a study was carried out in order to verify its usefulness and efficiency. Questionnaire items focused on two main issues:

(1) Perception of the students about if they have learnt Robotics in the course and the degree of agreement

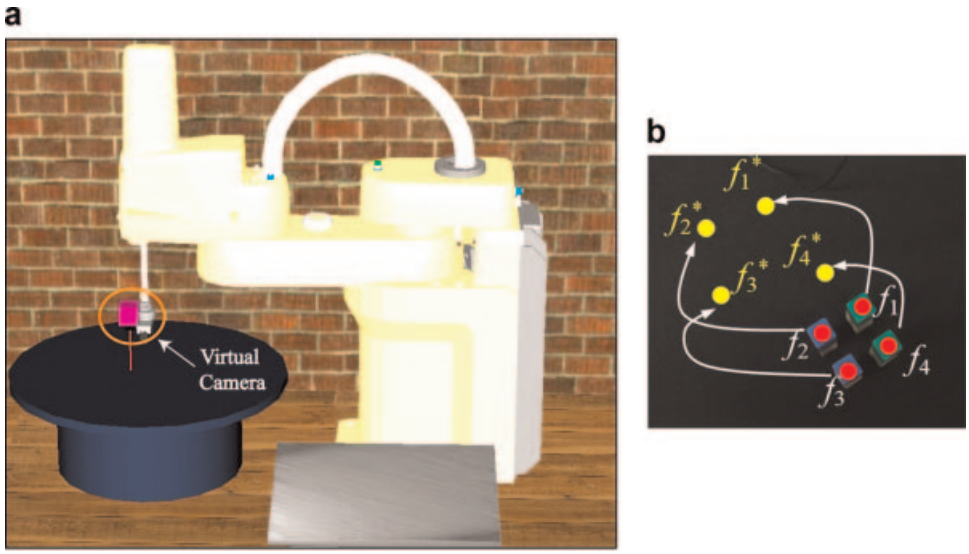

Figure 14 (a) Eye-in-hand configuration. (b) Current and desired visual features in an image based visual servoing task. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

with some statements about the effectiveness of the course (I1).

(2) The development of their own VLs in the learning process using EJS and EjsRL (I2).

Table 2 shows both the questionnaire made and the results obtained over 30 students from the course. Responses were rated on a five-point Likert scale ranking from strongly agree (1) to strongly disagree (5). Analysing the results, more than half of the students think that they have learnt more in this course than in an another course about Robotics. In addition, they recommended the course to other Computer Science students and the homework was helpful (issue I1). Moreover, almost all of students think that the VL is an efficient tool in the learning process and designing his/her own robotic VL using EJS and EjsRL has been helpful in understanding the robotic concepts and to encourage them to learn Robotics (issue I2).

In addition to the survey, a comparison between the final marks of the students who coursed the subject using the software tool presented in this article (academic year 2010) and the student who did not (academic year 2009), was performed. As can be seen in Figure 17, there is a high number of students who got better marks with the use of the software platform: 20% more with A qualification and 10% more with B qualification. The global qualification between 2 years increased from 7.1 to 8.3.

Finally, in the questionnaire performed, students were encouraged to write comments regarding the course. Most of the comments were such as: 'I liked the emphasis on the simulation results rather than the mathematical model of the robotic system' or 'This course has provided me with a lot of insight

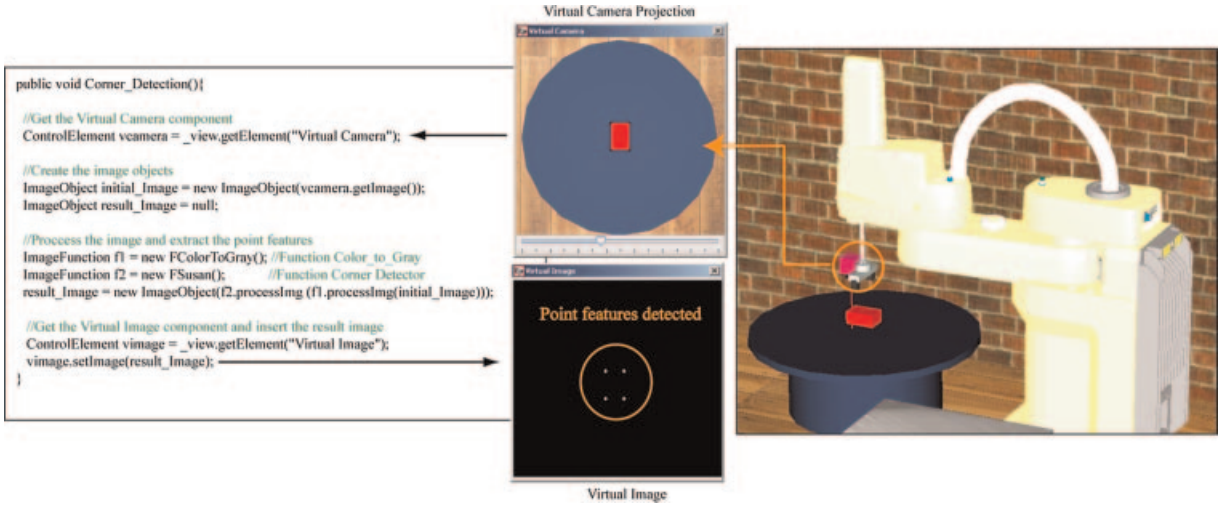

Figure 15 Implementation of the point detection of the EIH virtual camera projection. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

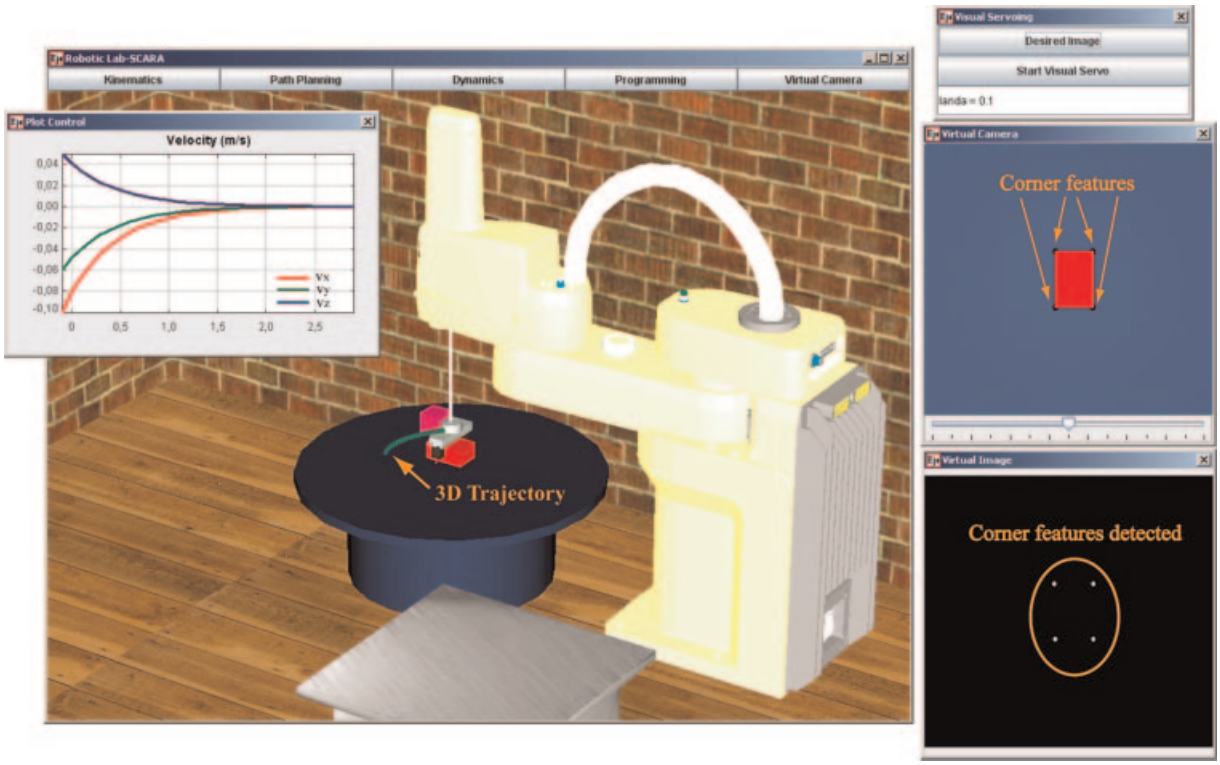

Figure 16 Simulation developed of a visual servoing task using point features. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

into how to model and design a robotic VL' or 'I recommend this course to other Computer Science students who are interested in Robotics'.

Some Final VLs Developed

Computer Science students were able to develop advanced VLs of real robotic systems such as a Barrett Hand [27] (Fig. 18a), a Scorbot RX of 5 DOF (Fig. 18b) and a Mitsubishi PA-10 of 6 DOF (Fig. 18c). A PhD student developed a virtual multi-robot system composed of two manipulators, since EjsRL allows users the instantiation of different robot objects in the EJS' environment (Fig. 18e). Finally, a teacher was able to create a visual servo control of redundant robot of 7 DOF (Fig. 18d). These complex VLs can be seen at www.aurova.ua.es/rcv.

CONCLUSIONS

In this article, a free Java-based software platform for the creation of advanced robotic and computer vision applications has been presented. This new tool is composed of two blocks:

Table 2 Student Questionnaire and Results Obtained in Percentage of Agreement

Question	Issue	% Strongly agree	% Agree	% Neutral	% Disagree	% Strongly disagree
Have you learnt more in the course than in another course about Robotics you have taken?	I1	50	30	15	5	0
Do you think that homework has been helpful in understanding the material?	I1	62.5	20	15	2.5	0
Do you recommend the course to other Computer Science students?	I1	57.5	30	12.5	0	0
Do you feel that you can use robotic concepts learnt in practical situations?	I1	42.5	30	22.5	5	0
Do you think that the VLs help you to understand concepts about robotic systems?	I2	55	32.5	12.5	0	0
Do you think designing your own robotic VL have been helpful in understanding the mathematical model of a robot manipulator?	I2	62.5	25	12.5	0	0
Have you encouraged learning more about Robotics the development of your own VL using Ejs and EjsRL?	I2	57.5	30	7.5	5	0

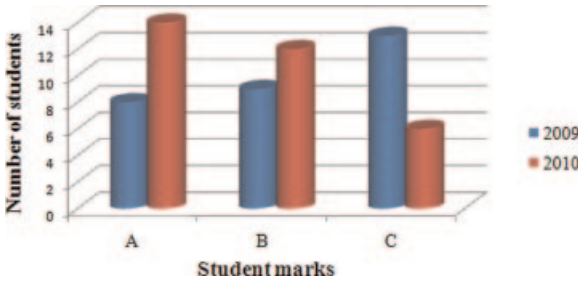

Figure 17 Comparison of students marks between the academic years 2009 and 2010. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

(1) a high-level Java library called EjsRL, which provides a complete functional framework for modelling and simulation of robotic systems; (2) EJS, an open-source tool which provides full graphical interface support. The integration of both parts permits to create complex and advanced robotic applications in an easy way.

EjsRL allows users to model complex robotic systems within EJS. For modelling arbitrary serial-link manipulators, users only have to specify their DH parameters and their physical properties. Then, the Java platform creates an object which has embedded all the robot behaviour (kinematics, dynamics, programming, etc.). The user-friendly interface of EJS enables users to easily and quickly create advanced Java applications. In addition to its full computer graphical support, this software provides several advanced features such as VRML, 3DS and OBJ extern file importation. All these features are complemented with a wide range of objects and operations for Robotics provided by EjsRL. Thus, the approach presented is very suitable to develop research and educational applications in the Robotics field apart from adding novel features that are not found in other toolboxes available today.

The article has shown the simple steps to design and develop a complete VL for Robotics education. The implementation methodology of all the Robotics algorithms has been described as an easy development which can be done by any user. In addition, a teaching methodology was carried on using the presented approach. This educational framework required that students develop their own robotic VLs to help them learn and retain concepts about Robotics. The results obtained show that students think that designing his/her own robotic VL by using EJS and EjsRL has been helpful in understanding the robotic concepts and to encourage them to learn Robotics. Finally, it is important to remark that different users (students and teachers) were able to develop high-level applications which illustrate the power of the proposed software platform.

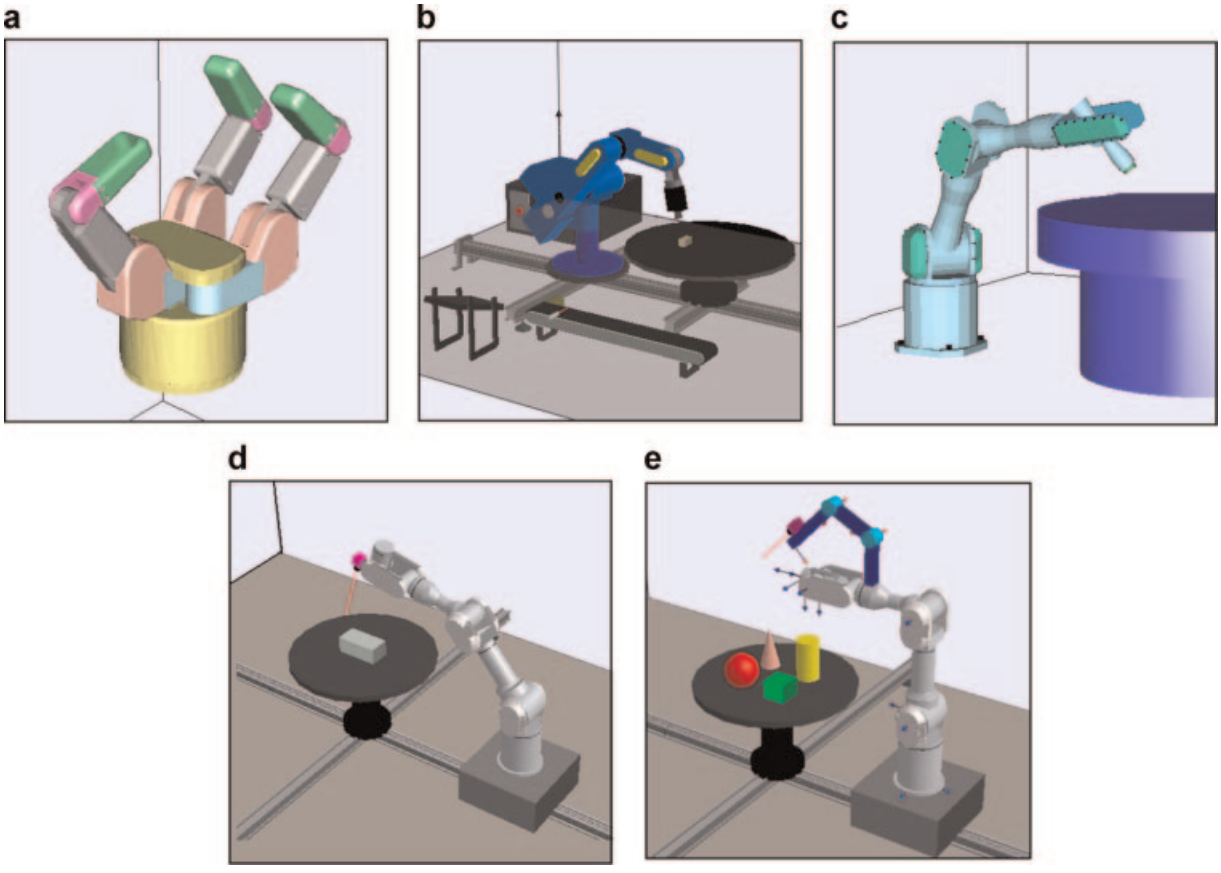

Figure 18 Advanced robotic VLs developed by the students and teachers: (a) the Barrett Hand simulation; (b) a Scorbot RX of 5 DOF; (c) a Mitsubishi PA-10 of 6 DOF; (d) a visual servo control of a redundant 7 DOF robot manipulator; (e) a multi-robot system. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Currently, the library EjsRL is being improved adding remote operation methods to allow users the communication with remote devices using the HTTP and TCP protocols. In this way, this new package of EjsRL could be used for several remote operations such as the teleoperation of real robotics laboratories.

ACKNOWLEDGMENTS

This work is supported by the Spanish Ministry of Education and Science through the research project DPI2008-02647 02647 as well as the 'Technology & Educational Innovation Vicepresident Office' of the University of Alicante through the aid of 'Technologic & Educative Research Groups (GITE)'.

APPENDIX A

EjsRL and EJS

EjsRL can be obtained from www.aurova.ua.es/rcv. In this web page there is a lot of information about how to use the library and readers can execute all the examples shown in the article. EJS can be downloaded from http://fem.um.es/EjsWiki/ where readers can experiment with a lot of simulations developed with this software. In order to use them in any operative system, it is necessary to install two Java environments: JDK (Java Development Kit) 1.6 and Java 3D 1.5 or higher, which are available for many platforms including MS Windows and Unix-based systems.

REFERENCES

[1] V. Potkonjak, M. Vukobratović, K. Jovanović, and M. Medenica, Virtual mechatronic/robotic laboratory—A step further in distance learning, Comput Educ 55 (2010), 465–475.
[2] S. Dormido, Control learning: Present and future, Annu Rev Control 28 (2004), 115–136.
[3] C. Martín-Villalba, A. Urquía, and S. Dormido, Development of an industrial boiler virtual-lab for control education using Modelica, Comput Appl Eng Educ, Published online in Wiley Online Library; DOI: 10.1002/cae.20449.
[4] T. Murphey, Teaching rigid body mechanics using student-created virtual environments, IEEE Trans Educ 51 (2007), 45–52.
[5] S. Dormido, R. Dormido, J. Sánchez, and N. Duro, The role of interactivity in control learning, Int J Eng Educ 21 (2005), 1122–1133.
[6] RoboWorks Software, Available at: http://www.newtonium.com/.
[7] Robot Assist Software, Available at: http://www.kinematics.com/.
[8] Easy-ROB Software, Available at: http://www.easy-rob.com/.
[9] RobCAD Tecnomatix, Available at: http://www.plm.automation.siemens.com.
[10] DELMIA for Robotics, Available at: http://www.plmv5.com/delmiarobotics/.
[11] GRASP simulation tool, Available at: http://www.bygsimulation.com/.
[12] R. Gourdeau, Object-oriented programming for robotic manipulator simulation, IEEE Robot Autom Mag 4 (1997), 21–29.
[13] A. Breijs, B. Klaassens, and R. Babuska, Matlab design environment for robotic manipulators, Proceedings of the 16th IFAC World Congress, Prague, 2005.
[14] R. Falconi and C. Melchiorri, Roboticad: An educational tool for robotics, Proceedings of the 17th IFAC World Congress, Seoul, 2008.
[15] P. Corke, A robotics toolbox for MATLAB, IEEE Robot Autom Mag 3 (1996), 24–32.
[16] S. Kucuk and Z. Bingul, An off-line robot simulation toolbox, Comput Appl Eng Educ 18 (2010), 41–52.
[17] T. J. Mateo and J. M. Andujar, Simulation tool for teaching and learning 3D kinematics workspaces of serial robotic arms with up to 5-DOF, Comput Appl Eng Educ, Published online in Wiley Online Library; DOI: 10.1002/cae.20433.
[18] M. Toz and S. Kucuk, Dynamics simulation toolbox for industrial robot manipulators, Comput Appl Eng Educ 18 (2010), 319–330.
[19] M. Cakir and E. Butun, An educational tool for 6-DOF industrial robots with quaternion algebra, Comput Appl Eng Educ 15 (2007), 143–154.
[20] Java Sun Website, Available at: http://www.oracle.com/technetwork/java/.
[21] F. Esquembre, Easy Java simulations: A software tool to create scientific simulations in Java, Comput Phys Commun 156 (2004), 199–204.
[22] C. Jara, F. Esquembre, F. Candelas, F. Torres, and S. Dormido, New features of easy Java simulations for 3D modeling, Proceedings of the 8th IFAC Symposium on Advances in Control Education, Kumamoto, 2009.
[23] J. Denavit and R. Hartenberg, A kinematic notion for lower-pair mechanisms based on matrices, J Appl Mech 22 (1955), 215–221.
[24] L. Sciavicco and B. Siciliano, Modeling and control of robot manipulators, 2nd ed., Springer Verlag, London, 2005.
[25] F. Chaumette and S. Hutchinson, Visual servo control I. Basic approaches, IEEE Robot Autom Mag 13 (2006), 82–90.
[26] S. Smith and J. Brady, SUSAN: A new approach to low level image processing, Int J Comput Vision 23 (1997), 45–78.
[27] Barrett Technology (Barrett Hand). Available at: http://www.barrett.com/.

BIOGRAPHIES

Carlos A. Jara was born in Alicante (Spain) and received the Industrial Engineering degree from Miguel Hernandez University of Elche (UMH) and Ph.D. degrees at the University of Alicante in 2005 and 2010, respectively. He works in the Automatics, Robotics and Computer Vision Group at the University of Alicante and he is now Associate Professor in the Department of Physics, Systems Engineering and Signal Theory. His research interests include virtual and remote laboratories, Robotics, industrial automation and visual control. Dr. Jara has belonged to the Spanish Committee of Automatics (CEA) since 2006.

Francisco A. Candelas was born in Alicante (Spain), and he received the Computer Science Engineering and Ph.D. degrees at the University of Alicante in 1996 and 2001, respectively. He has been a full-time Professor in the Department of Physics, Systems Engineering and Signal Theory at the University of Alicante since 1999. He has also been a researcher in the Automatics, Robotics and Computer Vision Group at the same university since 1996. His major research interests are virtual and remote laboratories, industrial automation and Computer Vision for real-time quality inspection. Dr. Candelas has belonged to the Spanish Committee of Automatics (CEA) since 1999.

Jorge Pomares was born in Alicante (Spain), and he received the Computer Science Engineer and the Ph.D. degree in the University of Alicante in 1999 and 2004, respectively. He is full-time Professor in the Department of Physics, Systems Engineering and Signal Theory of the University of Alicante since 2000. He also researches in the Automatics, Robotics and Computer Vision Group since 2000, where has involved in several research projects supported by the Spanish Government, as well as development projects in collaboration with regional industry. His current major researching interests are robotics, visual servoing and new teaching methodologies in engineering degrees. Dr. Pomares also belongs to Spanish Committee of Automatics (CEA) and he has published several international papers as well as participated as reviewer in international journals and conferences.

Fernando Torres received the Industrial Engineering and Ph.D. degrees from the Polytechnic University of Madrid (UPM) in 1991 and 1995, respectively. Since 1994, he has been at Alicante University, as a Professor in control, Robotics and Computer Vision. His research interests include automatic visual inspection, Robotics, manufacturing automation, visual servoing, morphological processing and new technologies for teaching. Prof. Torres is a member of the CEA-IFAC, IEEE and the Spanish Image Analysis and Pattern Recognition Society. He has published several international papers.