INT J COMPUT COMMUN, ISSN 1841-9836
8(4):622-634, August, 2013.

Using Augmented Reality in Remote Laboratories

H. Vargas, G. Farias, J. Sanchez, S. Dormido, F. Esquembre

Hector Vargas* and Gonzalo Farias
School of Electrical Engineering
Pontificia Universidad Católica de Valparaíso
Avda. Brasil 2147, Valparaíso, Chile
{hector.vargas, gonzalo.farias}@ucv.cl
*Corresponding author

Jose Sanchez and Sebastian Dormido
Department of Computer Science and Automatic Control
Spanish University for Distance Education
c/ Juan del Rosal 28040, Madrid, Spain
{jsanchez, sdormido}@dia.uned.es

Francisco Esquembre
Department of Mathematics
University of Murcia
Campus de Espinardo 30100, Murcia, Spain
fem@um.es

Abstract:
This paper introduces the concept of "augmented reality" as a novel way to enhance visualization in remote laboratories for engineering education. In a typical remote experimentation session, students get access to a remote real plant located at the laboratory to carry out their assignments. Usually, the graphical user interface allows users to watch the equipment by video stream in live. However, in some cases, visual feedback by video stream could be enhanced by means of augmented reality techniques, which mix together in one image, the video stream and computer generated data. Such mixture produces an added value to remote experimentation, increasing the sense of presence and reality, and helping to understand much better the concepts under study. In this work, a Java-based approach to be used in the remote experimentation context for pedagogical purposes is presented. Firstly, a pure Java example is given to readers (including the source code) and then, a more sophisticated example using a Java-based open source tool known as *Easy Java Simulations* is introduced. This latter option takes advantage of a new developed component, called camimage, which is an easy-to-use visual element that allows authors to capture video stream from IP cameras in order to mix real images with computer generated graphics.

Keywords: Augmented reality, Virtual-labs, Remote-labs, Engineering education.

1 Introduction

Augmented reality (AR for short) is a field of computer science which deals with the combination of real world and computer generated data. Virtual objects (computer generated graphics) can be overlaid with real world images in order to make them coexist dynamically in the same space. This basic idea is, nowadays, being widely applied in many areas such as [1] and [2]: TV marketing, mobile phones, video games, medicine, industry, among others [3–5].

In [1], author defines as AR-enabled applications when they fullfil with the following properties:

a) Combines real and virtual objects in a real environment.

b) Runs interactively, and in real time.

c) Registers (aligns) real and virtual objects with each other.

This last feature is the more complex one but, fortunately, it is not so important in the remote experimentation context because of the fact that in most cases, object and camera have static locations. Thus, remote laboratories should take advantage of the use of augmented reality techniques in order to give end users (mainly students) a major feeling of physical presence in the laboratory when working with the didactic equipment (also called plant) from distance.

Although the use of augmented reality in the field of engineering education could be quite convenient, there is no so much work done in this area so far, especially from an authoring point of view. Authors (mainly instructors) can try to combine video stream with computer generated images in their remote laboratories, however this process can require a specialized knowledge and advanced computer programming skills. In this context, the goal of this work is to provide a simple and easy-to-use approach to add augmented reality to remote laboratories.

The approach proposed uses *Easy Java Simulations* (EJS) to create the AR-enabled applications [6]. EJS is a freeware simulation tool developed by one of the authors of this article which has been awarded by the Science Prize for Online Resources in Education (SPORE) in november 2011 [7]. EJS can be used to build the graphical user interface of a remote laboratory. The augmented reality is added by selecting some interactive visual elements provided by EJS in combination with the element **camimage**, which is able to capture video stream from many different IP cameras by means the Java library **webcam.jar**. The development and use of the element *camimage* and the package *webcam.jar* is described in detail in this paper.

The article is organized as follows: Section 2 introduces the AR in the remote experimentation. In Section 3 a Java-based approach to add AR by using the Java library webcam.jar is described. Section 4 shows the use of *Easy Java Simulations* to create AR-enabled applications by means the element camimage. Some examples of remote laboratories are also shown in Section 4. Finally some conclusions and future work are discussed.

2 Augmented Reality in Remote Experimentation

The advances of the information and communication technologies has provided great new opportunities for education [8–10]. Networks, computer graphics and interactivity are just some great examples of them applied to engineering education.

In control education the impact of these technologies is even more significant. Experimentation in traditional laboratories is essential for students, who need to understand the fundamental concepts from both perspectives: theoretical and practical. However, the high costs associated with equipment, space, and maintenance staff, impose certain constraints on resources. A great effort of researching has focused on ways to overcome these limitations. Two of the most important results of such work are virtual and remote laboratories.

Virtual laboratories, which have become common place in the past few years, usually consist of a computer-based model of a real plant. This kind of simulations requires normally high levels of interactivity and visualization, since they are usually used to teach, in a human-friendly way, many key concepts in a particular discipline [11]. See some examples of virtual laboratories in [12] and [13].

The remote operation of real equipment is commonly referred to as remote laboratory and it allows students to manipulate physical plants (see Figure 1), located at the university, from

their home computers. This kind of experimentation reduces the time and location constraints of traditional laboratories [14]. See some examples of remote laboratories in [13, 15].

Figure 1: Heat Flow apparatus of Quanser.

Both virtual and remote laboratories provide, thanks to remote access, great opportunities for teachers to support the continuous process required for a life-long learning, as mentioned before.

Although the importance of virtual and remote laboratories for engineering education, the use of these tools as learning objects is not so usual. The main reason could be that the development of interactive applications is a difficult task from a computer programming point of view. Instructors, who are commonly not programming experts, can encounter problems trying to add user interaction or advanced visualization to existing virtual and remote laboratories. This is further complicated by the presence of different computer languages, programming techniques, network protocols, etc.

Despite the aforementioned difficulties, it is possible to find many examples of educational institutions that include virtual and remote laboratories as teaching tools in their current engineering and sciences curricula. Most of these examples offer the possibility of working on either a simulated version of a physical process or a real device located at the universities laboratories. However, such platforms have certain limitations [16] that should also be enhanced to take into account the advantages of augmented reality instead of showing only visual feedback by video stream, specifically:

- These developments are mainly focused on solving the technical issues related to the building of web-based lab solutions and not providing specific software tools designed to meet these goals. In general, they do not take into account the programming issues that hinder control engineering teaching staff when designing and developing virtual and remote laboratories [17–21]. Remote laboratory environments such as [17, 18] and [20] only provides raw video stream as feedback for learners.

- Most of these environments do not consider the social context of interaction and collaboration among students (and between teachers and students) in traditional hands-on laboratories [22, 23]. Such environments use augmented reality with their remote laboratories, but they lack of easy-to-use authoring tools to create new AR-enabled applications.

- Unlike the Cyberlab and eLabs FEUP [24], most of the cases are isolated experiences coming from university engineering departments that offer only a limited set of experiments (only those existing in their own labs). There is any kind of augmented reality added to these laboratories, only video stream is available on them.

This article proposes to take advantage of augmented reality in remote experimentation by mixing video and simulation all in one, in the same way that an AR-enabled application does it. Figure 2 show this idea.

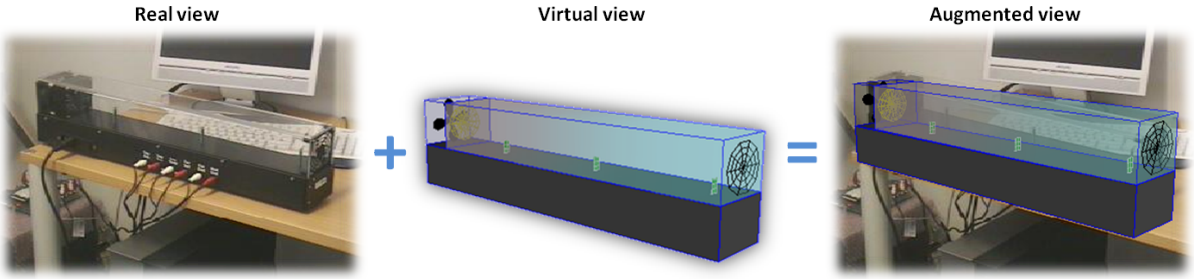

Figure 2: Heat Flow with augmented reality.

Based on previous description, it is possible to realize that, in the remote experimentation context, both real world information (remote laboratories) and computer generated information (virtual laboratories) could be used together.

The mixing between real-world and virtual objects at the same time (in one image) can be used in remote experimentation with pedagogical purposes under the following working schemes:

- *Virtual objects whose behaviour depend on simulated data.* The real-world visualization is overlaid with computer generated graphics based on the dynamics of a mathematical model of the plant.
- *Virtual objects whose behaviour depend on real data.* The real-world images are mixed with computer generated graphics by using real measurements of the remote target plant.

The first scheme can be used to compare the system's response of a mathematical model against the behaviour of a real equipment located at the laboratory. The second approach mixes real-world images into computer generated graphics in order to add extra visual information to the video combination. Both schemes are introduced here in order to be used when creating remote experimentation in engineering education.

In the Sections 3 and 4 two approaches to add augmented reality to remote laboratories are shown. The first approach describes in detail the use of a Java package, called *webcam.jar*. Second approach introduces the EJS visual element *camimage*, which uses internally the library *webcam.jar*. This latter approach is highly recommended to be used by instructors in order to create remote laboratories with augmented reality features.

3 A Java-based approach

In the remote version of a Web-based laboratory, adding a visual feedback module is an essential element in any tele-operated environment since it allows the users to feel and be aware of the consequences of their actions during a remote working session. As a result of this, users are more motivated and confident in the use of the system.

On the other hand, AR usually considers capturing of video stream (real world information) in a local mode whereas in remote experimentation the video stream should be obtained from a remote source.

To fullfil with the requirement above mentioned, two alternatives could be tested. The first one considers the use of an IP camera with a built-in video server dedicated to capturing

images and post-publishing them as video stream through its network interface while the second alternative considers using a conventional webcam with serial interface (USB o Firewire) directly connected to a server computer (where the server computer could be the same host that controls the real plant). The publishing of the video stream is not direct in this case since the image data must be acquired via webcam through its serial interface first and then socket techniques should be used to send these data across the network.

3.1 The Java library webcam.jar

In order to read the video stream published by the video server of an IP camera a Java library called *webcam.jar*[1] was developed. This API allows Java programmers to use the classes and methods of the library to get video images from any IP camera connected to the Internet.

The software provides a first level of interface with the camera (see listing 1), allowing to write high level software hiding the low level programming details such as the structure of the received packets, the decoding of the HTTP headers, or the sockets implementation.

Listing 1: Main Java class of webcam.jar.

```
1 package webcam;
2
3 // Main Java Interface of the library webcam.jar
4 public interface IVideo{
5
6   // Public methods to dialog with a video camera
7   connect();
8   disconnect();
9   boolean isConnected():
10  int getAvailableBuffer();
11  Image getImage();
12
13 }
```

The main class of the API, called *Video.class*, implements the Java interface shown in previous listing. The constructor of this class allows to create an instance of a *Video* object with the following input arguments:

- A *String* indicating the URL where the camera supplies the video stream.
- A *boolean* indicating what the reading format either (MPEG[2] (true) or JPEG[3] (false)).
- An *integer* to add a delay (in milliseconds) in the reading of the images.

Once an object of Video class has been created, a *Java thread* is ready to be started. As mentioned above, *Video* class provides a set of public methods to dialog with the camera. These methods are described below:

`public connect();`

When this method is invoked on a *Video* object previously created, the `run()` method of a *Java thread* is executed. This method opens a socket connection to the remote IP camera and it starts reading of the video stream according to the chosen format.

`public disconnect();`

[1] Accesible from `http://www.profesores.pucv.cl/hvargas/augmented/augmentedreality.html`

[2] The Moving Picture Experts Group, commonly referred to as simply MPEG, is a working group of ISO/IEC charged with the development of video and audio encoding standards.

[3] In computing, JPEG is a commonly used method of compression for photographic images. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. JPEG typically achieves 10 to 1 compression ratio with little perceivable loss in the image quality.

This method stops reading the video stream and closes the socket objects created at low level. The `run()` method of the thread is over when this method is executed.

`public boolean isConnected();`

Enquires if the link to the camera is up or down. Commonly, this boolean indicator is used when the `getImage()` is invoked to get the images.

`public int getAvailableBuffer();`

Enquires if there are available images in the input buffer. The `getImage()` method should be invoked if we are in this case.

`public Image getImage();`

The images captured from the IP camera can be obtained from the input buffer by using this method. The method returns an object of the *java.awt.Image* class. Image objects are easily displayed on *Java swing* [4] *components*.

Summarizing, Java programmers can import this library to capture the video stream of any IP camera connected to the network. With regards to the present essay, the library was developed in such away that its inclusion in the development process of a new virtual and remote laboratory can be achieved with a minimum programming effort.

3.2 A simple pure Java example of AR

Listing 2 shows a very naive example but descriptive of the basic idea about augmented reality in the remote experimentation context. In this example, a small virtual object is moved on real world video images coming from a remote IP camera connected to Internet (see Figure 3).

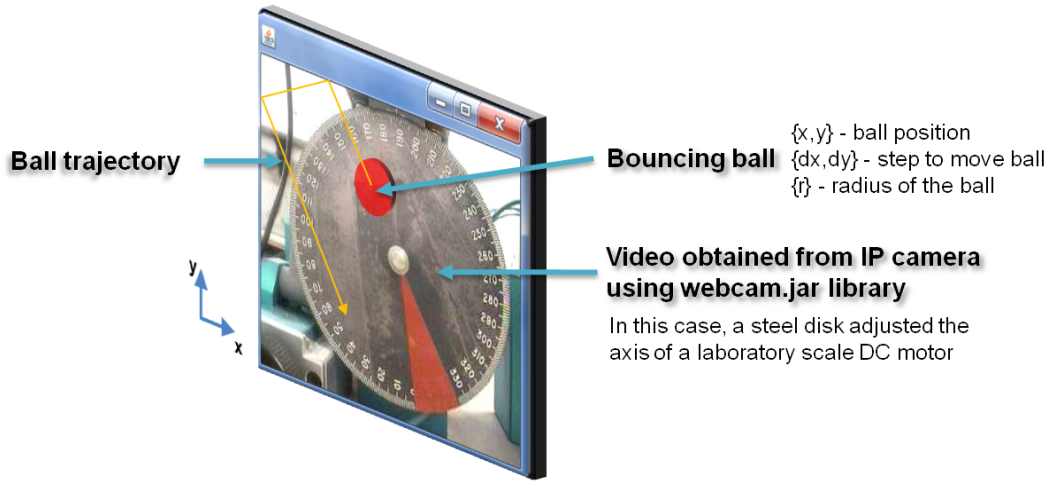

Figure 3: Simple Java-based AR application.

This example illustrates, in a very basic way, how to apply AR techniques in the remote experimentation context. However, as the reader can appreciate, drawing and animating virtual objects in Java is not a trivial task for non-expert programmers. Moreover, in most cases, augmented reality involves the creation of professional animations in which a set of virtual objects compose a more complex animated structure than a simple bouncing ball.

[4]Swing is a widget toolkit for Java. It is part of Sun Microsystems' Java Foundation Classes (JFC) - an API able to provide a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit.

Listing 2: A pure Java example of AR.

```
package arbasic;

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import webcam.Video;

// A bouncing ball on remote video streaming
public class ARExample extends JFrame implements Runnable {

  int x = 150, y = 50, r = 25;
  int dx = 11, dy = 7;
  Thread animator;
  String URL = "http://IP_address/mjpg/video.cgi";
  Video cam = new Video(URL, true, 20);
  volatile boolean pleaseStop;

  //It implements Runnable interface
  public void run() {
    while (!pleaseStop) {
      animate();
      try {
        Thread.sleep(40);// Wait 40 milliseconds
      }catch (InterruptedException e) {
      } // Ignore interruptions
    }
  }
  // It renders both video images and ball
  public void paint(Graphics g) {
    Graphics2D g2d = (Graphics2D)g;
    g2d.drawImage(video.getImage(), null, 0, 20);
    g2d.setColor(Color.red);
    g2d.fillOval(x-r, y-r, r*2, r*2);
  }
  // It calcules new position of ball and repaints
  public void animate() {
    Rectangle bounds = getBounds();
    if ((x-r+dx < 0) || (x+r+dx > bounds.width))
      dx = -dx;
    if ((y-r+dy < 0) || (y+r+dy > bounds.height))
      dy = -dy;
    x += dx;
    y += dy;
    repaint();
  }
  // It starts animation and video
  public void start() {
    animator = new Thread(this);
    pleaseStop = false;
    animator.start();
    video.connect();
  }
  // Flag to stop the animator thread
  public void stop() {
    pleaseStop = true;
  }
  // Main method of the Java class
  public static void main(String args[]) {
    final ARExample frame = new ARExample();
    frame.start();// Executing run() method
    frame.addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent we){
        frame.stop();
        System.exit(0);
      }
    });
    frame.setSize(350, 263);
    frame.setResizable(false);
    frame.setVisible(true);
  }
}
```

In this context, next section describes a new approach that considers the use of the open source software tool called *Easy Java Simulations* (EJS). EJS helps to create more complex Java programs that exhibit a high degree of interactivity and graphical standards.

4 A novel approach using EJS

As shown in previous section, displaying a simple Java animation on real world images is not a straightforward task for non-expert programmers. In order to cope with this problem, the following subsections show how it is possible to create more sophisticated AR-enabled Java applications by using the open source tool *Easy Java Simulations*.

4.1 What is EJS?

EJS is an open-source, free of charge, authoring tool specialized in the creation of interactive simulations for (mainly, but not only) pedagogic uses. EJS is designed to make it easy for a non programming specialist to implement a scientific model into computer form and to design and build a graphical user interface (GUI) which meets both the visualization and interaction capabilities required for an effective computer-based instruction of a given scientific phenomenon.

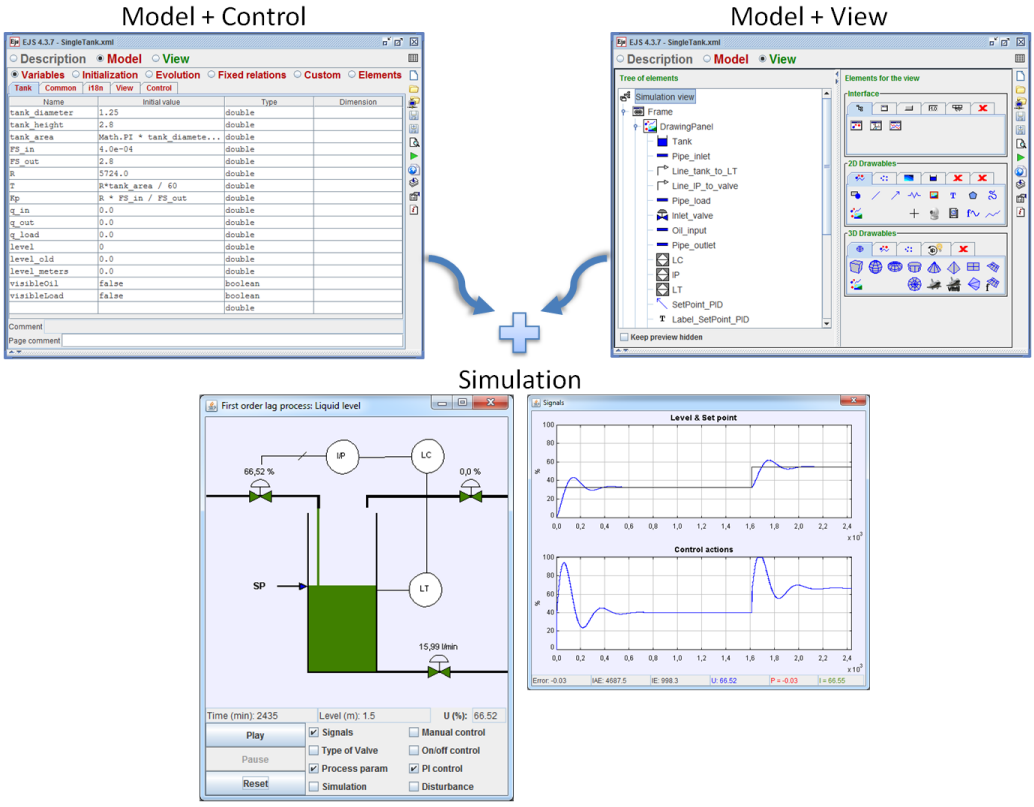

Figure 4: Creation process of an EJS application.

The architecture of EJS derives from the model-view-control (MVC) paradigm, whose philosophy is that interactive simulations must be composed of three parts (see Figure 4) :

- The *model*, which describes the process under study in terms of, 1) variables, which hold the different possible states of the process, and 2) relationships among these variables, expressed by computer algorithms.

- The *control*, which defines certain actions that a user can perform on the simulation.
- The *view*, which shows a graphical representation (either realistic or schematic) of the process states.

EJS makes things even simpler by eliminating the "control" element of the MVC paradigm and fuses one part in the *view* and the other one in the *model*. Thus, applications are created in two steps:

1. Defining the *model* to simulate by using the built-in simulation mechanism of EJS and,
2. Building the *view* showing the model state and answers to the changes made by users.

Figure 4 shows a simple virtual-lab created by EJS for teaching basic control concepts based on the well-known single-tank process. As appreciated, part of the control logic is programmed when defining the *model* and the another when creating the *view*.

EJS provides developers a set of predefined graphical elements to compose the graphical aspect of a simulation. In this context, EJS may accept new graphical elements to make easier the task of composing a *view*. Following this philosophy, the library **webcam.jar** was integrated into EJS as a new *view* element. The following subsection presents this new component and how it can be used by EJS developers.

4.2 Integration of webcam.jar into EJS

In order to simplify the use of the **webcam.jar** library, a new view graphical element of EJS called **camimage** was developed (see Figure 5). The novelty of this approach lies in that the element was implemented based on the characteristics of an existing graphical component of EJS, the **InteractiveImage**.

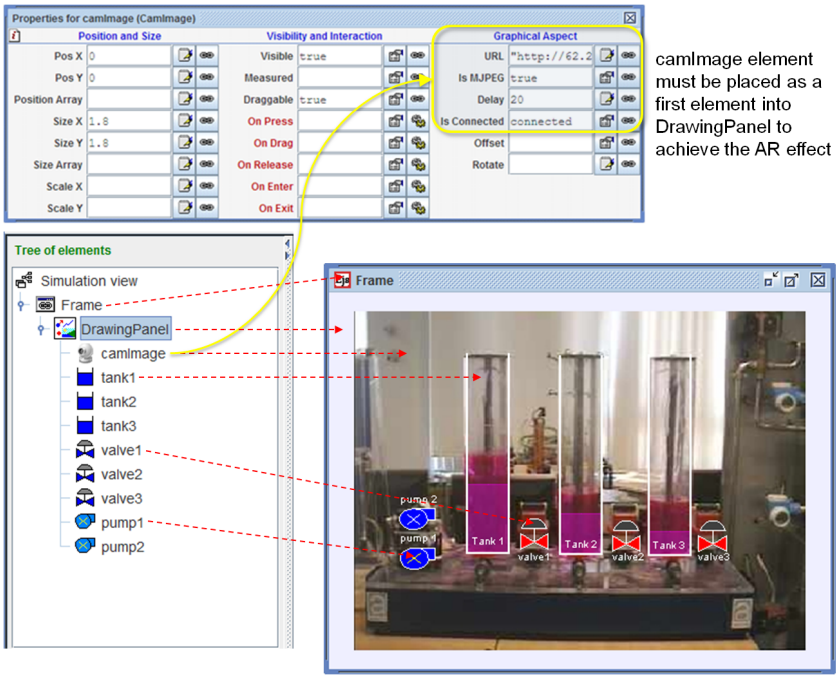

Figure 5: Simple illustration of the *augmented reality* concept for remote labs in EJS.

Thus, the `camimage` element inherits all methods and properties of this last object and, can therefore, be placed on an EJS container with coordinate axis such as the "DrawingPanels" or the "PlottingPanels" in order to render the images from the camera.

In addition to the properties inherited from the InteractiveImage element, four new parameters must be configured. These parameters are related to the IP camera used since a `Video` object (from the `webcam.jar` library) is instantiated when adding this object into an EJS view. The parameter **URL** has to be set with the URL of the IP camera video stream. The option **MJPEG** can be used to indicate whether the image format is MJPEG. The refresh rate of the images can be controlled by **DELAY**. Finally, the option **Connected** enables or finishes the connection with the IP camera.

Applying the augmented reality concept in EJS is now very simple. Figure 5 shows how to compose an EJS view with augmented reality features. This effect is achieved when adding a **camimage** element as a first object of a drawing container of EJS, since the captured images will always appear in the background, behind other virtual graphical elements that may form more advanced Java animations.

The direct application of the augmented reality concept and its benefits in remote experimentation will be illustrated by means of examples in the following subsections.

4.2.1 Example 1: An AR-enabled remote laboratory of a thermal process

Figure 5 showed how to apply AR techniques on a scene 2D using EJS. However, enhancing visualization in 3D scenes is also possible because of the fact that **camimage** element can be nested within a DrawingPanels3D container.

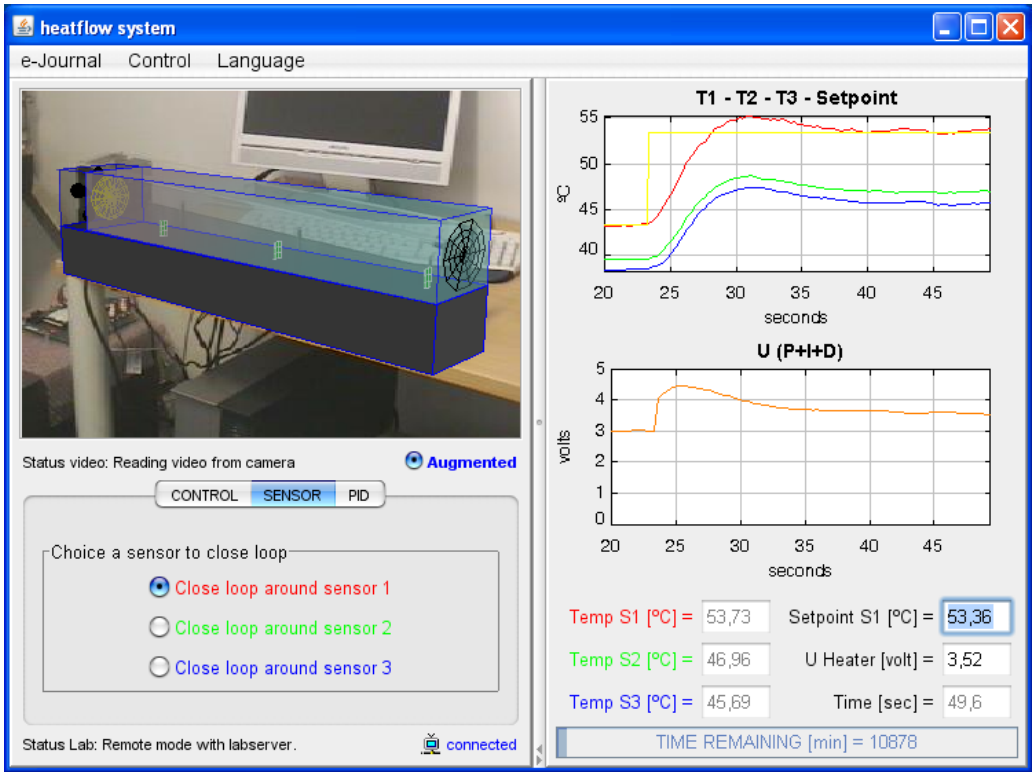

Figure 6: Working remotely with AR.

Figure 6 shows an example where the previously described idea is applied. This application is part of a set of virtual and remote laboratories that the Department of Computer Science

and Automatic Control at the Spanish University for Distance Education (UNED) provides to students for remote experimentation, see other example in [25]. The heatflow system is based on a thermal process that allows students to conduct practical experiments on temperature control systems with transport delays. The interface is easily accessed by students through Internet by means of a simple web browser that loads an EJS-based Java applet.

Students use this tool to study basic control concepts such as *systems' identification* and *PID controllers design*. Firstly, they perform open-loop experiments in order to get data registers that allow them to identify a model of the process. In a second stage, students must design PID controllers to regulate temperature in each sensor based on the previously identified models.

As the reader can imagine, temperature changes are not visible to the human eye. To facilitate visualization of the heat-flow dynamics, the interface's left panel displays a 3D representation of the apparatus in which the inner air's color changes according to temperature data obtained from a remote server. This virtual representation is displayed onto video streaming captured from a remote IP camera. In this case, the augmented reality is a very useful technique that helps students to get a better insight about the physical phenomenon under study.

4.2.2 Example 2: An AR-enabled Web-based lab of a robot arm

A very nice AR-enabled remote lab developed with the approach previously described is shown in Figure 7. The system located at the University of Alicante (UA) was entirely developed for the teaching and learning of automation and robotics.

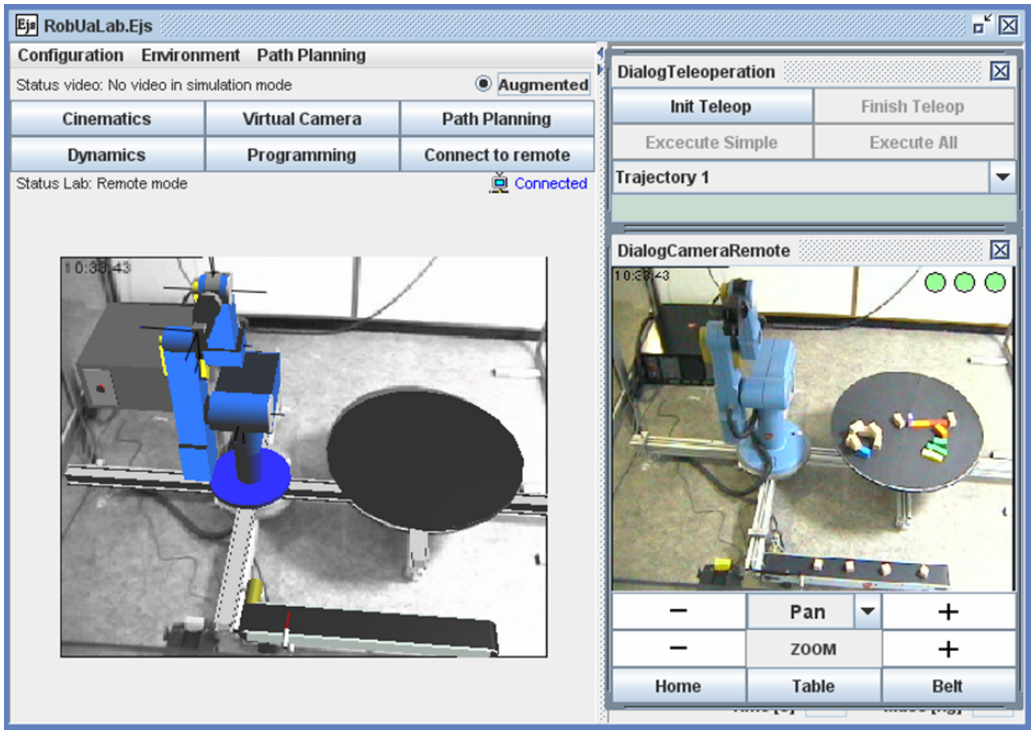

Figure 7: Working remotely with AR.

The main hardware components of the system are an industrial robot Scorbot ER-IX (Intelitek) with five degrees of freedom, a small warehouse for pieces, a conveyor belt, and a rolling table. Right side on Figure 7 shows the actual plant, which was built and assembled by a research group at UA. For further details about this laboratory see [26].

The video streaming captured from the remote IP camera is enriched by a virtual representation of the robot arm following the proposed approach. The dynamics of this virtual view can be updated by two ways: by using position data from the encoders of the actual robot arm, or by using position data calculated from the mathematical model of the system. Note that the location of the source of the comming position data is different in both cases. In the first one data arrive from the remote server, whereas in the second case data are computed locally by the EJS solver. In this example, AR allows to compare the simulated and real behaviour of the system by observing a motion gap between the virtual and real arm.

5 Conclusion and Future Works

The article introduces the using of the augmented reality in remote experimentation with pedagogical purposes. Augmented reality mixes real world images with computer generated data in order to give a major feeling of physical presence to students. This work focuses on providing to instructors a new approach to create application with augmented reality capabilities.

The approach described uses *Easy Java Simulations* in combination with a novel visual element called **camimage** to build the AR-enabled applications. The **camimage** element uses, in turn, the Java package **webcam.jar** to manipulate easily different types of IP cameras. Some examples of the use of the approach proposed are described. Future work will take into account the register issue, which could be quite useful when either the camera or the physical device are moving, such as the case of a remote laboratory of mobile robots.

Bibliography

[1] Azuma, R. (1997); A Survey of Augmented Reality, *Presence: Teleoperators and Virtual Environments*, 6(4): 355-385.

[2] Azuma, R. et al. (2001); Recent advances in augmented reality, *IEEE Computer Graphics and Applications*, ISSN 0272-1716, 21(6): 34-47.

[3] Feiner, S. et al. (1997); A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment, *Personal and Ubiquitous Computing*, ISSN 1617-4909, 21(6): 208-217.

[4] Kalkofen, D. et al. (2007); Interactive Focus and Context Visualization for Augmented Reality, *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISBN 978-1-4244-1749-0, pp. 1-10.

[5] Klopfer, E. and Squire, K. (2008); Environmental Detectives the development of an augmented reality platform for environmental simulations, *Educational Technology Research and Development*, ISSN 1042-1629, 56: 203-228.

[6] Esquembre, F. (2004); Easy Java Simulations: a software tool to create scientific simulations in Java, *Computer Physics Communications*, ISSN 0010-4655, 156(2): 199-204.

[7] *Ejs Java Simulations* home page, `http://www.um.es/fem/EjsWiki/`.

[8] Dormido, S. (2002); Control learning: Present and future, *IFAC Annual Control Reviews*, 28: 115-136.

[9] Dormido, S. et al. (2005); The role of interactivity in control learning, *The International Journal of Engineering Education*, 21(6): 1122-1133.

[10] Sánchez, J. et al. (2005); The learning of control concepts using interactive tools, *Computer Applications in Engineering Education*, 13(1): 84-98.

[11] Farias, G. (2010); Adding Interactive Human Interface to Engineering Software, *Ph.D. Thesis from Department of Computer Science and Automatic Control*.

[12] Farias, G. et al. (2010); Java Simulations of Embedded Control Systems, *Sensors*, 9(10): 8585-8603.

[13] Vargas, H. et al. (2009); Web-enabled Remote Scientific Environments, *Computing in Science and Engineering*, 11(3): 36-46.

[14] Vargas, H. (2010); An Integral Web-based Environment for Control Engineering Education, *Ph.D. Thesis from Department of Computer Science and Automatic Control*.

[15] Farias, G. et al. (2010); Developing Networked Control Labs: A Matlab and Easy Java Simulations Approach, *IEEE Transactions on Industrial Electronics*, 57(10): 3266-3275.

[16] Aliane, N. (2010); Limitations of Remote Laboratories in Control Engineering Education, *International Journal Online Engineering*, 6(1): 31-33.

[17] Casini, M. (2003); The Automatic Control Telelab: A User-Friendly Interface for Distance Learning, *IEEE Transactions on Education*, 46(2): 252-257.

[18] Casini, M. (2007); Web-Based Control and Robotics Education, *Publisher Springer*, pp. 127-151.

[19] Martín, C. (2007); Web-Based Control and Robotics Education, *Publisher Springer*, pp. 103-125.

[20] Internet Remote Experimentation from NUS (2011); `http://vlab.ee.nus.edu.sg/vlab`.

[21] Dormido, S. el al. (2005); Adding Interactivity to Existing Simulink Models Using Easy Java Simulations, *Proc. 44th IEEE Conference on Decision and Control and the European Control*, Sevilla, Spain.

[22] Nguyen, A.V. (2006); Iterative Design and Evaluation of a Web-Based Experimentation Environment, *User-Centered Design of Online Learning Communities*, pp. 286-313.

[23] Nguyen, A.V. (2007); Activity Theoretical Analysis and Design Model for Web-Based Experimentation, *Proc. Int'l Conf. Human-Computer Interaction*.

[24] Restivo M.T. and Silva M.G. (2009); Portuguese Universities Sharing Remote Laboratories, *Int'l J. Online Eng.*, 5: 16-19.

[25] Dormido, R. et al. (2008); Development of a Collaborative Web-Based Control Laboratory for Automation Technicians: The Three-Tank System, *IEEE Transactions on Education*, 51(1): 35-44.

[26] Vargas, H. et al. (2011); A Network of Automatic Control Web-based Laboratories, *IEEE Transactions on Learning Technologies*, 4(3): 197-208.