The Internet-Based UJI Tele-Lab: System Architecture*

R. Marín P. J. Sanz P. Nebot R. Esteller
Universitat Jaume I
Av. Sos Baynat, s/n. E-12006 Castelló - SPAIN
{rmarin, sanzp, al065289, esteller}@uji.es

Abstract - *In this paper, we present the system architecture that has been used in order to implement a whole Internet-Based Tele-Laboratory, which allows researchers and students to program remotely an online robot. In fact, the challenge has been demonstrating that remote programming combined with an advanced multimedia user interface for remote control is very convenient, flexible and profitable for the design of a Tele-Laboratory. In fact, the contribution consist of designing a system architecture that permits to any external program (i.e. remote experiment, speech recognition module, etc.) to have access to almost every feature of the already existing "UJI Online Robot" (i.e. cameras, object recognition, robot control, etc.) [1].*

Keywords: Robotics TeleLabs, Remote Programming, Education & Training, and Distributed Systems

1 Introduction

Online robots enable multiple users to have control over a remote robotic system in a teleoperated manner. This kind of systems are very useful for teaching robotics, due to the fact that they enable the general public (e.g. students) to gain experience of robotics technology directly [1]. In fact, the next step is giving more freedom to the operator, and not only enabling remote control of the robotic system, but also remote programming.

Enabling remote programming of the robot system permits developing external programs that take control over the whole set of robotic functions. Thus, for example, we could design an experiment in Java for performing a visual servoing manipulation, or we could even use this interface for designing a voice-operated robot. In fact, this is what we have done for the TeleLab.

The interest for the design of Internet-based Tele-Laboratories is increasing enormously, and this technique is still very new. A very good example of already existing experiments in this area can be found in [4].

2 Experimental Setup

The robot scenario is appreciated in Figure 1, where three cameras are presented: one taking images from the top of the scene, a second camera from the side, and a third camera from the front. The first camera is calibrated, and used as input to the automatic object recognition module and 3D-model construction. The other two cameras give different points of view to the user when a teleoperation mode is necessary in order to accomplish a difficult task (e.g. manipulating overlapped objects).

Once the user gets an online connection to the robot, the manipulator goes to the initial position, so the whole scene information is accessible from the top camera. At that moment, the computer vision module calculates a set of mathematical features that identify every object on the board [3]. Afterwards, the contour information is used in order to determine every stable grasp associated with each object [5], which is necessary for vision-based autonomous manipulation. This process is a must whether high level task specification from a user is required, and also in order to diminish network latency.

Meanwhile, we shall use the output from these visual algorithms in order to construct a complete 3D model of the robot scenario (i.e. the objects and the robot arm). Thus, users will be able to interact with the model in a predictive manner, and then, once the operator confirms the task, the real robot will execute the action [2].

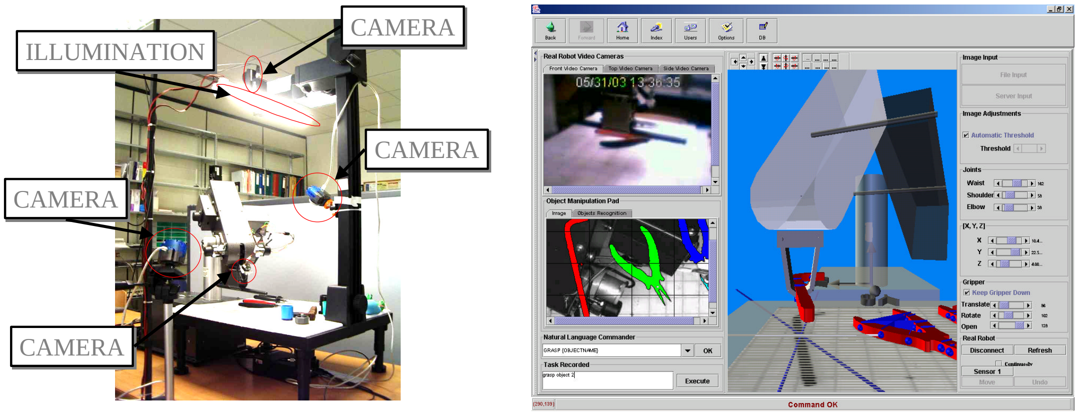
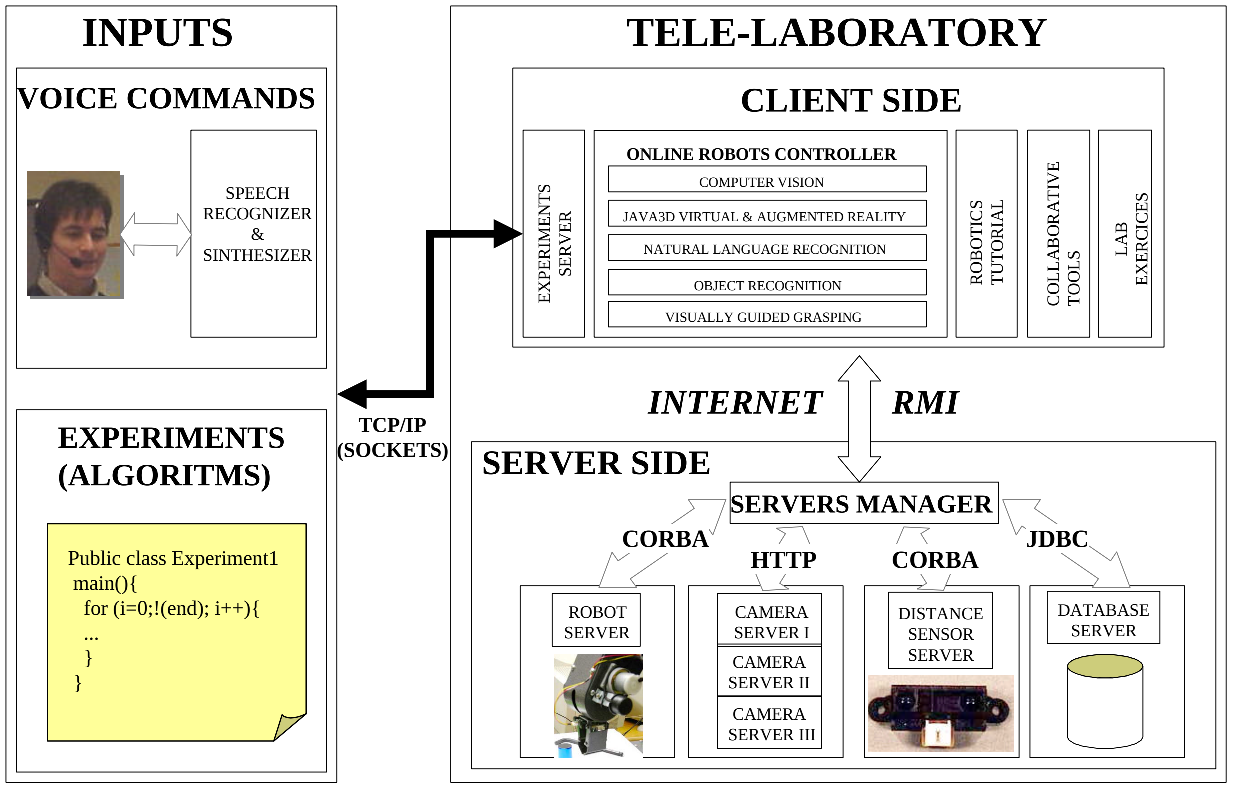


Figure 1. Experimental setup (left); and the user interface associated (right).

3 System Architecture

As we can see in Figure 2, the Tele-Lab accepts experiments (i.e. algorithms) as inputs using any programming language capable of managing TCP/IP sockets. We already provide a Java library for using the robot in a simply manner. Future efforts will be oriented to facilitate this library not only in Java but also with other programming languages (i.e. Matlab, C, etc.).

The outputs of the experiments are returned to the user by means of the Tele-Laboratory "Client Side" user interface, which permits the operator to see the results of their programmed actions directly on a multimedia Java3D user interface.



```
Public class Experiment1
 main(){
   for (i=0;!(end); i++){
   ...
   }
 }
```

Figure 2. System architecture for the Internet-based Tele-Lab

4 The "Experiment" Java Library

As seen in Figure 2, the interface that the TeleLab provides to remote applications (e.g. experiments) is called the "Experiments Server". The "Experiments Server" maintains an open TCP/IP socket over a well-known port (i.e. 7745). It is on this socket where the external applications must be connected to be able to communicate with the Tele-Laboratory.

Moreover, the "Experiments Server" is responsible for receiving the commands and the data from the applications (e.g. students experiments), as well as sending back to clients the results and evaluation of the execution of these actions on the TeleLab.

The "Experiment" Java Library has the benefit of giving the user the possibility to have a copy of object instances that are running in the TeleLab itself. To do that, it is necessary that those Java objects be "serialized" (i.e. the object can be transformed into a chain of bytes). To allow this, Java has the "Serializable" class, as a way to not only send strings through a socket, but also object's instances.

Again, if the object to be transferred by the socket contains attributes, every one of them must be serializable too. If they are not, the object could not be transferred.

The "Experiment" Java Library consists of a class/library and a skeleton for the accomplishment of experiments in the TeleLab. The experiments (i.e. user algorithms) need to have information about the real robot scenario (i.e. camera inputs, object recognition, etc.). The whole set of data referred to robot vision are all stored into a class called "SceneManager". Therefore, the transfer of this object to the experiments will be necessary. To do that, the "Experiments" Java Library includes a method called "getSceneManager", which provides the user the capability of getting on the experiment a copy of the SceneManager object that has the information of the real robot environment.

As explained in Figure 3, once a student or a scientist wants to program an experiment by using the "Experiments" Library, the best way (and unique) to begin is creating a Java class (e.g. Experiment1) that extends from the "Experiment" class. By doing this every aspect related to sockets programming and object serialization are encapsulated.

Thus, every experiment matches a template (see Figure 3) that we call "Experiment_template", and they all have the same structure: (1) extending "Experiment" class, (2) creating an instance of the experiment, (3) calling the "getSceneManagerSer" to obtain the serialized objects from the TeleLab, (4) executing the corresponding actions on the TeleLab, and (5) closing the connection.

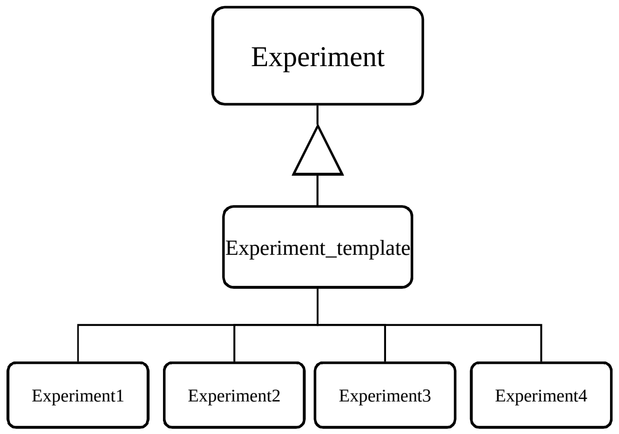


Figure 3. The "Experiment" Java Library Software Architecture

The template has the following form:

```
public class Experiment_template extends Experiment {
  public Experiment_template(String host,int port) {
    super(host,port);
  }

  public static void main(String[] args) {
    // Fixed part that connects with the server and obtains the data from
him
    Experiment_template et = new
                         Experiment_template("127.0.0.1",7745);
    et.connect();
    SceneManagerSer sms = et.getSceneManagerSer();

    // To fill out with the specific actions to do with the scene data
    // obtained from the server

    // Fixed part that disconnects the experiment from the server
    et.disconnect();
  }
}
```

5 Examples of Experiments

At this section we are presenting 4 simple examples that have been practiced in our laboratory by a pilot group of students.

5.1 Objects Attributes Experiment

The first example consists of designing an experiment that simply uses the "Experiment" library in order to obtain information about the objects that are currently on the remote robotics scenario (i.e. number of objects, area, etc.).

```
public class ExperimentExample1 extends Experiment {
  public ExperimentExample1(String host,int port) {
    super(host,port);
  }

  public static void main(String[] args) {
    // Fixed part that connects with the server and obtains the data from
him
    ExperimentExample1 ee1 = new
ExperimentExample("127.0.0.1",7745);
    ee1.connect();
    SceneManagerSer sms = ee1.getSceneManagerSer();

    System.out.println("Width: " + sms.imgWidth + " Height: " +
sms.imgHeight);
    for (int i=0;i<sms.numObjeto;i++) {
      SceneObject so = sms.getSceneObject(2);
      System.out.println("Area Object: " + so.area);
    }

    // Fixed part that disconnects the experiment from the server
    ee1.disconnect();
  }
}
```

As can be seen in the previous algorithm, we follow the “Experiment_template” structure and print on the console the area of every object present in the scene. This can be easily implemented by using the object serialization feature presented above.

5.2 Path Planning Experiment

In this second experiment the student uses the remote programming feature to bring the robot from a point to another (i.e. from [x1, y1, z1] to [x2, y2, z2]) by following a straight line.

```
public class ExperimentPath  extends Experiment {
  public ExperimentPath(String host,int port) {
    super(host,port);
  }

  public static void main(String[] args) {
    ExperimentPath ep = new ExperimentPath("127.0.0.1",7745);
    ep.connect();

    int[] ini = {-15,30,15};    int[] fin = {15,10,5};
    int incr = 2;    String orden;
    double t = 0.0;    long y = 0, z = 0;
    long x = ini[0];    int dist = fin[0] - ini[0];

    for (int i=0;i<=dist;i+=incr) {
      x += incr;
      t = (x - ini[0]) / (double)(fin[0] - ini[0]);
      y = Math.round(ini[1] + (t * (fin[1] - ini[1])));
      z = Math.round(ini[2] + (t * (fin[2] - ini[2])));

      orden = "move to position " + Long.toString(x) + " " +
         Long.toString(y) + " " + Long.toString(z);
      sendOrder(orden);
      orden = "confirm"; sendOrder(orden);
    }
    ep.disconnect();
  }
}
```

5.3 Pick and Place Experiment I

The third experiment consists of launching a grasping action over one of the objects in the scene, by using the most stable grasping points. In particular, we launch the action “grasp object 1” on the TeleLab, then we move the gripper to the quadrant 8 of the robot scenario, and finally we ungrasp the object by executing the command “ungrasp”. As can be seen in the following algorithm, the action “confirm” is necessary in order to send the action over the real robot and not only on the 3D virtual environment.

```
public class ExperimentPick extends Experiment {
  public ExperimentPick(String host, int port) {
    super(host, port);
  }

  public static void main(String[] args) {
    ExperimentPick ep = new ExperimentPick("127.0.0.1", 7745);
    ep.connect();

    sendOrder("grasp object 1");
    sendOrder("confirm");

    for (int i=0;i<5;i++) {
      sendOrder("move up");  sendOrder("confirm");
    }

    sendOrder("move to quadrant 8");
    sendOrder("confirm");

    sendOrder("ungrasp");
    sendOrder("confirm");

    ep.disconnect();
  }
}
```

In Figure 4 we can appreciate the states of the robot during the execution the the Pick and Place Experiment I

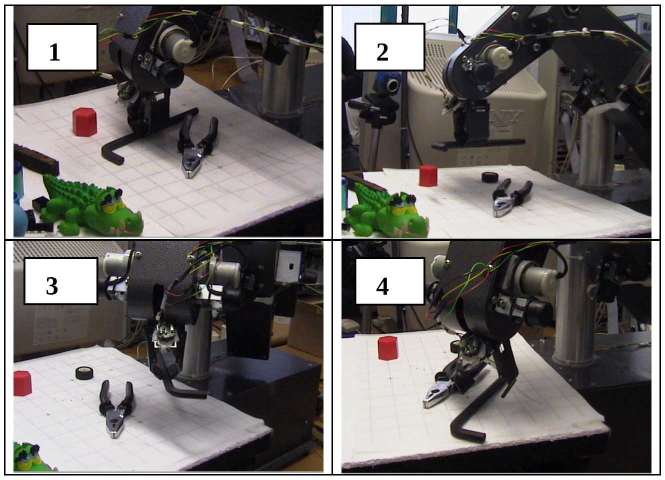


Figure 4. Snapshots of the educational robot when executing the Pick and Place Experiment I.

5.4 Pick and Place Experiment II

This second Pick and Place Experiment enables the user to execute a grasping action over two particular points of the object contour. For this example the two selected points are defined by the intersection of the Object's Maximum Inertia Axis and its contour. For a big broad of applications this alternative is sufficient.

As can be seen in the algorithm, in this situation we add a grasping to the object by using the object's attributes "p1" and "p2" that represents the contour points that intersect with the Maximum Inertia Axis.

After that, we execute the grasping command on that object by using the already created grasping.

```
public class ExperimentCandidate extends Experiment {
  public ExperimentCandidate(String host, int port) {
    super(host, port);
  }
  public static void main(String[] args) {
    ExperimentCandidate ec = new ExperimentCandidate("127.0.0.1",
7745);
    ec.connect();
    SceneManagerSer sms = ec.getSceneManagerSer();

    int object = 3;
    SceneObject so = sms.getSceneObject(object);
    sendOrder("add grasp " + so.p1 + " " + so.p2 + " to object " +
object);
    sendOrder("confirm");

    sendOrder("grasp object " + object + " using " + (so.nGrasps-1));
    sendOrder("confirm");

    for (int i=0;i<5;i++) {
      sendOrder("move up");  sendOrder("confirm");
    }

    sendOrder("move to quadrant 15");
    sendOrder("confirm");

    sendOrder("ungrasp");
    sendOrder("confirm");

    ec.disconnect();
  }
}
```

6 Results and Conclusions

The present paper presents an extension of the UJI Online Robot architecture that enables not only control a robot by interacting with an advanced user interface, but also letting the scientist and students program their own experiments by using the Java programming language.

This has been possible thanks to the definition of the "Experiments" library, that allows in a encapsulated way to provide the TeleLab clients with the serialized objects necessary to perform these action.

Until now we have performed the following remote programming experiments:

1. Extracting information about the robot scenario (i.e. objects areas, etc.).
2. Position control of the robot using different trajectories.
3. Pick and Place operations using already defined grasping points.
4. Pick and Place operations letting the scientist to calculate their grasping points.

The researchers that have been using the Tele-Lab until now have found it very useful due to the fact that it enables controlling the whole robotic system (cameras, object recognition, predictive interface, robot control, etc.) from a single library. It can be considered as a very good alternative for researchers to perform a rapid prototyping of their algorithms by using a real robotic system accessible from any computer. Moreover, a pilot group of students have been using the "Experiments" library to design their own experiments. They became very concerned and motivated with the experience.

Future efforts will focus on the design of more advanced experiments for remote manipulation, as well as the study of automatic methods for evaluation of the experiments.

In fact, we are preparing a Visual Servoing experiment to let students of the next undergraduate robotics course to use the TeleLab in order to explore this interesting subject.

Acknowledgments

This is to acknowledge sources of financial support for this research, that was provided in part by the Spanish Ministry of Science and Technology (CICYT) under project CICYT-DPI2001-3801, and by the "Conselleria de Cultura i Educació" (Generalitat Valenciana, Spain) under project GV01-244.

References

[1] R. Marín, P.J. Sanz., J.S. Sanchez, A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality. In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Washington, May, 2002.

[2] R. Marín, P. Vila, P.J. Sanz, A. Marzal. "Automatic Speech Recognition to Teleoperate a Robot via Web". In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS). Lausanne, October, 2002.

[3] R. Marín, J.S. Sanchez, P.J. Sanz. “Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System”. In Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA). Washington, May 2002.

[4] G. T. McKee, The Development of Internet-Based Laboratory Environments For Teaching Robotics and Aritificial Inteligence. In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Washington, May, 2002.

[5] P.J. Sanz, A. P. del Pobil, J. M. Iñesta, G. Recatalá. “Vision-Guided Grasping of Unknown Objects for Service Robots”. In Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 3018-3025, Leuven, Belgium. May 1998.