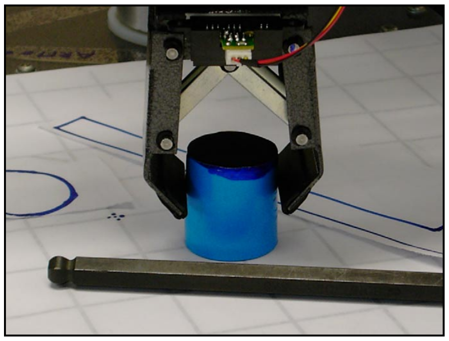
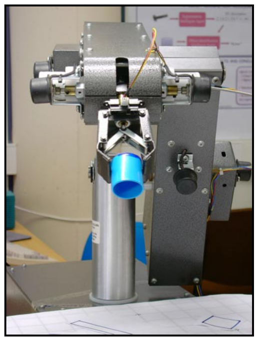
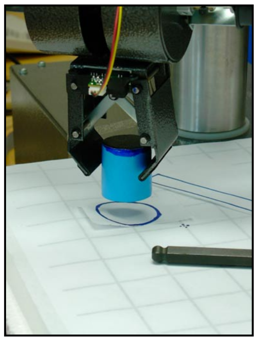
THE UJI ONLINE ROBOT: A DISTRIBUTED ARCHITECTURE FOR PATTERN RECOGNITION, AUTONOMOUS GRASPING AND AUGMENTED REALITY

Ph.D. Thesis
Jaume I University
Engineering and Computer Science Department
Robotic Intelligence Laboratory
20/3/2002

Author:	**Raúl Marín Prades**
Advisors:	**Pedro J. Sanz Valero**
	José S. Sánchez Garreta
	Ángel P. del Pobil

EL ROBOT VIA WEB DE LA UJI: ARQUITECTURA DISTRIBUIDA USANDO RECONOCIMIENTO DE PATRONES, AGARRE AUTÓNOMO, Y REALIDAD AUMENTADA

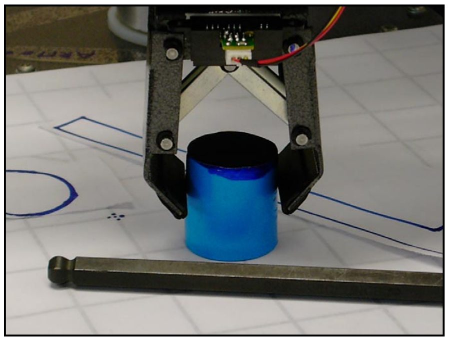
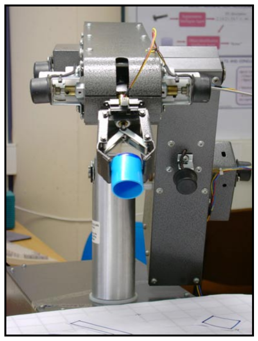
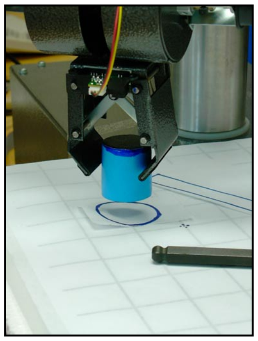
Tesis Doctoral
Universidad Jaume I
Departamento de Ingeniería y Ciencia de los computadores
Laboratorio de Robótica Inteligente
20/3/2002

Autor: Raúl Marín Prades

Directores: Pedro J. Sanz Valero

José S. Sánchez Garreta

Ángel P. del Pobil

BOARD OF EXAMINERS & ASSESSMENT

This book presents the thesis defended at the Jaume I University the 24/5/2002, which obtained a final assessment of "SOBRESALIENTE CUM LAUDE POR UNANIMIDAD" (First-Class Cum Laude Unanimously). Moreover, a second board of international examiners gave the accreditation to obtain the title of "EUROPEAN DOCTORATE".

For related information like videos, pictures and so on, you can access the following address:

http://rmarin.act.uji.es

Castellón (Spain), 2006 © Copyright
Jaume I University of Castellón
ISBN: 84-8021-350-7

“Pa la mama, papa, Evelyn, lo manyo i manya”

RESUMEN / MOTIVACIÓN

La presente tesis doctoral ha sido desarrollada en el ámbito del Laboratorio de Robótica Inteligente de la Universidad Jaume-I de Castellón. Sus objetivos, por tanto, se enmarcan dentro de las líneas y proyectos de investigación de este laboratorio que se ocupan de los diversos aspectos que se refieren al desarrollo de sistemas dotados de inteligencia robótica.

A grandes rasgos, el trabajo ha consistido en el diseño e implementación de un sistema telerobótico completo que permite controlar los movimientos de un robot manipulador via *Web*, utilizando para ello especificación de tareas de muy alto nivel (e.g. "Coge la llave allen"). En síntesis, la idea consiste en que la inteligencia necesaria para efectuar una operación es proporcionada en parte por el usuario y en parte por el robot, con lo cual la comunicación hombre-máquina se efectúa a un nivel muy superior (i.e. más cercana al lenguaje natural). Esto se consigue gracias al control semi-autónomo que posee el sistema al estar el usuario integrado en el bucle de control. Otro beneficio de este tipo de interacción usuario-robot será evitar la "fatiga cognitiva" del operador, típica de estos sistemas telerobóticos. Básicamente aparecen dos tipos de situaciones:

(1) Modo "EN-LÍNEA" ("on-line"). Cuando el robot está físicamente accesible, el usuario toma el control del robot.
(2) Modo "FUERA-DE-LÍNEA" ("off-line"). Por el contrario, en aquellos casos en los que el robot no está accesible (e.g. porque hay otro usuario conectado, etc.), se ofrece la posibilidad de programar tareas en un escenario 3D virtual, posibilitando la ejecución de las mismas sobre el robot real cuando éste se encuentre accesible de nuevo.

PRINCIPALES APORTACIONES DE LA TESIS

Como características innovadoras de la tesis podríamos destacar las siguientes:

1. **Reconocimiento de objetos:** Para poder especificar instrucciones del tipo "Coge el destornillador", es necesario la incorporación de un sistema de reconocimiento de objetos a partir de imágenes de una cámara. De esta manera, el sistema tiene capacidad de diferenciar unas tijeras de un destornillador, y actuar convenientemente.
2. **Reconocimiento y síntesis de voz:** Por otro lado, se ha incluido un módulo tanto de reconocimiento como de síntesis de voz, el cual permite controlar el robot de forma interactiva por medio de un micrófono y unos auriculares. De esta manera, el modo de interacción del usuario con el sistema se desarrolla de una manera mucho más natural.
3. **Aprendizaje incremental:** Para que funcione el reconocimiento de objetos es necesario que el robot tenga un entrenamiento previo. Al mismo tiempo, hemos permitido que esta fase de aprendizaje pueda ser efectuada por cualquier persona que se conecte al robot por medio de la Web. Técnicas de supervisión de este aprendizaje (reconocimiento automático con rechazo) han sido incorporadas de manera que se pueda evitar la adquisición de conocimiento erróneo.
4. **Agarre basado en visión**: Una vez reconocido un objeto en una escena, es necesario calcular geométricamente los puntos del contorno que aseguren un agarre estable del mismo. Así mismo, el sistema es capaz de proporcionar varias posibilidades de agarre, y es finalmente el usuario quien selecciona el agarre que considera más oportuno. Esta característica incorpora mayor versatilidad en la manipulación y en el control remoto del robot.
5. **Realidad Virtual no inmersiva:** Para solucionar problemas de transmisión de datos por Internet (latencia y retardo asociado a este medio de transmisión), el sistema incorpora una interfaz de usuario basada en realidad virtual no immersiva. De este modo, a partir de las imágenes generadas por las cámaras se construye un modelo 3D virtual del escenario del robot, con el cual

se pueden especificar tareas completas para luego ser confirmadas en el robot real en una única operación. Estas interfaces se conocen en telerobótica como entornos "predictivos".

6. **Realidad aumentada**: A la información real del escenario del robot se le añade información virtual generada computacionalmente con datos imprescindibles para la correcta manipulación de objetos a distancia (i.e. Ayudas a la localización de la garra en la escena, evitar situaciones donde exista oclusión, etc.). En algunas ocasiones el usuario tiene más información accediendo al robot remotamente que viéndolo directamente en el escenario real.
7. **Especificación de tareas (Programación "Off-line")**: El sistema permite especificar tareas completas de ensamblado y manipulación de objetos sin tener conexión real con el robot. Estas tareas pueden ser guardadas para la ejecución posterior en el robot real. La programación de tareas se efectúa en el entorno 3D virtual (modo "off-line").

ACTIVIDAD INVESTIGADORA RELACIONADA CON LA TESIS.

(1) Impacto en la Comunidad Científica

Resultados parciales de esta investigación se han publicado en diversos congresos internacionales, nacionales y revistas de investigación desde el año 1998 hasta la actualidad (e.g. "International Journal DISPLAYS", "IFAC International Conference", etc.). Lo más destacable hasta el momento en la producción científica relacionada con la tesis es, sin lugar a dudas, la aceptación y futura presentación de dos trabajos de investigación en el congreso mejor valorado del mundo en el campo de la robótica. Se trata del "*IEEE International Conference on Robotics and Automation 2002*" que se celebra en Washington DC (USA) en Mayo de 2002. También cabe destacar la publicación de algunos resultados noveles de la tesis en la revista "*International Journal on Robotics an Automation*", la cual sacará a la luz en noviembre del 2002 un resumen de los mejores trabajos en telerobótica por la web (Web Robots).

(2) Proyectos en Marcha

Las futuras investigaciones están enmarcadas dentro del interés propio del laboratorio de Robótica inteligente, el cual tiene como principales líneas de trabajo la robótica de servicios. En este aspecto caben destacar los dos proyectos en los cuales se están desarrollando aportaciones interesantes:

- Proyecto de la Generalitat Valenciana (GV01-244): "Diseño e implementación de un sistema robotizado para la manipulación de objetos en escenarios 3D dinámicos". Financiado para los años naturales 2002 y 2003. Investigador responsable del proyecto: *Pedro José Sanz Valero*

- Proyecto del Ministerio de Ciencia y Tecnología: "Tareas de servicio para un robot manipulador móvil". Financiado para los años naturales 2002-2004. Investigador responsable del proyecto: *Ángel Pasqual del Pobil*

Además, se están iniciando negociaciones con el Centro para el Desarrollo Tecnológico Industrial (CDTI) para ver la posibilidad de extender esta tesis al campo de desactivación de minas, dentro del proyecto "Advanced Global System To Eliminate Anti-Personnel Landmines (Apl)", en el programa europeo EUREKA.

(3) Futuro inmediato.

El futuro inmediato consiste en mejorar el sistema para que funcione en entornos más generales. Para ello, es necesaria la extensión de la tesis a campos tan significativos como el "Visual Servoing", el cual permitiría resolver la problemática asociada a la calibración ojo-mano en entornos menos estructurados y en robótica móvil. De hecho, esta nueva facilidad va a ser estudiada a raíz de la estancia del autor de la tesis en el laboratorio LASMEA de la Universidad Blaise Pascal (Francia) el próximo 1 de Junio del 2002.

POSIBLES APLICACIONES.

Por lo general, la ventaja de los sistemas telerobóticos vía Web radica en que el usuario experto tiene posibilidad de tomar control del robot desde cualquier parte del mundo, lo cual da evidentemente mucha flexibilidad en cuanto a acceso. Por otro lado, el uso de la Web como medio de comunicación supone realizar un esfuerzo extra a la hora de resolver problemas relacionados con el retardo y la latencia. En la medida en que la Internet evolucione hacia un medio de transmisión más rápido y fiable, las aplicaciones de la robótica vía Web serán mayores, ya que el usuario dispondrá de más información del escenario robótico remoto (i.e. sensores de fuerza, visión estereoscópica, etc.). De hecho, cuando en el año 1997 tuvimos la idea de diseñar una aplicación de manipulación robótica por Internet, teníamos confianza en que en unos pocos años el ancho de banda de la Internet (en aquellos tiempos 9600 bps) se multiplicaría enormemente. En la actualidad los sistemas de banda ancha (e.g. ADSL) ya se están usando ampliamente por el público en general, y esto nos hace ser muy optimistas en cuanto a las innumerables aplicaciones robóticas que podremos diseñar en un futuro no muy lejano.

También hay que tener en cuenta que en algunas aplicaciones el uso de la Web como medio de comunicación podría no ser el más adecuado, al menos en primera instancia (i.e. Informática médica, etc.). Para estos casos debemos remarcar que el sistema es capaz de funcionar tanto en modo web como en modo aplicación, con lo cual sería directamente aplicable en entornos donde una red privada es altamente necesaria (mayor ancho de banda, mayor seguridad, menor latencia y retardos, etc.). La única restricción que tenemos hasta el momento para utilizar la configuración de red privada es que el protocolo de transmisión sea TCP/IP. Si esta restricción se cumple el sistema funcionaría sin necesidad de adaptación alguna.

De este modo, hemos hecho una clasificación de las aplicaciones que este proyecto puede tener en diferentes áreas. Algunas de ellas ya se han llevado a cabo (i.e. Educación), y otras podrían efectuarse a medio/largo plazo en función del ancho de banda del medio de comunicación empleado y de los dispositivos de interacción hombre-máquina que se decidan utilizar.

1. Aplicaciones en las cuales el sistema ya ha sido validado:

1.1 Educación: Primeramente la tesis ha sido validada y probada con éxito dentro del marco del "Education & Training". Concretamente, más de ochenta alumnos estuvieron utilizando el sistema telerobótico para llevar a cabo tareas de clasificación y ensamblado de objetos, dentro de las prácticas de laboratorio de la asignatura de Robótica, en el primer cuatrimestre del curso 2001-2002. El resultado ha sido que los alumnos tenían acceso a un laboratorio docente de robótica por Internet que en este caso era tanto virtual como real, ya que el robot era accesible de forma real por el usuario. El sistema telerobótico sigue estando operativo y son bastantes los alumnos que siguen utilizándolo fuera ya del contexto de la asignatura de robótica.

1.2 Entretenimiento: De forma inesperada, nos dimos cuenta que muchas de las personas que se conectaban al robot (incluso una vez finalizadas las prácticas de robótica), lo hacían por entretenimiento. Para ellos era motivador el poder manejar a su gusto un robot desde un ordenador remoto. Incluso han habido personas que han realizado operaciones de manipulación remotas durante bastantes horas seguidas.

2. Aplicaciones a corto-medio plazo:

2.1 Robótica de vigilancia ("Security & Surveillance"): Mediante una adaptación del sistema de manera que sea emplazado sobre una plataforma móvil, en primera instancia podría ser usado como medio de soporte a la vigilancia del interior de edificios. Situando una cámara en la mano del robot podríamos programar el mismo para que reconozca situaciones anómalas (e.g. movimiento de personas en horarios poco frecuentes) y levantara una alarma que automáticamente recibiría el personal de seguridad en un terminal. Una de las ventajas del sistema sería la posibilidad de manipulación, con lo cual el robot podría abrir puertas, tomar el ascensor, etc.

2.3 Robótica de servicios: Tomando como punto de partida la configuración móvil introducida en el punto anterior, se podría hacer un mayor esfuerzo en la manipulación de manera que el robot fuera capaz de moverse por el edificio de profesorado y responder a peticiones de manipulación de los usuarios que se conectan por Internet. En particular, una aplicación en la cual ya se están efectuando algunos trabajos es poder mandar el robot al edificio de la biblioteca, buscar un libro determinado y volver al despacho del profesor con ese libro. Evidentemente, en este problema se deben tener en cuenta muchos factores. Por ejemplo, el robot debería ser capaz de desplazarse desde el edificio de profesorado a la biblioteca, superando obstáculos como rampas, pasos de peatones, etc. Una vez en la biblioteca debería encontrar la estantería concreta donde el libro hipotéticamente estaría almacenado. Finalmente, y este es una de las dificultades más interesantes, el robot haría un reconocimiento de los códigos de barras de los libros para finalmente sacarlo de la estantería.

3. Aplicaciones a más largo plazo:

3.1 Desactivación de minas: Como hemos visto en el apartado anterior, ya se están manteniendo conversaciones con instituciones con la finalidad de adaptar el sistema actual hacia la consecución de una manipulación robótica que permita desactivar una mina. Aprovechando trabajos existentes que permiten situar al robot en el escenario donde la mina está ubicada, el problema consiste en manipular esta mina de forma segura y desactivarla. De momento se necesitará estudiar de forma concisa los diferentes tipos de minas a manipular y las alternativas a seguir, como por ejemplo si dejar que el sistema sea teleoperado (inteligencia proporcionada por el usuario) o si algunas de las acciones se pueden efectuar de manera más supervisada.

3.2 Robótica espacial: Aprovechando que el sistema actual facilita la manipulación en un entorno de comunicación donde existen retardos (i.e. Internet), una configuración parecida podría utilizarse en situaciones donde este retardo es todavía mayor, la Robótica espacial. Además, en estos casos el uso de la Internet como medio de programación, facilita la especificación de tareas de manera colaborativa entre varios investigadores que se sitúan en diferentes partes del globo terráqueo. De hecho, la NASA ha estado trabajando con un sistema similar (WITS) para los nuevos robots que explorarán el planeta Marte en un futuro cercano.

3.2 Medicina hospitalaria: Las técnicas de cirugía utilizando realidad virtual y realidad aumentada se están empezando a imponer como vías fiables hacia la consecución de una cirugía menos invasiva. Aunque este tipo de aplicaciones necesiten un medio de comunicación altamente fiable y con un gran ancho de banda, hay que tener en cuenta que la evolución de la Internet se está llevando a cabo a pasos agigantados, y posiblemente en unos pocos años se puedan implantar aplicaciones donde el usuario pueda sentirse inmerso en un entorno remoto de manera que pueda llevar a cabo tareas de soporte en un quirófano o incluso cirugía. Las ventajas de la Internet como medio de conexión radican en que el cirujano tiene posibilidad de acceder al quirófano desde cualquier punto de la Web. Aunque esto pueda parecer ciencia ficción en la actualidad, el tiempo posiblemente nos de la razón, o aún mejor, la realidad puede que supere la ficción (como muchas otras veces ha ocurrido).

***

CONTENTS

ABSTRACT

The thesis has been developed at the Intelligent Robotics Laboratory of the University Jaume I (Spain). The objectives are focused on the laboratory's interest fields, which are Telerobotics, Human-Robot Interaction, Manipulation, Visual Servoing, and Service Robotics in general.

Basically, the work has consisted of designing and implementing a whole vision based robotic system to control an educational robot via web, by using voice commands like "Grasp the object one" or "Grasp the cube". Our original objectives were upgraded to include the possibility of programming the robot using high level voice commands as well as very quick and significant mouse interactions ("adjustable interaction levels"). Besides this, the User interface has been designed to allow the operator to "predict" the robot movements before sending the programmed commands to the real robot ("Predictive system"). This kind of interface has the particularity of saving network bandwidth and even being used as a whole task specification off-line programming interface. By using a predictive virtual environment and giving more intelligence to the robot supposes a higher level of interaction, which avoids the "cognitive fatigue" associated with many teleoperated systems.

The most important novel contributions included in this work are the following:

1. *Automatic Object Recognition*: The system is able to recognize the objects in the robot scenario by using a camera as input (automatic object recognition). This feature allows the user to interact with the robot using high level commands like "Grasp allen".

2. *Incremental Learning*: Due to the fact that the object recognition procedure requires some kind of training before operating efficiently, the UJI Online Robot introduces the Incremental Learning capability, that means the robot is always learning from the user interaction. It means the object recognition module performs better as time goes by.

3. *Autonomous Grasping*: Once an object has been recognized in a scene, the following question is, how can we grasp it? The autonomous grasping module calculates the set of possible grasping points that can be used in order to manipulate an object according to the stability requirements.

4. *Non-Immersive Virtual Reality*: In order to avoid the Internet latency and time-delay effects, the system offers a user interface based on non-immersive virtual reality. Hence, taken the camera data as input, a 3D virtual reality scenario is constructed, which allows specifying tasks that can be confirmed to the real robot in one step.

5. *Augmented Reality*: The 3D virtual scenario is complemented with computer generated information that helps enormously to improve the human performance (e.g. projections of the gripper over the scene is shown, superposition of data in order to avoid robot occlusions, etc.). In some situations the user has more information by controlling the robot from the user interface (web based) than seeing the robot scenario directly.

6. *Task specification*: The system permits specifying complete "Pick & Place" actions, which can be saved into a text file. This robot programming can be accomplished using both, the off-line and the on-line mode.

7. *Speech recognition/synthesis*: To our knowledge this is the first online robot that allows the user to give high-level commands by using simply a microphone. Moreover, the speech synthesizer is integrated into the predictive display, in such a way that the robot responds to the user and asks him/her for confirmation before sending the command to the real scenario.

As explained at Chapter I, the novel contributions have been partially published in several scientific forums (journals, books, etc.). The most remarkable are for example the acceptance of two papers at the *IEEE International Conference on Robotics and Automation 2002,* and the publication of an extended article at the *Special Issue on web telerobotics of the International Journal on Robotics and Automation* (November 2002).

We have proved the worth of the system by means of an application in the Education and Training domain. Almost one hundred undergraduate students have been using the web-based interface in order to program "Pick and Place" operations. The results are really encouraging (refer to Chapter VII) for more details. Although we are referring to the project as *"The UJI Online Robot"*[1]*,* in the Education and Training domain *"The UJI Telerobotic Training System"* term has been used instead.

Further work is planned to focus on applying Remote Visual Servoing techniques in order to improve the actual system performance. This would avoid having to spend long nights calibrating the robot and the cameras, as well extending the system capabilities to work on less structured environments. In fact, the author is planning a research placement at the LASMEA Laboratory of the Blaise Pascal University (France) for this summer, in order to get expertise in this subject and to obtain some interesting research results.

[1] In the robotics community the term "*online robot*" refers to web-based telerobots. On the other hand, "*Web robot*" is used by the Internet community to refer to webpage searchers.

ACKNOWLEDGMENTS

The following is a list of people who has collaborated in making this project a reality. Thanks to all of you! I really appreciate your help.

- Thanks very much to Dr. Pedro J. Sanz, for his daily effort supervising and suggesting new important ideas to the thesis. In some difficult situations I have felt he was more motivated by the project than myself, which has given me the courage to carry on with the work. Moreover, his expertise in Visually Guided Grasping and Robotics in general has been fundamental for the project.
- Thanks to Dr. J. Salvador Sánchez for contributing excellent ideas about Object Recognition, including the Incremental Learning material. These ideas have enabled us to obtain a system that responds the user commands in a more natural manner. Thanks as well for helping so much to correct the thesis document.
- Thanks to Prof. Dr. Ángel P. del Pobil for his expert advise in those situations where good ideas were needed so much. Thanks as well for your support as thesis advisor and Robotics Lab leader.
- Thanks to the Gabriel Recatalá for helping me so much when I began to work at the Intelligent Robotics Laboratory. Moreover, thanks for providing me the first software that enabled me to control the local movement of the Mentor robot. It was really the first step towards the construction of the whole Telerobotic System.
- Thanks to Antonio Morales for helping me to understand the Grasping Determination software, and for giving his authorization to include some of his figures at Chapter VI.
- Thanks to Enrique Cervera for providing help when needed in order to set up the UJI Telerobotic System webpage.
- Thanks to the Intelligent Robotics Lab for allowing me to develop such a system that has motivated me so much. I have really enjoyed it!
- Thanks to Roger Esteller for his collaboration when deadlines where so close and I did not have enough time to implement what was needed. Thanks as well for his contribution studying the way 3D Studio can be used to construct Java3D models.
- Thanks to Antonio J. Yepes for his collaboration setting up the first experiments with Java 3D simulation.
- Thanks to Paulo Vila for his collaboration in implementing the voice recognition grammar, and for working with me side by side (some times until 4 o'clock in the morning) to put all this work together.
- Thanks to Oscar Coltell for providing me exclusive use of a computer which enabled me to obtain results for the Object Recognition Module.

- Thanks to the Engineering and Computer Science Department, and especially to the Technology and Computer Science Area, for giving support in those situations where I needed so much time to complete this research.

- Thanks to the Spanish Ministry and the Generalitat Valenciana for giving funds to accomplish part of this research.

- Thanks to Henrik Christensen and Philippe Martinet for suggesting several interesting ideas at the EURON Summer School in Robotics in order to improve and enhance the project capabilities.

- Thanks to my family, and especially to Evelyn, for understanding the reason why I was unable to attend some of my obligations during the writing period. This whole thesis is dedicated to all of you! Congratulations!

Chapter I INTRODUCTION

First of all, the chapter describes the research motivation, which concentrates on the design of advanced multimedia interfaces in order to control robots over the Internet. Secondly, it describes the state of the art for telerobotic and teleoperated robots through the web, focusing on the manipulation applications.

After that, the *Problem Formulation* is presented, which consists of applying a set of advanced technologies (Multimedia, User Interfaces, Web Programming, etc) in order to improve the way humans interact with robots remotely and particularly within the Web Robotics field. The original idea was to improve the human robot interaction used by the existing online robots at that particular moment (Telegarden, Australia's Telerobot, etc.). For example, we thought about allowing the system to manage a simplification of the natural language input (e.g., "Grasp object one", "Grasp allen"). This implied the implementation of an Automatic Object Recognition module, which was (and still is) novel for this kind of applications. Then, we considered the possibility of offering a kind of simulated environment where people could train themselves in order to known how to control the robot capabilities. In fact, an extension to this would be to allow programming in that simulated interface, so that a whole task could be performed later with the real robot.

Finally, an outline of the whole thesis is given, which is complemented with the set of scientific publications (i.e. *journals, books,* etc.) that have been derived from research work.

Topics

1. Motivation: Internet Robots, why are they interesting? The limited bandwidth and time delay effect.
2. State of the Art
3. Problem formulation
4. Outline
5. Scientific Production

Let us first understand the facts, and then we may seek the cause.

(Aristotle).

1. Motivation

Robotics & Multimedia have been my main interest fields from the very beginning. Since 1993 I became very concerned about the importance of multimedia user interfaces and the way they can be applied to different fields like Education and Training, Telecommunications Management, and specially Robotics.

At 1993, I began to design graphical user interfaces for the company GestWin Ltd, where basically I got introduced to the top-down programming. This approach conceives the applications from the user point of view. I realized the user interface describes the overall system functionality, and most of the software success depends on its adequate design.

After that, in 1994 I began to do some research work at the Multimedia Research Group at Jaume I University [MultimediaLab][1]. This research group has been involved in developing tutoring systems for several years with the main objective of improving the teaching-learning capabilities in a university domain. Bearing this in mind we started implementing some multimedia tutorials directed to computer science students, in topics like "*data structures*" and so on [Marín et al., 1997][2]. Then, since the first version of Java (*JDK 1.0*) came up in 1995, we began to design some of these user interfaces to be managed through the web. The World Wide Web provided many advantages for some kind of applications like Education and Training (accessibility, maintainability, cheap connectivity, etc). After that, the first Robotics' tutorial was launched, with the addition of being implemented in Java in order to make it accessible all over the Internet [Sanz et al., 1998][3]. It helps undergraduates students of Robotics course to learn the difficult subjects explained at the classroom.

Then, at 1996 I did a training placement at BYG System Ltd, where I became interested in the importance of applying multimedia techniques in the Robotics field. There, I had the opportunity to become familiar with robotics simulation tools (i.e. "*Grasp2000*") and the way user interfaces can be used to control such systems [BYG][4]. After that I began to get involved with the Robotic Intelligence Research Lab at Jaume I University [RoboticsLab][5].

In 1997 I became part of the Switching and Access R&D Group of Lucent Technologies (*Bell Labs Innovations*) [Lucent][6]. For 3 years I did research and development work applied to

[1] [MultimediaLab]	Multimedia Research Group, Jaume I University of Castellón (Spain), (http://www.gm2.uji.es)
[2] [Marín et al., 1997]	R. Marin, P.J. Sanz., O. Coltell., et al. *Student-teacher communication directed to computer-based learning environments*. Displays, Elsevier Science. Special Issue on Displays for Multimedia (17) pp. 167-178. 1997. Abstract available on (http://www.system-concepts.com/displays/)
[3] [Sanz et al., 1998]	P.J. Sanz, S. Adell, An undergraduate Robotics Course via Web. Teleteaching'98 Conference, a part of the 15th IFIP World Computer Congress. Distance Learning, Training, and Education. Austrian Computer Society (book series of). Edit. by Gordon Davies, pp. 859-867, Viena, Austria. 1998.
[4] [BYG]	BYG System Ltd webpage (http://www.bygsystems.com), Nottingham (UK).
[5] [RoboticsLab]	Robotic Intelligence Lab, Jaume I University of Castellón (Spain), http://robot.act.uji.es.
[6] [Lucent]	Switching and Access R&D Group of Lucent Technologies (Bell Labs Innovations) (http://www.bell-labs.com/).

remotely controlled network devices by using advanced techniques like Java and CORBA [Marin et al., 1998][1][Marin et al., 1999][2]. Since then, the distributed architecture for controlling remotely a robot through the web came up [Marin et al., 1998b][3]. The original idea was defining an advanced user interface that enabled high level interaction with a robot through the web, by using a simplification of the natural language (voice input), object recognition, and learning capabilities. That really was the beginning of the UJI Online Robot System, whose long-term objective is providing students with a complete intelligent tutorial for learning advanced Robotics from anywhere in the Internet.

1.1. Telerobotics vs Teleoperation

Telerobotic systems are interesting because they combine a very fast and precise device (robot) with the intelligence of a Human being (The user is in the loop). Besides this, they permit the user to control the robot from a distance, which makes the system much more challenging (in my opinion) than a traditional one.

First of all we are going to point out the two different kinds of remote controlled systems, teleoperation and telerobotics.

Teleoperation technology supports a form of control in which the human directly guides and causes each increment of motion of the slave (remote controlled robot). Typically the slave robot follows the human motion exactly (within its physical capabilities). In this situation the whole task planning is originated from the human part [Hannaford, 2000][4].

Telerobotics technology implies communication on a higher level of abstraction in which the human communicates goals (high level tasks) and the slave robot synthesizes a trajectory or plan to meet that goal. Telerobotics primarily supports information interaction because of the higher level of abstraction. In both cases, the operator accepts sensor information transmitted from the remote site to explore the remote environment, plan tasks, verify that tasks are completed, and create plans to resolve problems. Basically a telerobotic system allows the execution of tasks by means of high level commands like e.g. "*Grasp the scissors*". In this situation it is the robot which takes care of the way we approach to the object and then calculates the different grasping possibilities to perform the manipulation.

[1] [Marín et al., 1998] R. *Marín, "Aplicaciones Distribuidas Orientadas a Objetos como Medio para conseguir la Calidad Total:* Standard CORBA", Proceedings of *the "VII Jornadas Técnicas de Calidad en Tecnologías de la* Información *(CIECAT'98)", Telefónica I+D,* Madrid, *1998.* (http://www.tid.es/calidad/ciecat/contra.html)

[2] [Marín et al., 1999] R. *Marín, E. Jimenez, "Gestores de Red basados en* CORBA: Un caso real", Proceedings of the IX Telecom I+D Congress, Madrid (Spain), 1999. (http://www.telecomid.com)

[3] [Marín et at, 1998b] "Distributed Arquitecture for a Learning Telerobotic System" Marín R, Recatalá G, Sanz PJ, Iñesta JM and Pobil AP, Proceedings of the 7th European Workshop on Learning Robotics (EWLR-7), Edinburgh (UK), 1998.

[4] [Hannaford, 2000] B. Hannaford. Feeling is Believing: History of Telerobotics Technology: The robot in the garden: telerobotics and telepistemology in the age of the Internet, K. Goldberg (ed.), MIT Press, Massachussetts, 2000

Another significant definition is the one provided by [Sherindan, 1992][1] in terms of *supervisory control.*

The term supervisory control is derived from the close analogy between the supervisor's interaction with subordinate human staff member in a human organization and a person's interaction with "intelligent" automated subsystems. A supervisor of humans gives directives that are understood and translated into detailed actions by staff subordinates. In turn, subordinates collect detailed information about results and present it in summary form to the supervisor, who must then infer the state of the system and make decisions for further action. The intelligence of the subordinates determines how involved their supervisor becomes in the process. Automation and semi-intelligent subsystems permit the same sort of interaction to occur between a human supervisor and the computer-mediated process.

In the strictest sense, supervisory control means that one or more human operators are intermittently programming and continually receiving information from a computer that itself closes an autonomous control loop through artificial effectors and sensors to the controlled process or task environment. In a less strict sense, supervisory control means that one or more human operators are continually programming and receiving information from a computer that interconnects through artificial effectors and sensors to the controlled process or task environment. In both definitions the computer transforms information from human to controlled process and from controlled process to human, but only under the strict definition does the computer necessarily close the loop that excludes the human, thus making the computer an autonomous controller for some variables at least some of the time.

Both definitions have something in common, a *telerobot* allows the execution of high level tasks permitting the operator to interact in a more friendly manner. This is the case for example of our telerobotic system, which is able to execute commands like "*Take the cube*", in a way very similar to natural language. Moreover, for those situations where the user needs precise control over the robot position, the system enables the operator to move every robot joint separately, which means it can be considered a teleoperated robot too.

[1] [Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

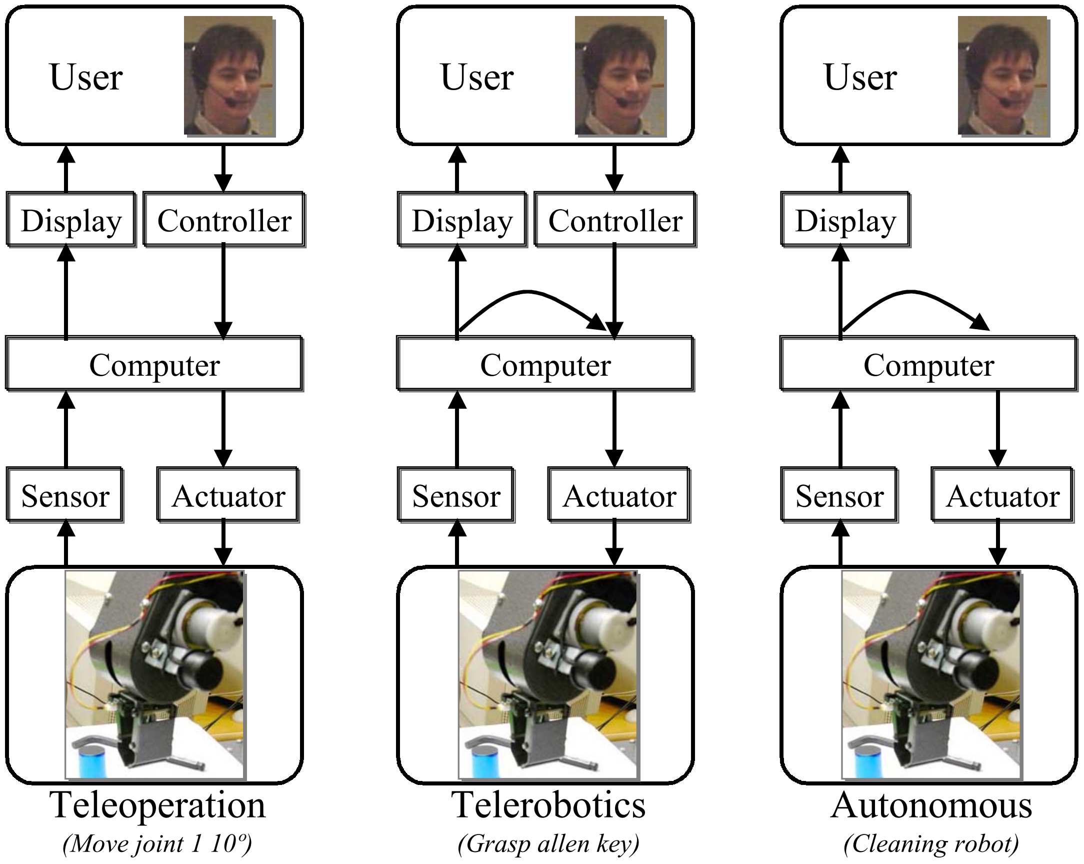


Figure I-1. Teleoperation, Telerobotics and Autonomous systems based on their control configuration (adapted from [Sherindan, 1992][1]).

As can be seen at Figure I-1 the control configuration is determined by the kind of tasks and the level of abstraction that should be used. For example, if we need a cleaning robot that is constantly looking for garbage and bringing it to the correct place, the design of an autonomous system is well justified. For other situations like e.g. classifying and assembling objects on a board in terms of the user constraints, it is the operator who must make the decision. Hence, depending on the way in which the system is used, and the level of performance of which the robot is capable, the system to be used could be considered as teleoperated or telerobot.

1.2. Internet Robots: Why are they interesting?

So far we have explained our interest in telerobotics and high-levels of interaction with the robot. A next step consists of selecting the communication channel that will allow operators to get access to the robot.

At this point we are going to present the reasons that have motivated us to use the Internet as a medium to interconnect the client user interface to the remote robot. Obviously, it has many advantages and some drawbacks that will be treated too (e.g. security restrictions and time delay).

[1] [Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

First of all we are going to enumerate the *advantages*:

1. *Easy* and *cheap* connection. The Internet provides one of the cheapest ways to interconnect two ore more computers and particularly several computers with a device (e.g. robot). It is very easy to find an Internet point, basically the only thing we need is a telephone connection point and a modem. Besides this, as many TCP/IP based technologies are evolved (e.g., sockets, RMI, CORBA, etc) for the Internet case, it is very easy to find the proper software utility needed to implement a particular application.

2. *Maximum accessibility rate*: If we are looking for a medium that enables many people to get access to a device from any location, the Internet is a very good alternative. Even some companies have introduced Internet based solutions in order to allow their staff to work from home. Other people e.g. those who travel around the world for business and are always connected to their companies' information by means of a mobile phone and a laptop computer. The Internet essence could be expressed as "*Get what you need anytime, anywhere*".

3. *Inexpensive distribution*: Once a release of a given company software (e.g. Adobe PhotoShop 6.0) has been launched the problem consists of making that software reach their customers. A good way could be, for example, creating several thousands of CD-ROMS and bring them to the software stores. Another possibility would be publishing this software into a web page and (previous payment or not) let users download the software to be installed in the local hard disk. This second situation supposes the distribution cost is almost null. The third alternative is avoiding this downloading and installation procedure and automatically launching the software from a web page. This situation has many advantages because users do not have to worry about versions and software installations. Everything is made transparent to him. Obviously this situation will only be possible if user has enough Internet bandwidth.

4. *Inexpensive maintainability*: Since the Internet, and particularly the World Wide Web, defines a very cheap distribution medium, it means the maintenance is going to be cheap too. For example, if we realize there is a error to be corrected on the telerobotic training system program, as the code is stored in a web server it is very easy to correct it and publish it again on the web page. All this would be accomplished in a way which is easy for the operator to understand.

5. *Advanced programming techniques*: As the Internet has acquired such importance, companies like e.g., Sun Microsystems have become concerned about the situation and have developed programming languages like *Java* that enable the design of advanced user interfaces. Thus e.g. Java includes routines to create 3D virtual worlds (*Java 3D*), advanced images manipulation (*Java Advanced Imaging*), and speech synthesis and recognition technology (*Java Speech*). On the other hand, in order to develop a distributed system (e.g. telerobotic system), tools like RMI (*Remote Method Invocation*), Jini and CORBA allow easy connection to remote services (e.g. robot control). Some of these tools (e.g. *CORBA* and *RMI*) enable any software (programming language independent) access to any service on any operative system (platform independent).

6. *Possibility of sponsorship (e.g. Telegarden and Robotoy)*: Due to the fact that the huge numbers of people have access to the Internet, there have been situations were robotics companies have sponsored online robots. Connecting, for instance, an AdeptOne robot to the Internet means thousands of people will be able to control it and see how it operates. It could be considered like a demonstration platform for a particular robot.

As pointed out in [Hannaford, 2000][1], the Internet functions essentially the same as any other communication link. The important difference is that the Internet may allow *public participation*.

On the other hand, by using the Internet as an access medium it means some drawbacks must be taken into account too:

1. *Not valid for real-time systems (limited bandwidth and delays)*: At the present time, depending on several factors like e.g., distance between operator and device (robot), or work load of the network at a particular moment, the available Internet bandwidth varies considerably. Moreover, time delays come up which mean that operator will take some milliseconds to realize that is the real robot situation. On the other hand, from the moment the operator commands a task, it will take some time for the robot to execute it. This important factor means that some compensating solutions must be taken into account, as explained at next point.

2. *Security restrictions*: For those situations where not everybody should get access to a given robot (e.g. robotic research in an industrial laboratory), some access control should be implemented. If a telerobot is created for a specific task and is intended for operation only by a predefined set of trusted, authorized users, then issue of authenticity can be addressed by standard Internet security measures. In these situations the Internet obliges users to implement security restrictions, by means of password authentication or other possibilities.

1.3. Internet delays and bandwidth limitations: How to manage it

Let us consider a communication scheme that imparts a several-second round-trip delay and limits the available bandwidth. One must be concerned with the effects of that constrained channel and perceptible delay [Sayers, 2001][2].

The low bandwidth means one can no longer simply digitalize and transmit images from the remote site as before. One simple technique is to choose a small image size and compress each image (for example, using JPEG compression). In this case, processing time is traded off (to encode and decode each image) in return for lower transmission bandwidths. For example, for the UJI Online Robot situation, we send compressed JPG files for the active cameras once every 4 seconds (by default). The camera images are 320x240 pixels size at gray scale, which means 4K amount of data.

Besides information capacity, a second key property of the communication channel is time delay, the time required for a message to arrive at the destination. Since a telerobotic system requires communication in two directions, we must consider delays introduced by both links. These delays introduce dissociation between the operator's commands and the action of the server robot. More significantly, delay of the returning sensor information is also introduced

[1] [Hannaford, 2000] B. Hannaford. Feeling is Believing: History of Telerobotics Technology: The robot in the garden: telerobotics and telepistemology in the age of the Internet, K. Goldberg (ed.), MIT Press, Massachussetts, 2000

[2] [Sayers, 2001] C. Sayers. Fundamentals of Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001

between the operator's action and the resulting sensory feedback displays of the robot response [Hannaford, 2000][1].

When the time delay between operator action and perceived response is greater than about 250 ms it becomes cognitively apparent to the user. The behavior of operators under these conditions was first studied by Sherindan and Ferrely [Sherindan et al, 1963][2].

There are two sources of this time delay. Most fundamentally, there is the delay due to the speed of light. While significant in contemplated space applications, this delay is rarely dominant except for interplanetary applications such as the Sojourner Mars Rovers. The time delay due to light-speed between the earth surface and a communication satellite in geosynchronus orbit can be significant however; about 250 milliseconds per round trip.

The second major source of delay is that introduced by switches in computer networks. These delays are caused by the processing of information in computers at nodes in the network. As explained in [Sayers, 2002], this delay was measured in 1996 (as well as loss rate of UDP packets) for different distances on the Internet (see Figure I-2).

Distance scale	*Loss rate (%)*	*Min*	*Delay (ms) avg*	*max*
Room	0.0	2.0	3.0	18.0
Dept	0.0	2.5	3.5	23.0
Campus	0.0	3.0	3.0	5.0
Planet	11.5	278.3	421.0	746.8

Figure I-2. Internet time delays (Adapted from [Sayers, 2001][3])

To cope with the delay, one convenient approach would be to do nothing and rely on the operator to adapt. Although attractively simple, such an approach will not work well. Consider the case where the operator notices the remote manipulator is about to hit a wall. The operator reacts immediately, but, by the time the image of the robot is seen, it is already several seconds too late. The collision happens long before the operator's desperate attempts at recovery make their way to the distant site. Knowing that risk, operators quickly adopt a "*move and wait*" strategy: making a small, relatively safe motion, then waiting to see that it succeeded before moving again [Ferrell et al., 1967][4]. This is frustratingly slow.

Coping with the delay requires a two-pronged approach. First, at the remote site, one must add some intelligence to the robot controller [Lindsay, 1992][5][Sherindan, 1992][1][Stein, 1994][2].

[1] [Hannaford, 2000] B. Hannaford. Feeling is Believing: History of Telerobotics Technology: The robot in the garden: telerobotics and telepistemology in the age of the Internet, K. Goldberg (ed.), MIT Press, Massachussetts, 2000

[2][Sherindan, 1963] T. B. Sherindan and W. R. Ferrell, Remote Manipulative Control with Transmission Delay, IEEE Transactions on Human Factors in Electronics HFE-4(1963): 25-29

[3] [Sayers, 2001] C. Sayers. Fundamentals of Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001

[4] [Ferrell et al., 1967] W. R. Ferrell and T.B. Sherindan. Supervisory control of remote manipulation. IEEE Spectrum 4 (10): 81-88, October 1967.

[5] [Lindsay, 1992] T. S. Lindsay, Teleprogramming-remote site task execution. Ph.D. dissertation. The University of Pennsylvania, Philadelphia, 1992.

One does not need a very smart robot, but one needs enough intelligence to imbue it with a sense of self-preservation and the ability to protect itself and its environment. The controller need not know what has gone wrong or how to fix it; it has to react fast enough and protect itself and the environment for long enough so that the distant human operator has time to react. By providing the slave robot with additional information a priori, one can mitigate the effects of the delay.

By making the robot "*smarter*" (e.g. ability to manipulate unknown objects), one can also raise the level of communication between operator and remote robot, enabling the operator to assume a more supervisory role. Even if the communications were not delayed, this is still beneficial. It frees the operator from the need to command every detail of the operation at the remote site, and helps enormously to avoid the "*cognitive fatigue*" effect [Murphy, 2000][3].

The second approach for dealing with a constrained communications link is to supply the operator with a more sophisticated interface [Hirzinger, 1993][4][Sayers, 1996][5]. For example, one could give the operator a simulation of the remote robot and allow for interaction with it. That is called a *predictive display* [Berjczy et al., 1990][6]. It allows the operator to see the effect of commands well before they are actually executed by the remote robot. If the remote environment were sufficiently constrained, then one could even consider simulating the dynamic motion of objects at the remote site [Hirzinger, 1993]. By providing the operator station with a model of the remote robot that reacts instantaneously, one can largely insulate the operator from the effects of the communications delay. Even if the link delay were not an issue, there may still be some benefit in providing the operator with a simulated view of the remote site. Such a system affords viewpoints unavailable from real cameras, it is unaffected by poor visibility at the remote site, and it frees the operator from the need to work at exactly the same speed as the remote site. The operator may experiment with actions off-line, or work more quickly or more slowly using the real remote robot [Conway et al., 1990][7].

---

[1] [Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

2 [Stein, 1994] M. R. Stein. Behavior-Based Control For Time Delayed Teleoperation. Ph.D. thesis, University of Pennsylvania MS-CIS-94-43, 1994.

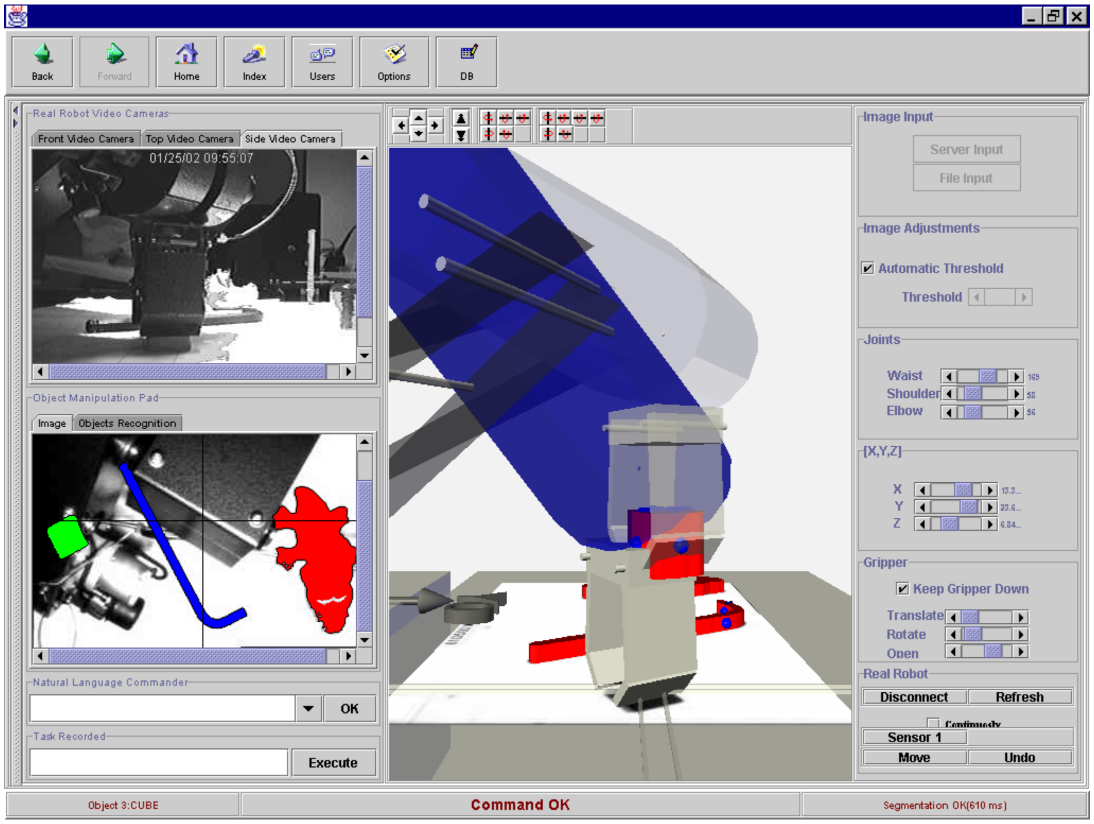
3 [Murphy, 2000] R. R. Murphy, Introduction to AI Robotics, The MIT Press, 2000.

4 [Hirzinger, 1993] G. Hirzinger. ROTEX the first robot in space. In International Conference on Advanced Robotics, pp. 9-33, 1993.

5 [Sayers, 1998] C. Sayers, Remote Control Robotics. New York: Springer Verlag, 1998.

6 [Bejczy et al., 1990] A. K. Bejczy, Steven Venema, and W. S. Kim., Role of computer graphics in space telerobotics: Preview and predictive displays. In Cooperative Intelligent Robotics in Space, pp. 365-377. Proceedings of SPIE, vol. 1387, 1990.

7 [Convay et al., 1990] L. Conway, R. Volz and M. Walker. Tele-autonomous systems: Projecting and Coordinating Intelligent Action at a Distance. IEEE Robotics and Automation Journal, vol. 6. No. 2, 1990.



Object 3:CUBE

Command OK

Segmentation OK(610 ms)

Figure I-3. Predictive display for the UJI Online Robot. It predicts not only the next robot position but the objects' too.

As can be seen at Figure I-3 the interface shows from the cameras input how the real robot has actually grasped the cube object. Then, in the 3D virtual environment we can see two robots, one that monitors the real robot (down position), and the other that is predicting a new movement in order to elevate the cube. The predictive robot is shown in gray in order to distinguish it from the real one.

Another interesting feature that significantly improves the operator performance is the "*augmented reality*" functionality, which enhances the real information from the robot scenario with some virtually generated data that is crucial for a proper human-robot interaction (see Chapter II for more details).

A number of working systems have been constructed for performing teleoperation via a constrained communications link. All these approaches have something in common: by providing additional information at each site, one can reduce the effects of delayed communications. The difficulty is that the bandwidth is also constrained. Thus, one must make tradeoffs in sending information between sites. Although having additional information may save time, sending that information itself consumes time. Deciding which information to send between sites, and when to send it, is perhaps the most difficult and challenging part of efficiently controlling a remote robot via the Internet.

1.3.1. Detecting unpredictable situations

Now, a predictive display exists for the operator. One can send commanded motions to a remote robot and, using knowledge of robot kinematics and dynamics, one can make the robot move to where it was commanded. It might appear that one has a complete system, but unfortunately this is less than halfway. What has been neglected is that the real world is far from perfect. Regardless of how exacting computer prediction might become, it will never be able to anticipate everything that the real world contains. Thus, merely sending commands to the remote robot is not sufficient. One must have ways to detect, and recover from, unexpected events [Sayers, 2001][1].

In the UJI Online Robot, for example, errors are detected by comparing predicted sensor readings (encoded within commands from the operator station) with actual sensor reading (measured as the slave executes each command). Any significant discrepancy causes the slave to pause and an error to be sent to the operator station.

Each error is expensive. When the error occurs, a signal has to propagate from the remote site back to the operator, and then the operator's new posterror commands have to travel to the remote site. Thus one loses at least on round-trip communications delay. In practice, the cost of the error is much greater. It takes time for the operator to diagnose each problem. Even worse, an error will often result in a discrepancy between the operator station model of the remote site and the real remote site. Regardless of how perfect remote site sensors might become, resolving discrepancies in the world model will always consume valuable time and bandwidth resources. Because of these costs, it is worthwhile making some effort to try and avoid errors or at least mitigate their effects.

One should aid the operator in performing actions that are likely to be successful, while dissuading the operator from actions that are likely to fail. This is the case, for example, for the predictive visually guided grasping at the UJI Online Robot (see Figure I-3). The simulated robot will only consider as stable the grasp points generated by the grasping determination module. Of course, an operator could try to grasp an object by some other point manually. For this case the prediction will consider the grasping to have failed before sending the movements to the real robot.

The Internet will certainly improve. As the delays decrease and bandwidths increase, online robots will become much more like conventional teleoperation systems. The unique characteristics of online robots will then be dictated by their intended audience and the limited operator interface available via a computer screen.

Having an intelligent human "*in the control loop*" is a powerful ally. Designs should strive to make the best use of that intelligence, with the operator solving the difficult problems, while the system takes care of the details.

[1] [Sayers, 2001] C. Sayers. Fundamentals of Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001

2. State of the Art

Many different telerobotic systems have been reported since Goertz introduced the first mechanical teleoperator at the Argonne National Laboratory four decades ago [Goertz, 1954][1]. Fundamentally, the purely mechanical devices were limited to about 5 meters separation between the two sides. Furthermore, this separation had to be fixed at the time of installation, neither side could be moved relative to the other. Newer applications demanded that the remote side be able to move (for example along the length of a particle accelerator). The response was to develop an electronic version of the mechanical manipulator [Hannafor, 2000][2].

As explained by Taylor and Dalton (see [Taylor et al., 2000][3]) telerobotics finds application in hazardous environments, control of underwater remotely operated vehicles, space systems, nuclear industry applications, education and training, and recently entertainment too. Besides this, with the appearance of such a widely accessible and low-cost interface called World Wide Web the devices like robots and other resources can be available to a broad range of users at an accessible price. This characteristic is very convenient for projects developed with a limited budget and where the number of different people interacting with the system is crucial in order to train the robot or get other kinds of information from users, like for example, response times.

For those systems where the accessibility and the low cost are more important than a limited and irregular bandwidth the World Wide Web interface is the answer. The Internet's key advantage is the flexibility of where the operator can gain access to communication.

One of the essential points in the design of a web based telerobotic system is the definition of the Human-Robot interaction. In most telerobotic systems, user interaction is still very computer-oriented, since input to the robot is accomplished by filling in forms or selecting commands from a panel. Very little attention has been paid to more natural ways of communication such as natural language, or gestures.

Another essential point is making the robot able to learn from the operator and from practice. It means the system's robustness and effectiveness increases as time goes by as the process is performed again and again.

This section describes the most significant telerobotic systems through the web, focusing on the ones that enable *manipulation*. Between them we could for example introduce the "*Telegarden*", that allows planting and watering a garden by using the Internet. The second one is the "*Australian Telerobot through the web*", that allows the users to manipulate little pieces of wood in order to make their own constructions. Some more manipulators and mobile robots are presented.

1 [Goertz, 1954] Goertz, R. and Thompson, R. (1954): Electronically Controlled Manipulator: Nucleonics, 1954.

2 [Hannaford, 2000] B. Hannaford. Feeling is Believing: History of Telerobotics Technology: The robot in the garden: telerobotics and telepistemology in the age of the Internet, K. Goldberg (ed.), MIT Press, Massachussetts, 2000

3 [Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

After that, we will focus on the user interface, and the way the operator is able to control the task specification process. It will allow us to compare the different means of interaction offered to the user. Besides this, we will discuss some different kinds of interaction that can be used in order to control a robot, depending on the application (e.g. "*Predictive systems*"). The general idea is obtaining a set of parameters that will be used later on to establish a formal comparative between the different systems.

2.1. The Mercury Project

2.1.1. Application

The Mercury Project is known as the first telerobotic system on the web. In August 1994, Ken Goldberg's team mounted a digital camera and air jet device on a robot arm so that any-one on the Internet could view and excavate for artifacts in a sandbox located in the Robotics laboratory at the University of Southern California [Goldberg et al., 1995][1].

The primary goal was creating a very reliable system capable of operating 24 hours a day and surviving sabotage attempts. Moreover, the system had to be low in cost, because of budget limitations. The secondary goal was creating an attractive web site that encourages us-ers to repeat their visits [Goldberg et al., 2001][2].

2.1.2. Hardware setup

The project used the IBM SR5427 robot built by Sankyo in early 1980. Its four -axis design is common in industrial assembly for pick-and-place operations because it is fast, accurate, and has a large 2.5D workspace.

To allow users to manipulate the remote environment, the initial plan was to place a simple gripper at the end effector. Anticipating incompetent and potentially damaging use, com-pressed air was used instead as the medium for manipulation.

The CCD camera was an EDC 1000 from Electrim Inc., chosen based on size and cost. Image data was sent from the camera back through a custom serial line to a video capture card. The camera image had a resolution of 192 by 165 pixels with 256 shades of gray, which were truncated to 64 to reduce transmission rates. Standard florescent fixtures primarily illuminated the robot workspace.

2.1.3. The User Interface

To facilitate use by a wide audience of nonspecialists, all robot controls were made avail-able via Mosaic, the only browser available at that time. The 2D interface matched the work-space of the robot, so users could move by clicking on the screen with a few buttons for out-

[1] [Goldberg et al., 1995] K. Goldberg, M. Mascha, S. Gentner, J. Rossman, N. Rothenberg, C. Sutter, and J. Widgley, "Beyond the Web: Manipulating the Real World", Computer Networks and ISDN Systems Journal 28, no 1 (December 1995).

[2] [Goldberg et al., 2001] K. Goldberg, Roland Siegward (ed.), Beyond Web Cams: An introduction to Online Robots, MIT Press, Massa-chusetts, 2001.

of-plane effects. The user interface centered on a bitmap that was called the "control panel" as shown in Figure I-4.

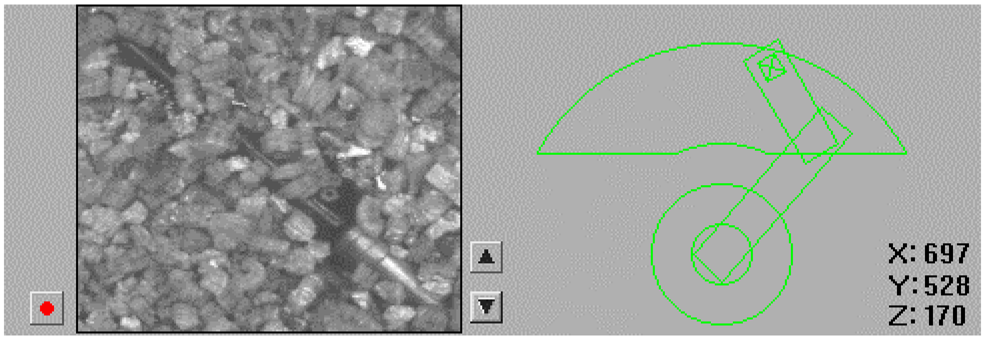


Figure I-4. The control panel. It shows the schematic top view of the real robot position and the camera image from the robot end-effector. Up/Down buttons are included for Z motion of the camera, and the round button is used to blow a burst of compressed air into the sand (adapted from http://www.usc.edu/dept/raiders/).

When the operator clicked on the control panel using the mouse, the XY coordinates were transferred back to the server, which interpreted them to decode the desired robot action. This action can be:

1. A global move to center the camera at XY in the schematic workspace
2. A local move to center the camera at XY in the camera image
3. Moving the camera to one of the two fixed Z heights
4. Blowing a burst of compressed air into the sand directly below the camera

2.1.4. Communications Technology

Network throughput averaged 20 Kbytes/sec, which was poor compared with 500 Kbytes/sec that can be achieved between two workstations in close proximity on the campus network. As one of the operating systems used was MSDOS, it forced the implementation of busy/wait cycles to obtain concurrence between the robotic/camera operations and the networking duties [Goldberg et al. 2000b][1].

To maintain compatibility with the widest possible set of user platforms, only the HTTP protocol that was standard in 1994 (version 1.0) was used, thus restricting the project to the use of single still images.

2.1.5. Statistics

The Mercury Project was online for seven months starting in late August 1994. During this time it received over 87.700 user connections. An off-center air nozzle caused sand to accu-

[1] [Goldberg et al. , 2000b] K. Goldberg, S. Gentner, C. Sutter, and J. Wiegley, "The Mercury project: a feasibility study for Internet robots: design and architecture of a public teleoperated system accessible by anyone on the Internet at any time", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

mulate in one side of the box, so the sand had to be periodically regroomed. Other than a few random downtimes, the robot was online twenty-four hours a day.

2.2. The Telegarden

2.2.1. Application

This telerobotic installation allows WWW users to view and interact with a remote garden filled with living plants. Members can plant, water, and monitor the progress of seedlings via the movements of an industrial robot arm. The garden is actually real, and located in the Ars Electronica Museum in Austria.

2.2.2. Hardware setup

This project uses an Adept-1 robot and a multitasking control structure so that many operators can be accommodated simultaneously[1].

Planting and watering are accomplished with a series of pneumatic actuators. A pump can be activated to spray one tablespoon of water into the garden at a specified point. When planting is requested, the system lowers a "foot" via pneumatic cylinder and digs a small hole, then a seed is sucked up from a tray at the edge of the garden, the seed is then dropped into the hole. Finally the foot is lowered again to cover the hole, and a burst of water is applied. As of Feb 1996, well over 1000 seeds have been planted.

2.2.3. The User Interface

Users are presented with a simple interface that displays the garden from a top view, the garden from a global composite view and a navigation and information view in the form of a robot schematic. By clicking on any of the images one commands the robot to move to a new absolute location or one relative to where they just were.

[1] The Telegarden web page: http://telegarden.aec.at/

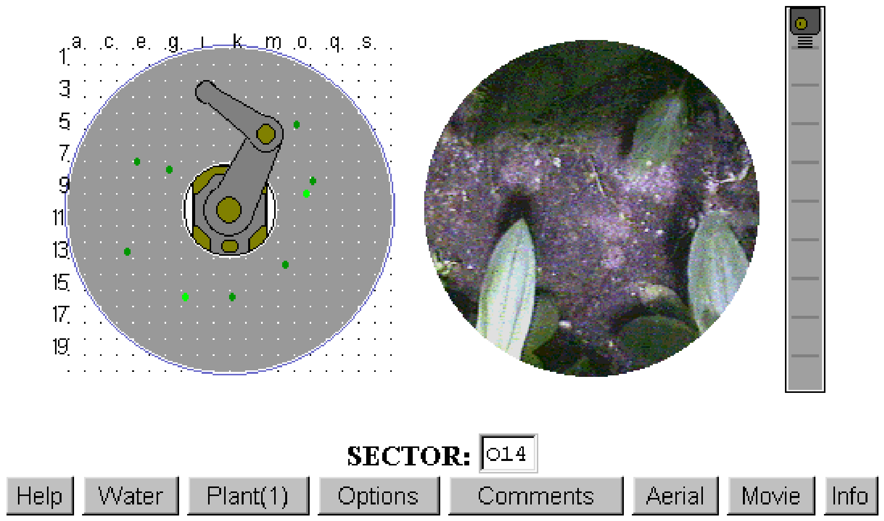


Figure I-5. Telegarden User Interface. It allows moving the camera by clicking the position over the schematic top view of the real robot. After that several possibilities are offered, like planting a seed, watering of moving the camera up and down. (adapted from http://www.usc.edu/dept/garden/).

The robot, upon completion of the move, will return a refreshed image of the garden. In this manner one can explore the entire garden remotely using simple mouse clicks.

The robot performs user requests on a first come first served bases. Because of this multiplexing of the robot more than one user can be within the garden at once.

By using the member tracker overlay one can not only see who but where other members are within the garden. If they wish to communicate with these other members they can enable or enter the village square. The village square is a public message chat system where people within the garden can discuss subjects of their choice.

To water the garden users align the camera image over the section of the garden to water and press the water button. This will command the robot to release a small squirt of water over the area in view. To plant a seed a user is first requested to find a spot that is relatively empty (there are no restrictions to where one can plant) and then asked to press the plant button. This will cause the robot to poke a small hole in the ground, proceed to the seed bowl, suck up a seed and desposit it back into the previously dug hole.

2.2.4. Communications Technology

The WWW system is driven mostly from an Intel Pentium based workstation equipped with an image capture board, running the Linux operating system and the NCSA HTTPD web server software. Robot control is achieved through a serial port connection to the robot controller. The interactive WWW interface options are created via a large custom Common Gateway Interface program written in C. It maintains databases of user and system status, log and chat comments, movie creation etc.

Whereas the Mercury Project required operators to wait up to 20 minutes on a queue for a 5-minute turn to control the robot, the Telegarden applies multi-task to allow "simultaneous" access. After X-Y coordinates are received from a client, a command is sent to move the robot arm, a color CCD camera takes an image, the image is compressed based on user options such as color resolution, lighting, and size, and then returned to the client. All coding is done in C and the 1-second cycle time of the Adept arm allows us to respond to requests in 5-10 seconds depending on load.

Another novel feature is the ability to request a video "time-lapse" of a single view over several weeks or a "pan" along a user specified trajectory. The video is recorded during non-peak hours and compressed into mpeg late at night. User's can vote on their own and other's videos in a Movie page.

2.2.5. Statistics

The Telegarden was developed at the University of Southern California and went online in June 1995, and it has been online since in August 1995 [Goldberg et al., 1996][1]. In its first year at USC, over 9000 members helped cultivate. In September 1996, the Telegarden was moved to the new Ars Electronica Center in Austria.

Other interesting robots have been implemented at the University of Southern California, for example, the Ouija [Goldberg et al., 2000][2], that allows a distributed user group to simultaneously teleoperate and industrial robot arm via the web. As far as the authors know, it is the first collaboratively controlled robot on the Internet[3].

2.3. The Australian Telerobot through the web

2.3.1. Application

The Australian Telerobot enables users to manipulate different kind of objects over a board, following a "Pick & Place" strategy. The challenge with the interface design is to provide enough information to make interpretation easy while minimizing transmitted data and maintaining a simple layout. Ways of doing this include restricting the number of degrees of freedom that an operator controls as well as making use of Java and JavaScript technology [Taylor, 2000][4].

2.3.2. Hardware setup

Image feedback is provided by a number of Pulnix TM-6CN cameras connected to a single Matrox Meteor II frame grabber. New images are taken whenever a new robot state is received

[1] [Goldberg et al., 1996] K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, C. Sutter, and J. Wiegley. A Telerobotic Garden on the World Wide Web. SPIE Robotics and Machine Perception Newsletter 5(1). Reprinted in SPIE OE Reports 150, June 1996.

[2] [Goldberg et al., 2000] K. Goldberg, S. Bui, B. Chen, B. Farzin, J. Heitler, D. Poon, R. Solomon and G. Smith. Collaborative Teleoperation on the Internet. In IEEE International Conference on Robotics and Automation (ICRA). San Francisco, CA. April, 2000.

[3] Ouija 2000 web page: http://ouija.berkeley.edu

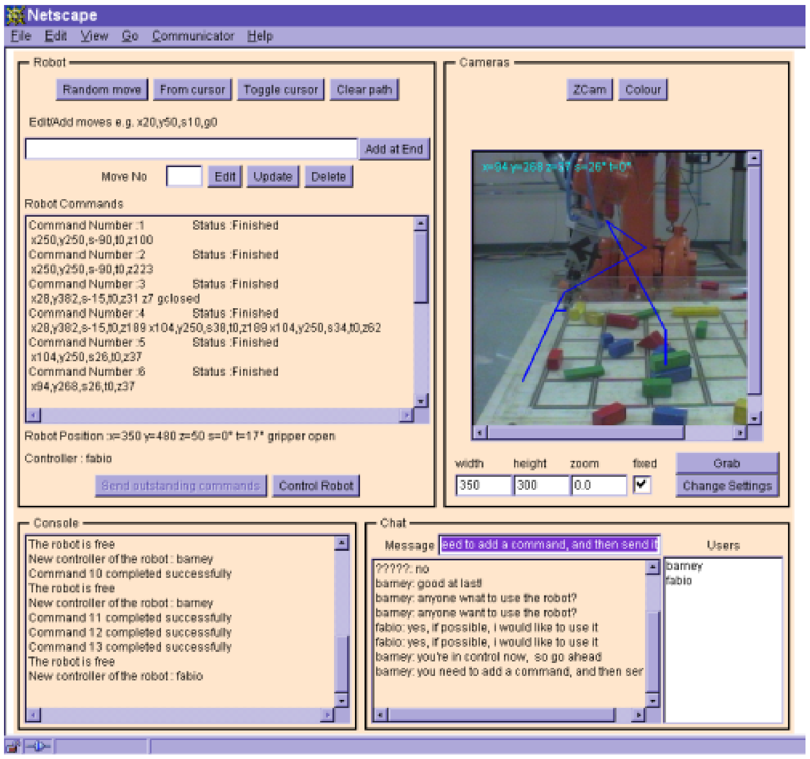
[4] [Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

on the robot channel, or when a user explicitly requests that a new image be taken. The position of the cameras are: one camera looking down the X axis; one looking down the Y axis; one looking down on the table; and one on the third joint of the robot arm. The X and Y cameras are calibrated. Calibration is performed by recording the position of the robot in camera images for twenty different robot poses, the camera matrix is then obtained.

The server side is running on a Pentium II with windows NT. The only extra card required is the framegrabber.

2.3.3. The User Interface

The Interface is divided in four parts (see Figure I-6): The top left corner provides the user with the latest images of the workspace. Different camera views can be selected. Commands can be applied to change the image size and quality of the images. For calibrated cameras various objects can be overlaid. These include the last n positions of the robot and an augmented reality cursor for specifying robot pose.



```
Edit/Add moves e.g. x20,y50,s10,g0

Robot Commands
Command Number :1          Status :Finished
 x250,y250,s-90,t0,z100
Command Number :2          Status :Finished
 x250,y250,s-90,t0,z223
Command Number :3          Status :Finished
 x28,y382,s-15,t0,z31 z7 gclosed
Command Number :4          Status :Finished
 x28,y382,s-15,t0,z189 x104,y250,s38,t0,z189 x104,y250,s34,t0,z62
Command Number :5          Status :Finished
 x104,y250,s26,t0,z37
Command Number :6          Status :Finished
 x94,y268,s26,t0,z37

Robot Position :x=350 y=480 z=50 s=0° t=17° gripper open
Controller : fabio

x=94 y=268 z=37 s=26° t=0°

width  height  zoom  fixed
350    300     0.0   ✓
```

```
The robot is free
New controller of the robot : barney
Command 10 completed successfully
The robot is free
New controller of the robot : barney
Command 11 completed successfully
Command 12 completed successfully
Command 13 completed successfully
The robot is free
New controller of the robot : fabio
```

```
Message: eed to add a command, and then send it

?????: no
barney: good at last!
barney: anyone what to use the robot?
barney: anyone want to use the robot?
fabio: yes, if possible, i would like to use it
fabio: yes, if possible, i would like to use it
barney: you're in control now,  so go ahead
barney: you need to add a command, and then ser

Users
barney
fabio
```

Figure I-6. The Australian Telerobot's User Interface. It allows moving the robot to specific 3D positions and execute teleoperated command based actions over the robot. (adapted from http://www.usc.edu/dept/raiders/).

The console area provides a single visible place to show all error and status messages. Robot errors and any unexpected communications situations are all displayed in the console.

Robot commands follow a simple scripting syntax, allowing multiple waypoints to be specified in a single command. These commands can be typed directly (e.g., "*moveL targetpos, vslow, fine, grippertool\Oobj:=table;*") or generated from an augmented reality cursor displayed on the camera images. The cursor maps the five degrees of freedom of the robot to the two degrees of freedom of the image. To generate commands from the cursor, the user moves it to the required position and presses the "from cursor" button. The command is then pasted to the edit window where user can make changes before queuing and sending it to the real robot [Dalton, 1998][1].

2.3.4. Communications Technology

Formerly, the Common Gateway Interface (CGI)[2] was used in order to interconnect the client user interface to the server side. CGI processes are launched by a web browser in response to certain HTTP requests, and the CGI result is sent as the HTTP response [Dalton, 2001][3]. The main drawback is that after a client request has been processed by a server, there is no way for the server to contact the client. The client must always initiate contact. It means to receive constant updates, the client must poll the server at regular intervals. Polling is inefficient as requests must be made even when there are no changes and the server must handle these requests using resources to process them.

The next step at the Australian Telerobot was redesigning the user interface by means of Java[4]. With the introduction of Java, the code can be executed on the client side, whereas previously all code had to be in the CGI process. Java is being used by a number of online telerobotics projects, including e.g. the NASA pathfinder interface [Backes et al., 1998][5], the UJI Online Robot [Marín et al., 2002][6], etc.

Using Java applets, a client can provide both a more sophisticated interface and use its own protocol to communicate with the server. It means the limitations of HTTP can be overcome. With a constant connection between client and server, both sides can communicate immediately when new information needs to be transmitted. For telerobotics, this ability for server initiated conversation is important, as changes in the state of the robot and workspace needs to be uploaded to the client with minimal delay.

The protocols used to communicate between the Java applet and the server at the Australian Telerobot are currently two. One uses XML[7] and its communication protocol, and the other delegates the task to Java's Remote Method Invocation (RMI)[8].

[1] [Dalton, 1998] B. Dalton, H. Friz,, and K. Taylor. Augmented reality based object modelling in internet robotics. In Matthew R. Stein, ed., SPIE Telemanipulator and Telepresence Technologies V, vol. 3524, p. 210-217, Boston, 1998.

[2] The Common Gateway Interface. http://www.w3.org/CGI

[3] [Dalton, 2001] B. Dalton. A Distributed Framework for Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001

[4] The Java Sun webpage. http://java.sun.com

[5] [Backes et al., 1998] P. G. Backes, K. S. Tao, and G. K Tharp. Mars pathfinder mission Internet-based operations using WITS. In proceedings of IEEE International Conference on Robotics and Automation (ICRA), p. 284-ed.291, May 1998.

[6] [Marín et al., 2002] R. Marín, P.J. Sanz., J.S. Sanchez, A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[7] Extensible markup language (xml) 1.0. http://www.w3.org/TR/1998/REC-xml-19980210

[8] Java Remote Method Invocation. http://java.sun.com/products/jdk/rmi/index.html

The XML approach is very useful for Java applets and cross-language use, and it was used as a result of problems with RMI support in standard web browsers. Internet Explorer 4.0 does not support RMI by default, and although Netscape Navigator supports RMI, there are some little troubles using RMI callbacks. Let us note these problems can be sorted out by the installation of the last Java-Runtime (*Java Plug-in 1.3.x*). For example, for the UJI Online Robot, as the installation of the Java Plug-in and the Java3D runtimes is a requirement, it means the RMI can be used broadly.

There are a number of disadvantages in the use of XML. First, new data types must have a defined XML translation, which must be known by both, client and server side. Second, XML is extremely verbose and can therefore increase bandwidth requirements. This could be reduced by compressing the data before sending, but at the cost of extra processing time.

The RMI approach is more simple, because all the serializing and deserializing of objects is handled by RMI itself, no data formats need to be defined. Moreover, it also enables the implementation of callbacks, and can be easily integrated with CORBA.

As introduced in [Dalton, 2001][1], the Australian Telerobot team is planing to include a CORBA distributed framework, similar to the one offered by the UJI Online Robot [Marín et al., 2002][2]. This would maximize cross platform and language interoperability.

2.3.5. Statistics

Registrations to the system were averaged 23 a week over a period of 34 weeks, which gave a database of 794 registered operators as of October 1997 [Taylor, 2000][3].

2.4. Some other examples of Internet robots

Apart from the Telegarden and the Australian Telerobot, there have been reported many other Internet robots. For a detailed description please refer to the Nasa Space Telerobotic Program[4]. Some of them are not intended to be used over the WWW, but anyway they present some interesting ideas that must be taken into account. Moreover, as introduced before, our interest field is remote manipulation over the Internet, so the following description is basically focused on that area.

A very interesting system is the one reported by [Lloyd, 1997][5] at the University of British Columbia. A central problem in model-based telerobotic technology is obtaining and main-

[1] [Dalton, 2001] B. Dalton. A Distributed Framework for Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001

[2] [Marín et al., 2002] R. Marín, P.J. Sanz., J.S. Sanchez, A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[3] [Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[4] NASA Space Telerobotic Program. http://ranier.hq.nasa.gov/telerobotics_page/realrobots.html

[5] [Lloyd et al., 1997] J. E. Lloyd, J. S. Beis, D. K. Pai, D. G. Lowe. Model-based Telerobotics with Vision.K. In proceedings of IEEE International Conference on Robotics and Automation (ICRA), p. 1297-ed.1304, April 1997.

taining accurate models of the remote site. The system facilitates this using a fast gray-scale vision system at the remote site, which can recognize objects of known type and return their spatial positions to the operator. Basically, it takes a 2D image and finds edges for every object on the scene. Computing geometrical properties of those edges the system is able to recognize an object as belonging to a particular class. This work focuses on 3D model construction from vision, enabling the previous simulated interaction through an Internet connection (not web based). The UJI Online Robot offers a model 3D construction from vision too, which has the particularity of allowing the manipulation of not only known objects, but unknown too, thanks to the inclusion of a grasp determination and execution algorithm. Besides this, it allows many other advantages like e.g., interacting by using voice as input.

Between our Spanish colleagues a simulated robot using Java 3D technology has been reported too by the University of Tenerife (Spain)[1]. Basically this prototype enables the user to control virtually a generic Puma robot by specifying the joint values or the world gripper position. No 3D modeling from vision is reported. Neither object recognition nor grasp determination is implemented. However, it is a good example of what can be accomplished by means of the Java Technology. Moreover, the documentation provided is very complete and gives many techniques details than can be useful for some other Web based robots.

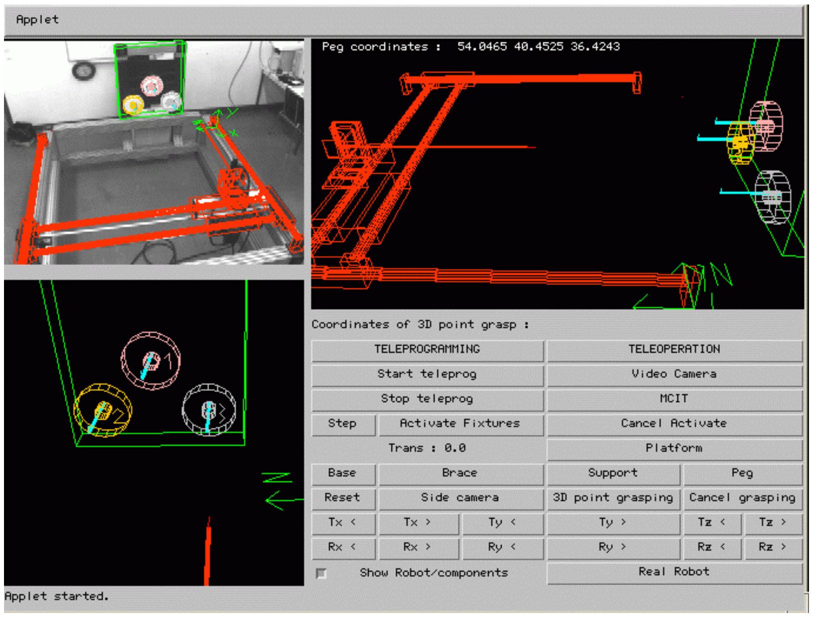
Another project to be taken into account is VISIT, which uses the predictive approach in order to communicate user interface and robot with existing time delay [Kosuge et al., 2001][2]. The system is designed based on advanced robotic tecnologies, such as computer vision, advanced motion control and so forth, so that the operator can execute a task in a remote site by utilizing a simple mouse. The computer vision module extracts object characteristics from a camera input, in order to incorporate them to the predictive model. No object recognition is presented.

The ARITI[3] system also presents a display interface enabling any person to remotely control a robot via the Web. A mixed perceptual concept based on a Virtual Reality (VR) and Augmented Reality (AR) technologies is part of the control user interface, allowing one to easily perform a task and describe the desired environment transformation that the robot has to perform (see Figure I-7).

[1] (http://www.cyc.dfis.ull.es/simrob/)

[2] [Kosuge et al., 2001] K. Kosuge, J. Kikuchi, and K. Takeo, "VISIT: A Teleoperation System via the Computer Network" Beyond webcams, and introduction to online robots, MIT Press, Massachussetts, 2001.

[3] (http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/)



Peg coordinates : 54.0465 40.4525 36.4243

Applet started.

Figure I-7. The ARITI user interface (adapted from http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/)

In Chapter VIII, a comparative analysis of the most significant systems explained in this section is presented. This analysis is focused basically on the user interface, which is, in our opinion, one of the most important components of a telerobotic system via the web.

Although our main interest relates to remote manipulation via the Web, we are also interested in extending the project to Mobile Robotics via the Internet. In this subject several novel contributions have been made that allow various mobile robots to be controlled via Web, and we consider them important references for further work in this subject [Simmons et al., 2001][1][Saucy et al., 2001][2][Siegwart et al., 2001][3].

3.Problem Formulation

Once we had studied the different online robots that allowed remote manipulation, we considered it would be very convenient to improve the way users controlled the robot movements. With this consideration in mind and taking into account the limitations of current online robots, we started working on the UJI Online Robot project, at late 1997.

[1] [Simmons et al., 2001] R. Simmons, R. Goodwin, S. Koenig, J. O'Sullivan, and G. Armstrong. Xavier: An Autonomous Mobile Robot on the Web: Beyond webcams, and introduction to online robots, MIT Press, K. Goldberg and R. Siegwart (ed.), Massachussetts, 2001.

[2] [Saucy et al., 2001] P. Saucy, F. Mondada. KhepOnTheWeb: One Year of Access to a Mobile Robot on the Internet: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001.

[3] [Siegwart et al., 2001] R. Siegwart, P. Balmer, C. Portal, C. Wannaz, R. Blank, and G. Caprari. RobOnWeb: A Setup with Mobile Mini-Robots on the Web: Beyond webcams, and introduction to online robots, MIT Press, K. Goldberg and R. Siegwart (ed.), Massachussetts, 2001.

The robot should include "*adjustable autonomy"*, which means the user should be able to adjust the level of interaction of the robotic agent [Murphy, 2000][1]. For some situations, high level commands like for example, "Grasp the allen", should be permitted. Besides this, user could be interested in programming tasks in a more accurate and personal manner, so that a *teleoperation* mode should be included as complement to the *supervised* one.

Having all this in mind, a set of important requirements must be taken into account:

1. *Distributed architecture:* As the robots (initially just one) are going to be accessible through the web and multiple users can have access to them in on-line and off-line mode, a distributed architecture should be conceived that enables such functionality. Some modules will have to be replicated for every robot. However, others will be able to be shared by the different manipulators. Moreover, in order to avoid situations where some servers become overloaded, some kind of flexibility should be present that let the programmer to launch the different software agents in different hosts in a simple and easy manner (adjustable architecture).

2. *Computer vision:* In order to allow user to specify robot actions by referencing directly the objects in the scene, first step would be having a computer vision procedure that takes care of the camera, illumination, scene segmentation and extraction of object characteristics in general. Although the robot scenario is going to be quite structured, having a robust vision system is very important in order to assure a proper performance, independently of illumination variations, camera calibration, etc.

3. *Automatic Object Recognition:* In order to allow the user to specify tasks in a more natural manner (i.e. "Grasp screwdriver"), the computer vision information must be used in order to identify the existing objects in the scene. Taking into account that the system must respond as fast as possible and we must take care of the Internet latency effect, a compromise between velocity (simplicity) and accurateness must be assumed. Once an image from the real scenario has been acquired, the object recognition module must generate a proper result very rapidly. Moreover, as multiple users are going to interact with the system in a concurrent manner (one on-line and others off-line), this module must be strategically situated in order to avoid the whole system to be overloaded.

4. *Autonomous Grasping:* Up to now we are able to identify the scene objects from the camera inputs. But, what about the object manipulation? In order to perform a grasping operation over a scene object a module that identifies the possible stable grasping points is needed. Otherwise, the user would have the responsibility of teleoperating the whole robot action in order to perform the proper manipulation. Of course some times it would be necessary (i.e. overlapped objects, etc.), but the idea is allowing a high level command (i.e. "Grasp {object}") that works well for the majority of the situations.

5. *Speech Recognition:* The idea of having a set of high level commands to teleoperate the robot would become even more interesting if these actions could be commanded using a microphone and headphones. There already exist some commercial speech recognition packages that work well for very specific problems (like ours). The interesting fact

1 [Murphy, 2000] R. R. Murphy, Introduction to AI Robotics, The MIT Press, 2000.

would be integrating this module with the whole telerobotic system in order to allow the user to teleoperate the robot via web by using the voice.

6. *3D Predictive Display (Virtual & Augmented Reality):* Finally, in order to avoid the Internet latency effect on the user, we have considered interesting including a 3D predictive display that enables knowing the robot commands effects before launching them over the real robot. In fact, this functionality can be used as well as an off-line task specification scenario. More over, some important capabilities should be taken into account in order to improve the user interaction performance. They are including a user interface with virtual and augmented reality features, which are interaction facilities that help the user to specify tasks in a more accurate manner.

7. *Project Validation (Education and Training):* It must be taken into account that the project does not conclude by obtaining some interesting research conclusions. The real challenge consists of making all this work together in order to be used as a real tool in the Education and Training Domain. Things must work properly for the majority of the situations. The system must be robust enough to be available to the Robotics undergraduate students almost 24 ours a day. The original idea consists of implementing a Telerobotic Training System that would assist undergraduate students to comprehend the theoretical concepts in the Robotics field. Moreover, the system should include a web-based access to one or more real robots in order to allow students to program their robot tasks concurrently and remotely. As the number of students is so high (around one hundred) and the robots available in the laboratory are just three, some kind of off-line programming mode should be included. Hence, a student could program the robotic tasks by using a virtual environment and then, once the robot is accessible, check if they work on the real scenario.

The UJI Online Robot Project is known at the Jaume I University and in some research publications as *The UJI Telerobotic Training System,* because it is being used as a way for students to learn Robotics (Virtual Laboratory). This is the reason why at the following chapters both terms will be used to refer to the same system.

4.Outline

The thesis document is organized in the following chapters:

1. *Chapter II-Overall System Description*: This chapter introduces the experimental setup (robot scenario) and gives an overall description of the main system capabilities (i.e. object recognition, etc). Then, the description explains the system's features from the user point of view. Hence, the multiple ways of interaction as well as the 3D virtual and augmented reality facilities are presented. Finally, examples of programming "Pick & Place" operations are given for both, the on-line and the off-line interaction modes.

2. *Chapter III-System Architecture:* This chapter gives low level details on the way modules have been distributed in order to allow the system to perform properly. Thus, the hardware and software architecture are explained in detail. Finally, explanations of extending the system to teleoperate multiple robots are given, which are very important in

order to identify which modules must be duplicated and which could be shared (*multi-robot architecture*).

3. *Chapter IV-Automatic Object Recognition:* This chapter explains the procedure followed from the time the scene image is captured from the camera until every object is recognized as belonging to a certain class (i.e. Allen key, screwdriver, etc.). Information about the image processing technique, object descriptors, classification methods and learning procedure is given in detail. Finally, a set of experiments demonstrates the configuration that gives robust results conforming to the computing time requirements.

4. *Chapter V-Autonomous Grasping:* This chapter explains the low level details of the Grasping Determination procedure, so that the best grasping points to manipulate an object conforming to the stability requirements can be established. Secondly, it describes the way the autonomous grasping functionality has been integrated into the UJI Online Robot, for both, the on-line and the off-line approaches.

5. *Chapter VI-Speech Recognition:* This chapter explains the design and implementation details of the Speech Recognition module, so that commands can be given to the robot by using a microphone and headphones. The way this module fits into the System Architecture is explained in detail too.

6. *Chapter VII-The System In Action:* The system has been used as a tool for undergraduate students to program "Pick and Place" robotic tasks using the Web as accessing medium. Details of the experiment are given. Moreover, as the students have been practicing with other online robots (i.e. Telegarden, Australian Telerobot, etc.), their opinions are presented as a means of comparative analysis. Results are really encouraging!

7. *Chapter VIII-Conclusions:* This chapter is organized in two main parts. Firstly, a list of the novel contributions to the web based telerobotics is given. Secondly, the further work is presented, which fundamentally is focused on the inclusion of "Remote Visual Servoing" techniques to the UJI Telerobotic Training System.

8. *Appendix I-Technical Specifications:* This appendix gives technical details of the robots used for this project.

9. *Appendix II-Implementation Details:* As many advanced software techniques (i.e. CORBA, Java3D, etc.) have been used for the implementation of this project, some low-level details are given in order to understand how they can be used in order to implement such a telerobotic system.

10. *Appendix III-Frequently Asked Questions:* Experts in the Robotics field have been formulating very interesting questions that, in some situations, have allowed us to improve the system capabilities. Some of them are really interesting.

5.Scientific Publications

Some of the results and methodologies presented at this thesis have been accepted and published in several scientific publications (journals, books, etc.):

1. (*International Journal, 2002*) R. Marín, P.J. Sanz, A.P. del Pobil. *Teleoperated Robot System Via Web: The UJI Telerobotic Training System.* accepted for publication in the Special Issue on Web-based Robotics and Automation of the International Journal of Robotics and Automation, 2002.

2. (*International Journal, 1997*) R. Marín, P.J. Sanz, O. Coltell, J.M. Iñesta, F. Barber, and D. Corella *Student-Teacher Communication Directed to Computer-based Learning Environments,* Displays, Special Issue on Displays for Multimedia, Elsevier Science, pp. 167-178, 1997.

3. (*IEEE International Conference, 2002*) R. Marín, J.S. Sanchez, P.J. Sanz. *Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System.* In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

4. (*IEEE International Conference, 2002*) R. Marín, P.J. Sanz., J.S. Sanchez, *A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality.* In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

5. (*Book Chapter, 2000*) R. Marín, P.J. Sanz, A.J. Jimeno, J.M. Iñesta *"Design of a Telerobotic Interface System by using Object Recognition Techniques",* in Advances in Pattern Recognition; SSPR' 2000 & SPR'2000. 8th International workshop on Structural and Syntactic Pattern Recognition, & 3rd International workshop on Statistical Techniques in Pattern Recognition, Francesc J. Ferri, J.M. Iñesta, Adnan Amin, Pavel Pudil. Lecture Notes in Computer Science, Springer-Verlag, New York, 2000.

6. (*Book Chapter, 1998*) Marín R, Recatalá G, Sanz P J, Iñesta J M, del Pobil A P.*, "Telerobotic System Based on Natural Language and Computer Vision",* in Tasks and Methods in Applied Artificial Intelligence; IEA-98-AIE. 11th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, A.P. del Pobil, J. Mira and M. Ali, Lecture Notes in Artificial Intelligence 1416, Springer-Verlag, Berlin, 1998.

7. (*International Conference, 2002*) R. Marín, P.J. Sanz. *Augmented Reality to Teleoperate a Robot through the Web.* In 15th IFAC World Congress b'02, Barcelona (Spain), 2002.

8. (*International Conference, 2002*) R. Marín, P.J. Sanz, *Telerobotic Training System Through the web: A case study,* International Conference on Learning Networks (NL'2002), ref. *100029-03-PS-076, Berlin, 2002.*

9. (*International Conference, 2001*) R. Marín, P.J. Sanz. *The UJI Telerobotic Training System. 1st EURON Workshop on Robotics Education and Training (RET2001),* Alicia Casals, Antoni Grau (ed.), Weingarthen, Germany, 2001.

10. (*International Conference, 2001*) R. Marín, P.J. Sanz. *Telerobotic Training System through the web: Low level Architecture.* In 1st IFAC Conference on Telematics Applications in Automation and Robotics, K. Schilling & H. Roth, IFAC Publications, Elsevier Science Ltd., Weingarthen, Germany, 2001.

11. (*International Conference, 2001*) R. Marín, J.S. Sanchez, P.J. Sanz. *Design and Evaluation of a Telerobotic System with Object Recognition Capabilities,* Applied Informatics-Artificial Intelligence and Applications; Advances in Computer Applications, M.H. Hamza, IASTED International Conference on Robotics and Applications (RA 2001), Florida (USA), 2001.

12. (*International Conference, 2001*) Marín R, Sanz PJ.*, Multimedia Telerobotic Tutorial by Interacting with a Real Robot through the Internet,* Proceedings of the International Workshop on Multimedia Applications 2001, Valencia, Spain, 2001.

13. (*International Conference, 2000*) R. Marín, P.J. Sanz, A.P. del Pobil, *A Web Based Interface for Service Robots with Learning Capabilities,* Workshop W20 at the European Conference on Artificial Intelligence 2000 (ECAI'00), *Workshop Notes: "Service Robotics Applications and Safety Issues in an Emerging Market: Human-Machine Interaction",* Thomas Röfer, Axel Lankenau, Reinhard Moratz (ed.), *pp. 45-48, Hamburg.*, 2000.

14. (*International Conference, 1998*) Marín R, Recatalá G, Sanz PJ, Iñesta JM and Pobil AP, *Distributed Arquitecture for a Learning Telerobotic System,* Proceedings of the *7th European Workshop on Learning Robotics (EWLR-7),* European Machine Learning and Robotics Community & Department of Artificial Intelligence at the University of Edinburgh, pp. 53-63, *University of Edinburgh* , 1998.

15. (*Spanish Conference, 2001*) Marín R, J.S. Sanchez, Sanz PJ., *Distance-based models for remote object recognition,* Proceedings of the "IX Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA2001", Guijón, Spain, 2001.

16. (*Spanish Conference, 1999*) Marín R, Sanz PJ., *A Telerobotic System for Service Applications: A case study,* Proceedings of the "VIII Conferencia de la Asociación Española para la Inteligencia Artificial, *CAEPIA-99 TTIA-99*", Murcia, Spain, 1999.

17. (*Spanish Conference, 1998*) Marín R, *"Aplicaciones Distribuidas Orientadas a Objetos como Medio para conseguir la Calidad Total: Standard CORBA",* VII Jornadas Técnicas de Calidad en Tecnologías de la Información (CIECAT´98). Círculo Español para la Calidad en las Telecomunicaciones, pp. *83-95, Telefónica I+D, Madrid.* , 1998.

6.Summary

This chapter has given an introduction to the technology that is being used until now in Online Robots. The majority of these systems could be improved enormously by experimenting new human-robot interaction techniques, which are, in our opinion, fundamental components of a telerobotic system.

The web-based telerobotic systems are especially interesting because they must take care of the Internet latency and delays effects. Besides this, as the Web allows the operator to be located at any place in the world, these factors have greatly motivated the experimentation of new ways to program online robots using a very high level of interaction.

After formulating the problem, this chapter has given an outline of the whole thesis distribution. Advanced techniques like "Object Recognition", "Speech Recognition", "Autonomous Grasping", and "Virtual and Augmented 3D scenarios" are explained through the thesis in an organized manner.

Finally, in order to illustrate the impact that these experiments have originated in the scientific community, a list of the most significant research publications is given too.

References

[Backes et al., 1998] P. G. Backes, K. S. Tao, and G. K Tharp. Mars pathfinder mission Internet-based operations using WITS. In proceedings of IEEE International Conference on Robotics and Automation (ICRA), p. 284-ed.291, May 1998.

[Bejczy et al., 1990] A. K. Bejczy, Steven Venema, and W. S. Kim., Role of computer graphics in space telerobotics: Preview and predictive displays. In Cooperative Intelligent Robotics in Space, pp. 365-377. Proceedings of SPIE, vol. 1387, 1990.

[BYG] BYG System Ltd webpage (http://www.bygsystems.com), Nottingham (UK).

[Convay et al., 1990] L. Conway, R. Volz and M. Walker. Tele-autonomous systems: Projecting and Coordinating Intelligent Action at a Distance. IEEE Robotics and Automation Journal, vol. 6. No. 2, 1990.

[Dalton, 1998] B. Dalton, H. Friz,, and K. Taylor. Augmented reality based object modelling in internet robotics. In Matthew R. Stein, ed., SPIE Telemanipulator and Telepresence Technologies V, vol. 3524, p. 210-217, Boston, 1998.

[Dalton, 2001] B. Dalton. A Distributed Framework for Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001

[Ferrell et al., 1967] W. R. Ferrell and T.B. Sherindan. Supervisory control of remote manipulation. IEEE Spectrum 4 (10): 81-88, October 1967.

[Goertz, 1954] Goertz, R. and Thompson, R. (1954): Electronically Controlled Manipulator: Nucleonics, 1954.

[Goldberg et al., 2000b] K. Goldberg, S. Gentner, C. Sutter, and J. Wiegley, "The Mercury project: a feasibility study for Internet robots: design and architecture of a public teleoperated system accessible by anyone on the Internet at any time", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Goldberg et al., 1995] K. Goldberg, M. Mascha, S. Gentner, J. Rossman, N. Rothenberg, C. Sutter, and J. Widgley, "Beyond the Web: Manipulating the Real World", Computer Networks and ISDN Systems Journal 28, no 1 (December 1995).

[Goldberg et al., 1996] K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, C. Sutter, and J. Wiegley. A Telerobotic Garden on the World Wide Web. SPIE Robotics and Machine Perception Newsletter 5(1). Reprinted in SPIE OE Reports 150, June 1996.

[Goldberg et al., 2000] K. Goldberg, S. Bui, B. Chen, B. Farzin, J. Heitler, D. Poon, R. Solomon and G. Smith. Collaborative Teleoperation on the Internet. In IEEE International Conference on Robotics and Automation (ICRA). San Francisco, CA. April, 2000.

[Goldberg et al., 2001] K. Goldberg, Roland Siegward (ed.), Beyond Web Cams: An introduction to Online Robots, MIT Press, Massachusetts, 2001.

[Hannaford, 2000] B. Hannaford. Feeling is Believing: History of Telerobotics Technology: The robot in the garden: telerobotics and telepistemology in the age of the Internet, K. Goldberg (ed.), MIT Press, Massachussetts, 2000

[Hirzinger, 1993] G. Hirzinger. ROTEX the first robot in space. In International Conference on Advanced Robotics, pp. 9-33, 1993.

[Kim et al., 1993] W. S. Kim and A.K. Bejczy, "Demostration of a high-fidelity predictive/preview display technique for a telerobotic servicing in space, IEEE Trans. Robotics and Automation 9, no. 5 (October 1993): 698-702.

[Kosuge et al., 2001] K. Kosuge, J. Kikuchi, and K. Takeo, "VISIT: A Teleoperation System via the Computer Network" Beyond webcams, and introduction to online robots, MIT Press, Massachussetts, 2001.

[Lindsay, 1992] T. S. Lindsay, Teleprogramming-remote site task execution. Ph.D. dissertation. The University of Pennsylvania, Philadelphia, 1992.

[Lloyd et al., 1997] J. E. Lloyd, J. S. Beis, D. K. Pai, D. G. Lowe. Model-based Telerobotics with Vision.K. In proceedings of IEEE International Conference on Robotics and Automation (ICRA), p. 1297-ed.1304, April 1997.

[Lucent] Switching and Access R&D Group of Lucent Technologies (Bell Labs Innovations) (http://www.bell-labs.com/).

[Marín et al., 1997] R. Marin, P.J. Sanz., O. Coltell., et al. *Student-teacher communication directed to computer-based learning environments*. Displays, Elsevier Science. Special Issue on Displays for Multimedia (17) pp. 167-178. 1997. Abstract available on (http://www.system-concepts.com/displays/)

[Marín et al., 1998] R. Marín, "Aplicaciones Distribuidas Orientadas a Objetos como Medio para conseguir la Calidad Total: Standard CORBA", Proceedings of the "VII Jornadas Técnicas de Calidad en Tecnologías de la Información (CIECAT´98)", T*elefónica I+D, Madrid, 1998.* (http://www.tid.es/calidad/ciecat/contra.html*)*

[Marín et al., 1999] R. Marín, E. Jimenez, "Gestores de Red basados en CORBA: Un caso real", Proceedings of the IX Telecom I+D Congress, Madrid (Spain), 1999. (http://www.telecomid.com)

[Marín et al., 2001] R. Marín, P.J. Sanz, *Telerobotic Training System through the web: Low Level Architecture,* Proceedings of the IFAC Conference: Telematics Applications in Automation and Robotics, K. Schilling and H. Roth (ed.), Oxford: Elsevier Science Ltd, 2001.

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

[Marín et al., 2002b] R. Marín, J.S. Sánchez, P.J. Sanz. Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System. In Proceedings of the IEEE International Conference on Robotics and Automation, Washington, May, 2002.

[Marín et at, 1998b] "Distributed Arquitecture for a Learning Telerobotic System" Marín R, Recatalá G, Sanz PJ, Iñesta JM and Pobil AP, Proceedings of the 7th European Workshop on Learning Robotics (EWLR-7), Edinburgh (UK), 1998.

[MultimediaLab] Multimedia Research Group, Jaume I University of Castellón (Spain), (http://www.gm2.uji.es)

[Murphy, 2000] R. R. Murphy, *Introduction to AI Robotics*, The MIT Press, 2000.

[RoboticsLab] Robotic Intelligence Lab, Jaume I University of Castellón (Spain), http://robot.act.uji.es.

[Sánchez, 1998] J. S. Sánchez, "Aprendizaje y Clasificación basados en Criterios de Vecindad. Métodos Alternativos y Análisis Comparativo", Castellón (Spain), thesis dissertation, 1998.

[Sanz et al., 1998] P.J. Sanz, S. Adell, An undergraduate Robotics Course via Web. Teleteaching'98 Conference, a part of the 15th IFIP World Computer Congress. Distance Learning, Training, and Education. Austrian Computer Society (book series of). Edit. by Gordon Davies, pp. 859-867, Viena, Austria. 1998.

[Saucy et al., 2001] P. Saucy, F. Mondada. KhepOnTheWeb: One Year of Access to a Mobile Robot on the Internet: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001.

[Sayers, 1998] C. Sayers, Remote Control Robotics. New York: Springer Verlag, 1998.

[Sayers, 2001] C. Sayers. Fundamentals of Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001

[Sherindan, 1963] T. B. Sherindan and W. R. Ferrell, Remote Manipulative Control with Transmission Delay, IEEE Transactions on Human Factors in Electronics HFE-4(1963): 25-29

[Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

[Siegwart et al., 2001] R. Siegwart, P. Balmer, C. Portal, C. Wannaz, R. Blank, and G. Caprari. RobOnWeb: A Setup with Mobile Mini-Robots on the Web: Beyond webcams, and introduction to online robots, MIT Press, K. Goldberg and R. Siegwart (ed.), Massachussetts, 2001.

[Simmons et al., 2001] R. Simmons, R. Goodwin, S. Koenig, J. O'Sullivan, and G. Armstrong. Xavier: An Autonomous Mobile Robot on the Web: Beyond webcams, and introduction to online robots, MIT Press, K. Goldberg and R. Siegwart (ed.), Massachussetts, 2001.

[Stein, 1994] M. R. Stein. Behavior-Based Control For Time Delayed Teleoperation. Ph.D. thesis, University of Pennsylvania MS-CIS-94-43, 1994.

[Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Taylor, 2000] K. Taylor, B. Dalton. Internet Robots: A New Robotics Niche. IEEE Robotics & Automation Magazine, March, 2000.

Part I: System Description

Chapter II OVERALL SYSTEM DESCRIPTION

This chapter focuses on the overall system description throughout the presentation of the user interface, which determines basically the set of features that the system offers to the user. By understanding the human robot interaction, we will be able to comprehend how the system functions as a whole. First of all, the experimental setup is presented. Basically, an educational robot (Mentor) provided with 3 static cameras and a distance sensor configures the server side. This section presents technical information for the robot and cameras, as well as the sensor. Besides this, a description of two other robots (Mitsubishi and AdepOne) is shown due to the fact that the system is able to interact with them in a simulated manner (in the future also online).

Secondly, this chapter focuses on the user interface, which shows the advantages of letting the user decide which form of interaction is the most appropriate for him/her in a given situation. Sometimes, people prefer to use voice commands in order to perform an action (e.g. "Grasp the cube"). On the other hand, there are situations where users prefer to click on a scene object and then select the "Grasp" option directly. The general idea is providing a set of well-organized interaction possibilities and letting the user select which one is more convenient for him at a particular moment. Thirdly, we will focus on the advantages of 3D virtual environments in order to save network bandwidth, letting the user predict their interaction in the 3D model before sending the command to the real robot. Besides this, we will see the possibilities offered by 3D models for developing virtual and augmented reality approaches. Then, we will see how everything has been put together in order to allow an "on-line" robot programming through the web, by having the advantage of using augmented reality in order to better control the robot. Besides this, for those situations where the robot is busy doing a previously programmed task, or it is just being used by someone else, we are still able to accomplish an "off-line" robot programming (simulated), which can be later on executed as a task for the real robot.

Finally, as the project has been applied to the Education and Training domain, some interesting features are presented that have been very useful to undergraduate students during their programming experience with the UJI Telerobotic Training System.

Topics:

1. Experimental Setup
2. Main project capabilities
3. Multiple ways of interaction (Teleoperation vs. Supervised Telerobotics)
4. 3D Models: Virtual and Augmented Reality
5. On-line programming
6. Off-line programming & Task specification
7. Education & Training

The best way to scape from a problem is to solve it (Alan Saporta).

1.Experimental setup

First of all, we are going to present the experimental setup, which is basically the set of hardware devices we have access to from the WWW.

In Figure II-1, the robot scenario is shown, 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). Besides this, they permit the operator to check the proper execution of high level commands like "*Grasp the cylinder*".

Secondly, in order to perform a proper 3D model construction from the top camera input, it is necessary to know the mean height of every object in the scene. A distance sensor installed on top of the gripper is used when this calculation is necessary (e.g., an unknown object is detected). Besides this, it permits the implementation of some collision detection algorithms, giving more intelligence and autonomy to the robot. As explained later, several experiments with distance sensors have been performed, which give an idea of the different situations where this setup works successfully.

An interesting feature is the design of a specific circuit that allows the automatic connection of the environment lights when a remote user gets into the system. This circuit is programmed through the server parallel port. At the moment, the circuit allows remote control of the lamps, and in a near future we will do the same with the cameras and even the robot, in order to avoid them always being switched on.

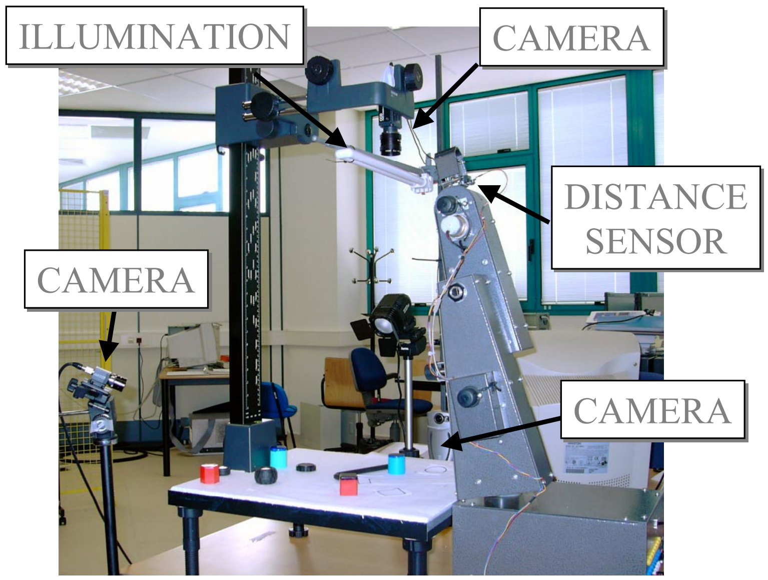


Figure II-1. Experimental setup: Three cameras, a distance sensor, illumination circuit, and a Mentor robot

1.1. Mentor robot description

The Mentor is an educational and inexpensive robot distributed by Alecop[1]. It has five degrees of freedom and direct control over the gripper opening. Here some of the reasons that justify selecting this robot for the project are presented:

- *Low cost:* The Mentor is a practical and economic way to learn how to use robots. Its cost makes it possible to think about providing more than one robot over the Internet, in order to maximize the students chances to control a real robot instead of a simulation. In fact, in our laboratory we have acquired three Mentor robots that are being used simultaneously by the undergraduate students.

- *Five Axes and a Gripper:* Its human-arm configuration is very similar to the one presented into the traditional PUMA, which is widely used in industry. It has five degrees of freedom and direct control over the opening of the gripper. All axes offer closed-loop servo control system, which facilitates enormously its programming. Besides this, as the gripper is already installed and integrated into the robot, it is not necessary to spend time looking for compatible grippers. The robot is fully operational once connected to the computer.

- *Safe and Sturdy:* The youngest students (and some researchers too!) can be left alone to experiment and learn. The Mentor is ideal for letting many people interact with it and resist collisions and remote manipulations. It is very robust and reliable, which means it can be used securely over the Internet and in a student's laboratory.

The Mentor has an articulated arm with joints similar to that of the human arm, of a type that is widely used in industry. Each joint is driven by a DC servomotor with its position monitored by a potentiometer. A built-in controller provides closed-loop control of the system and constantly offers monitoring data to the computer. Computer programming can be accomplished by setting the data for each joint over some configurable memory positions. Alternatively, the motors may be switched off and the Mentor then moved by hand (lead by the nose).

As can be appreciated in Figure II-2, one of the major advantages that the Mentor robot offers is the possibility to interconnect multiple sets of analog and digital inputs. For example, in our case, we have connected several types of distance sensors directly to the robot, which means they can be monitored from the computer and transmitted over the Internet.

[1] (http://www.alecop.es)

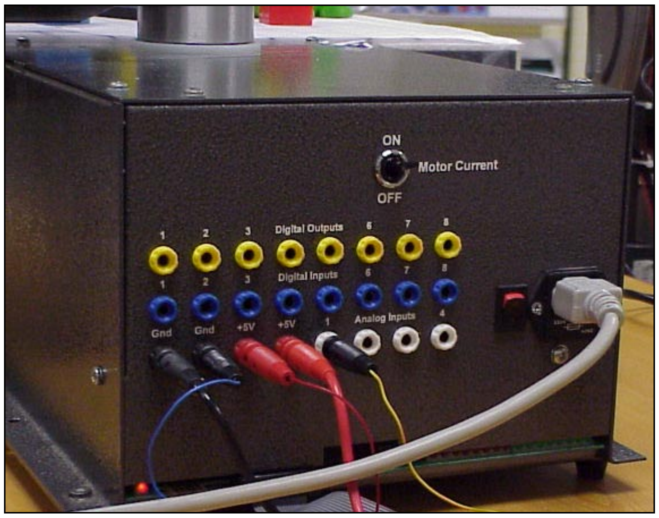


Figure II-2. Robot controls: Possibility to connect multiple sensors.

As explained before, the Mentor robot offers some capabilities that make it very interesting for education and training over the Internet (low cost, robustness, etc). On the other hand, it has some limitations that make difficult the execution of some complicated tasks. Anyway, we consider the advantages justify accepting the drawbacks. The limitations are the following:

- *Robot calibration:* The DC servo motors that control the robot joints are programmed using step values instead of joints' degrees. Moreover, different Mentor robots generate different gripper positions in response to the same input, which means a calibration procedure is required for every robot in the laboratory. Thus, for every joint, a dependency table to translate steps to degrees and vice versa is needed. Note this procedure is accomplished by hand and can introduce some little measurement errors that can affect the overall robot performance.

- *Robot reach:* As the robot mobility is very limited, there are positions over board that can not be reached by the robot. Depending on the application, this characteristic is very important. In our case, as the objective is letting students control a real robot to perform task operations, this limitation can be justified.

- *Robot repeatability:* As explained at Appendix I, the robot repeatability is 2 mm. In our opinion, there are situations where the repeatability is greater depending on the actual robot. Obviously, this situation means that some small errors occur when performing a grasping task that can make it fail. A possible solution consists of using small objects, with a width of less than forty millimeters.

- *Gripper design:* The Mentor gripper (see Figure II-4) is basically designed to manipulate simple objects like cylinders, cubes, etc. Grasping more realistic objects like Allen keys or a scissors is much more difficult.

1.2. Cameras

In order to perform a robot interaction based on vision input, it has been necessary to install 3 cameras, one from the top of the scene (*camera 1*), another from the front (*camera 2*), and finally the third one from the side (*camera 3*). *Camera 1* is calibrated and used as input to the

3D-model construction process and automatic object recognition procedure. Almost every scene-based movement is accomplished through the *camera 1*. Moreover, *camera 2* and *camera 3* are used as monitoring inputs, in order for the operator to supervise if the high level commands (e.g. "*Grasp allen*") are properly executed. Besides this, there are some situations where it is difficult for the robot to carry out these commands, like, for example, when several objects are overlapped. For these situations, it is necessary for the user to control the robot in a more "*teleoperated*" manner, instead of supervised. Thus, the operator will use every camera input to control directly the robot kinematics and then resolve these complicated situations by hand.

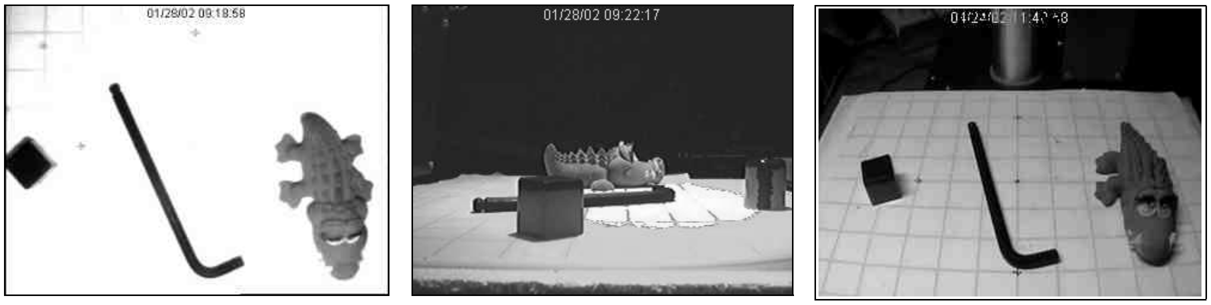


Figure II-3. Cameras input

At Figure II-3 an example of information from the cameras for a given scene is shown. Just note when teleoperating the robot (moving joints directly) the *camera 1* input will be almost hidden by the robot itself. As will be explained later, several techniques like virtual and augmented reality can be utilized to avoid this effect.

1.3. Distance sensor

As explained before, in order to determine the real object heights and constructing the 3D model, several experiments with distance (or proximity) sensors have been accomplished. At this point, we are going to explain the results from this experiments.

1.3.1. Sensor description

Basically, there are four types of sensors that can be used for this situation: IR (InfraRed), Acoustic, Capacitive, and Inductive.

Infrared proximity sensors

Infrared proximity sensors work by sending out a beam of IR light, and then computing the distance to any nearby objects from characteristics of the returned (reflected) signal. There is a number of ways to do this, each with its own advantages and disadvantages:

- *Reflected IR strength:* A simple IR proximity sensor could be built out of essentially just an IR LED and IR photodiode. This simple sensor, though, would be prey to background light (i.e., your IR "receiver" would be responding to naturally present IR as well as reflected IR).

- *Modulated IR signal:* A better solution would be to modulate the transmitted IR (e.g., to send out a rapidly-varying IR signal), and then have the reception circuitry only respond to the level of the received, matching, modulated IR signal (i.e., to ignore the

DC component of the received signal, and only trigger off the AC component). This method, though, is still at the mercy of the characteristics (in particular, IR reflectance) of the obstacle we are trying to sense.

- *Triangulation:* The best way to use IR to sense an obstacle is to sense the angle at which the reflected IR is returned to your sensor. By use of a bit of trigonometry, you can then compute distance, knowing the location of your transmission and reception elements.

There are some IR proximity sensors with this logic built in, like for example the Sharp GP2D15 IR Ranger (see Figure II-4). It has a built-in detection range of 24 cm, is reasonably priced, and is available from Acroname[1].

The GP2D15 interface is 3-wire with power, ground and the output voltage (the sensor outputs Vcc when it sees something at 24 cm distance); it requires 4.5 - 5.5 V power for operation, and consumes about 50 mA of current as long as it is powered. So, its advantages are (1) its simple interface, and (2) easy, reliable sensing of obstacles at a distance. Its disadvantages are (1) its requirement for 5 V power, and (2) its requirement for 50 mA of current regardless of whether anything is being sensed.

Acoustic proximity sensors

One very common method of avoiding hazards (at least for larger robots) is via sonar ranging. Here, acoustic signals are sent out, with the time of echo return being a measure of distance to an obstacle. This does, unfortunately, require fairly accurate timing circuitry, so acoustic sensors really require a processor of some sort to drive them. Also note that acoustic sensing essentially requires the use of commercial sensors.

The most common acoustic proximity sensor is the kind used in polaroid cameras, the "Polaroid Sonar Ranging Primer" (see [Polaroid Sonar][2] for more details). There are also some other possibilities, like the Devantech SRF04 UltraSonic Ranger (see [Dvantech Sonar][3].

Capacitive proximity sensors

Another possibility is detecting distance to objects by sensing changes in capacitance. When power is applied to the sensor, an electrostatic field is generated and reacts to changes in capacitance caused by the presence of a target. The main disadvantage of this sensor (often called a capaciflector) is that its usefulness is dependent on properties of the obstacles it is sensing (namely, their dialectric constant). The higher the dielectric constant (say, water), the more sensitive a capacitive sensor is to that target. The sensing of distance depends on the dielectric constant of the target and the surface areas of the probe and the target.

[1] http://www.acroname.com/robotics/parts/R49-IR15.html

[2] http://www.acroname.com/robotics/info/articles/sonar/sonar.html

[3] http://www.acroname.com/robotics/parts/R93-SRF04.html

Inductive proximity sensors

Another method for sensing distance to objects is through the use of induced magnetic fields. The primary problem with this method is that it is largely confined to sensing metallic objects.

1.3.2. Sensor Sharp GP2D15 experiments

As can be seen in Figure II-4, the GP2D15 ranger has been used by installing it over the gripper and connecting it through the Mentor analog input interface (see Figure II-2).

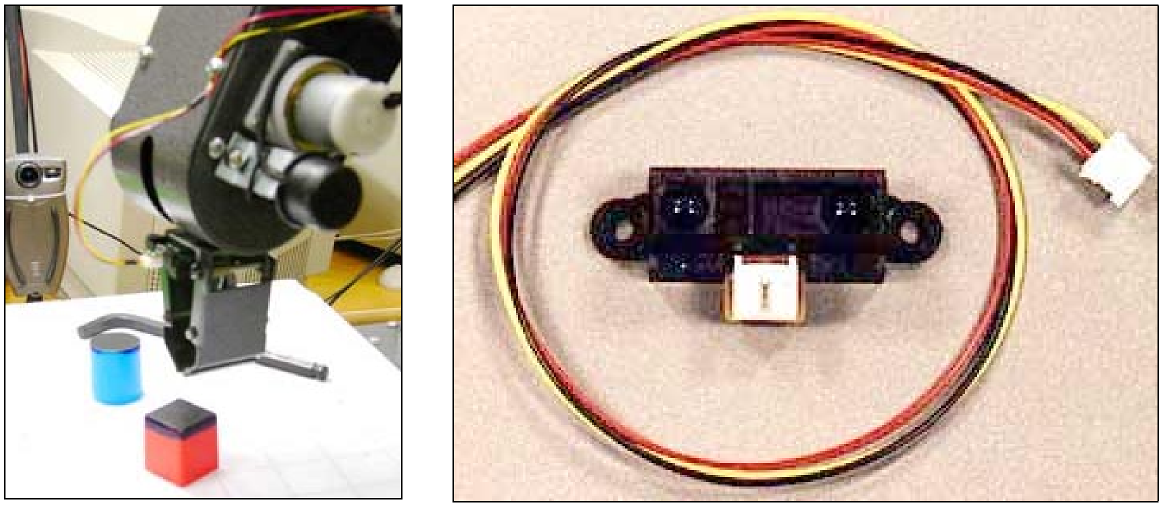
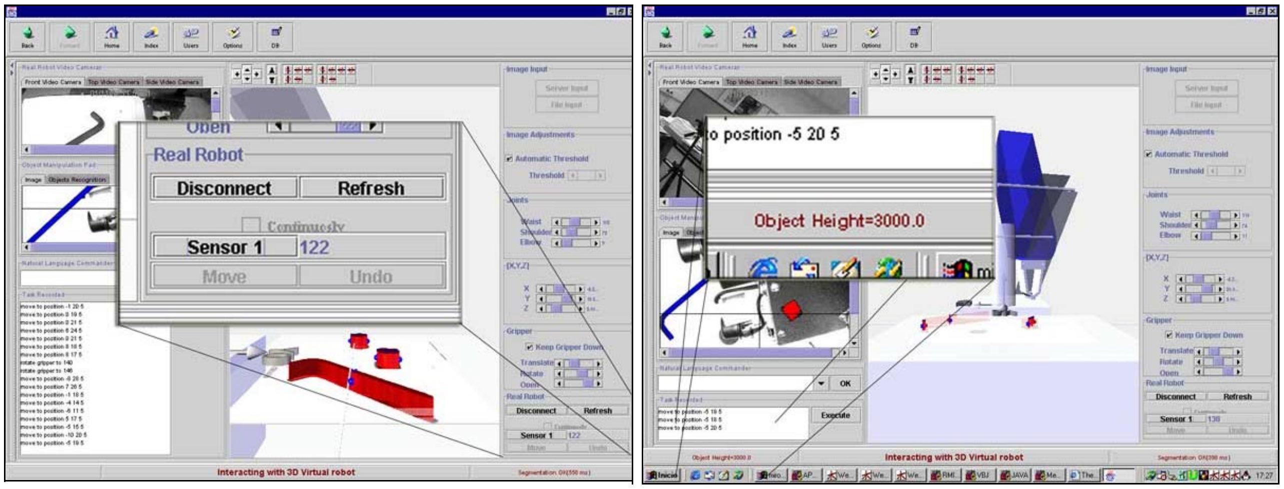


Figure II-4. Gripper and Sharp GP2D15 distance sensor



122

o position -5 20 5

Object Height=3000.0

Figure II-5. Reading the objects heights by using the Sharp GP2D15 distance sensor

After calibrating the sensor for the current illumination and the objects characteristics, the user interface (see Figure II-5) is able to read the real sensor value (e.g. 122 in Figure II-5) and then obtaining its representation in millimeters (e.g. 3,000 in Figure II-5). Several experiments using this configuration have been performed, varying the illumination parameters and the objects in the scene.

Experiment I

The first experiment consists of calibrating the distance sensor Sharp GP2D15 in order to be used on objects of the same material (black color surface with constant luminosity). The results show the relationship between the sensor value and the object's height. Note some variations were present depending on the actual illumination intensity and the light incidence angle over the object.

Sensor Value	Object Height (cm)
61-64	2
64-66	3
65-69	4
67-70	5
69-73	6
72-74	7
77-83	8
81-84	9
84-87	10
92-96	11

Figure II-6. Sensor calibration table for Experiment I: Objects with common surface material

As can be appreciated at Figure II-7, the relationship between sensor value and object height is somehow lineal.

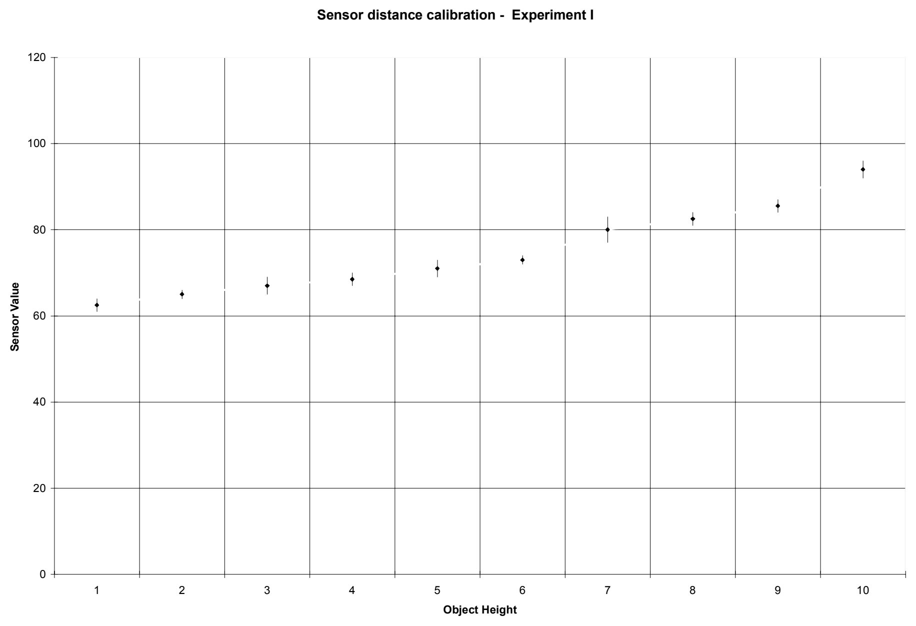


Figure II-7. Sensor calibration graphic for Experiment I

The conclusions from experiment I are the following:

- The Sharp sensor is a simple and inexpensive way to implement object's height calculation when the illumination is constant and the objects are formed of the same material.

- The variable angle of light incidence over the objects will result in some errors too. As can be appreciated at Figure II-7, these error rates are quite acceptable (less than a 2%).

Experiment II

The second experiment consists of relaxing the constant object material constraint in order to use the distance sensor in a more general manner. In fact, we have selected three different objects with different shades of black on their upper surfaces: black cube, lego wheel, and metallic allen key. Moreover, the illumination conditions have been modified too in order to make it constant through the whole objects' scene.

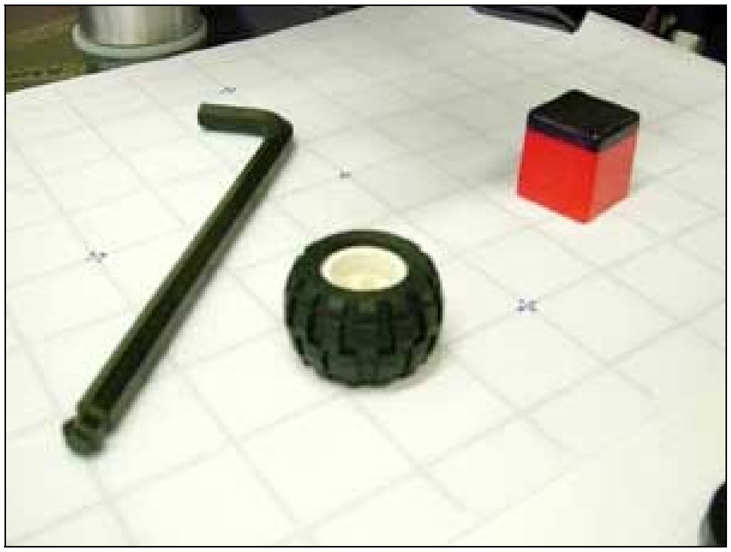


Figure II-8. Three real world objects with different variations of the same black color

The cube has a height of 3 cm, and its top surface is regular. On the other hand, the wheel presents an irregular surface and its height is 3 cm too. Finally, the allen key has an irregular metallic surface, with a height of 1 cm.

Object	Sensor Value	Object Height (cm)
Scene surface (white paper)	121-123	-
Cube	134-139	3
Wheel	126-130	3
Allen key	116-121	1

Figure II-9. Sensor calibration table for Experiment II

As can be appreciated at Figure II-9, the sensor values for the cube and wheel are very different, while their height is similar (3 cm). More over, for the allen key (metallic surface), the sensor result is below the level of the board.

In summary, the sharp sensor performs well when illumination is constant and objects presents the same surface (color and material). Under other conditions some other way to extract the height of the objects must be used.

The final conclusions from the experiments are the following:

- *Illumination dependent:* Depending on the actual illumination intensity on the room, the results obtained from the sensor varied over ±2 centimeters, which for our purpose is just to much.
- *Position dependent:* As the illumination intensity presents little variations over the scene, depending on the actual object position the results varied over ±1 centimeters.
- *Object material dependent:* The sensor responds perfectly when the object surface is equivalent to the white photographic paper, having a 90% of reflectance. Thus, in a real environment where different kinds of objects can appear over the scene, the situation is different. Thus, we could calibrate the sensor to respond perfectly when the objects had the same top surface (e.g. black color on top of the cylinder and cube in Figure II-4). On the other hand, when calculating the Allen key height, as the material was metallic, the response was unacceptable.

Thus, in order to obtain a more accurate object height we are considering the possibility to use more sophisticated sensors like the Polaroid Sonar explained before. We are actually studying these various possibilities.

At the moment, the system uses a different strategy in order to perform a proper 3D model construction from a camera input (it is a temporary solution). It consists of using the object recognition module in order to know the actual object height. Thus, when an Allen key is recognized in the scene, as we know in advance that this object has a height of 1.4 centimeters, the model is constructed accordingly. The only hazard situation is having to provide manually the approximate height when an unknown object appears in the scene (incremental learning). It means the height measurement is accomplished by the operator using the user interface. As soon as the right sensor is detected, the procedure will be updated accordingly.

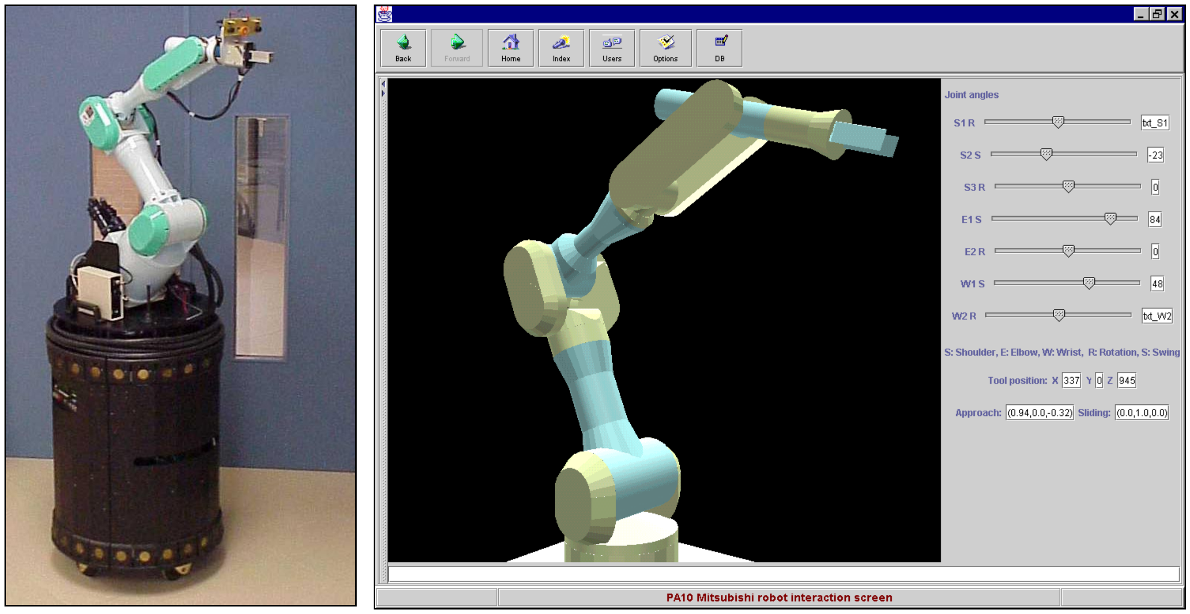
1.4. Mitsubishi PA-10

Apart from the Mentor educational robot, the UJI Telerobotic Training System is being prepared to control industrial robots too, like the Mitsubishi PA-10 and the AdeptOne. At the moment, users can interact with them over the Internet on a simulated manner. Online connections are still being implemented and restricted for research purposes.

As seen in Figure II-10, the Mitsubishi PA 10 robot [PA10][1] is a powerful robotic arm with 7-axis redundancy control that can be controlled by a PC or a built-in control unit. Important features that make the PA-10 useful are its innovative open system and its flexibility with human arm like maneuverability.

The PA-10 is lightweight at 35 kg, and powerful enough to carry a 10 kg load. Using a 7-axis redundancy control obstacle avoidance can be better implemented. Work in confined spaces can also be skillfully handled. It can even reach behind work-objects, greatly reducing the need for repositioning of the robot or the work-object. The absolute joint angle detection eliminates the need for any zero-point setting.

[1] (http://www.mitsubishitoday.com/1/PA10.htm)



Joint angles

S1 R txt_S1

S2 S -23

S3 R 0

E1 S 84

E2 R 0

W1 S 48

W2 R txt_W2

S: Shoulder, E: Elbow, W: Wrist, R: Rotation, S: Swing

Tool position: X 337 Y 0 Z 945

Approach: (0.94,0.0,-0.32) Sliding: (0.0,1.0,0.0)

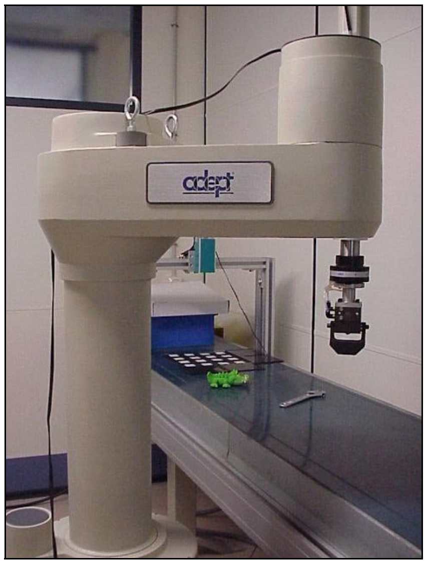
Figure II-10. Mitsubishi PA-10 mounted over a Nomadic mobile robot and using a stereo pair camera input (Its current 3D model is presented)

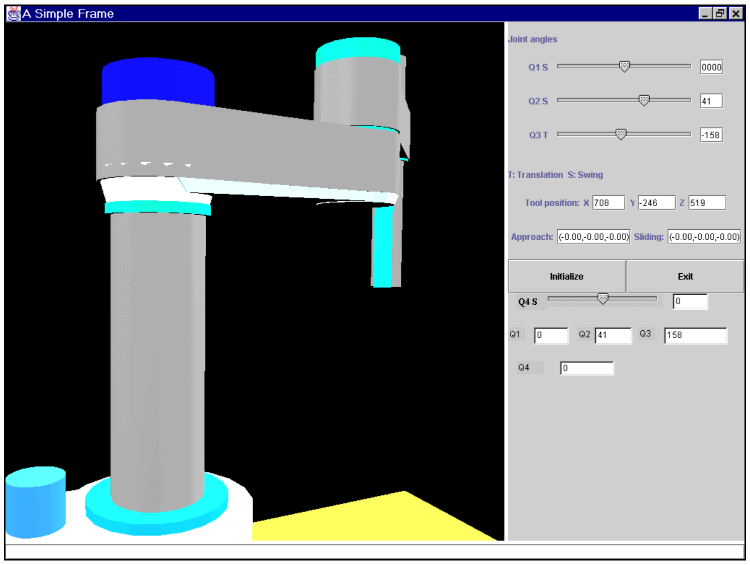
The stereo camera pair mounted over the gripper and the Nomadic robot facilitate the implementation of more interesting applications in service robotics. The long-term objective would be to carry out high level tasks like for example "Bring the scissors from Pedro's table", using the WWW as the programming medium.

1.5. Adept One

The AdeptOne industrial robot is designed for medium sized payloads. This SCARA robot has been optimized for high-speed performance while incorporating maintainability and serviceability features. The AdeptOne robot is represented in more than 5,000 application installations worldwide. It achieves 0,025 mm repeatability combined with a 12 Kg. payload capability and high Joint 4 inertia performance. In addition, the AdeptOne Quill length of either 203 mm or 356 mm allows it to handle large grippers for increased production cycle throughput with ease [Adept][1].

[1] (http://www.adept.com/main/products/robots/index.html)





Joint angles
Q1 S 0000
Q2 S 41
Q3 T -158
T: Translation S: Swing
Tool position: X 708 Y -246 Z 519
Approach: (-0.00,-0.00,-0.00) Sliding: (-0.00,-0.00,-0.00)
Q4 S 0
Q1 0 Q2 41 Q3 158
Q4 0

Figure II-11. AdeptOne installed on the robotics laboratory. Objects are moving over a programmable conveyor belt, while watched by a static camera from the top (Its current 3D model is presented).

At Figure II-11 the AdeptOne work space can be seen. A camera is monitoring the objects coming along the conveyor belt. After being recognized and processed in order to extract its corresponding grasping points, the robot is able to execute the corresponding task. At the moment, the robot control is accomplished through an already implemented CORBA server than can be accessible from the Internet in a restricted manner. The actual work is focused on the conveyor belt programming and the selection of proper gripper configurations. The long-term objective is allowing students to access the real robot from the laboratory (using the WWW) in a controlled way.

2. Main project capabilities

In this section, the main telerobotic system capabilities are described in a very high level manner. As we can appreciate at Figure II-12 once connected to the robot, the manipulator goes to the initial position, so the whole scene information is accessible from the top camera.

Then, the camera image is captured and transferred to the *computer vision* module, which identifies every isolated object in the scene, as well as calculates a set of mathematical features that uniquely distinguish every object on the board (moments, contours, area, etc). See Chapter IV for more information.

The third step consists of using the contour information and the moments calculated previously, in order to determine every stable grasping point associated with each object. This process is named *Grasp Determination*, and its application is fundamental to allow high level task specification (e.g. "grasp object 1"). For more details, please refer to Chapter V. Another important output of the grasp determination procedure is the *k*-torsion vector, which defines the external curvature representation of every object. This information can be very useful for carrying out the following procedure, which is the *automatic object recognition* process.

By using the computer vision output (moments, area, etc.) and the grasp determination information (curvature, grasping points, etc.), the *object recognition* module calculates a set of invariant descriptors associated with every object (thinness ratio, elongatedness, Hu moments, etc.). Then, it compares this information with the already learnt representation of known classes stored in a database. By applying the object recognition algorithms described in Chapter IV, an object name (e.g. allen key, screw, etc.) is associated with every object in the scene. So that, interaction with the robot can be accomplished by using commands like "Grasp allen", instead of "Grasp object 1".

At this point, we shall use the output from the grasp determination and the computer vision module in order to *construct a 3D model* of the robot scenario. 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. As explained in the section below, the 2D camera input is not sufficient to construct the 3D model. In order to calculate the object's heights the object recognition output are used. Moreover, the distance sensor can help to calculate the object's heights for those situations where the objects have the same reflectance properties (material).

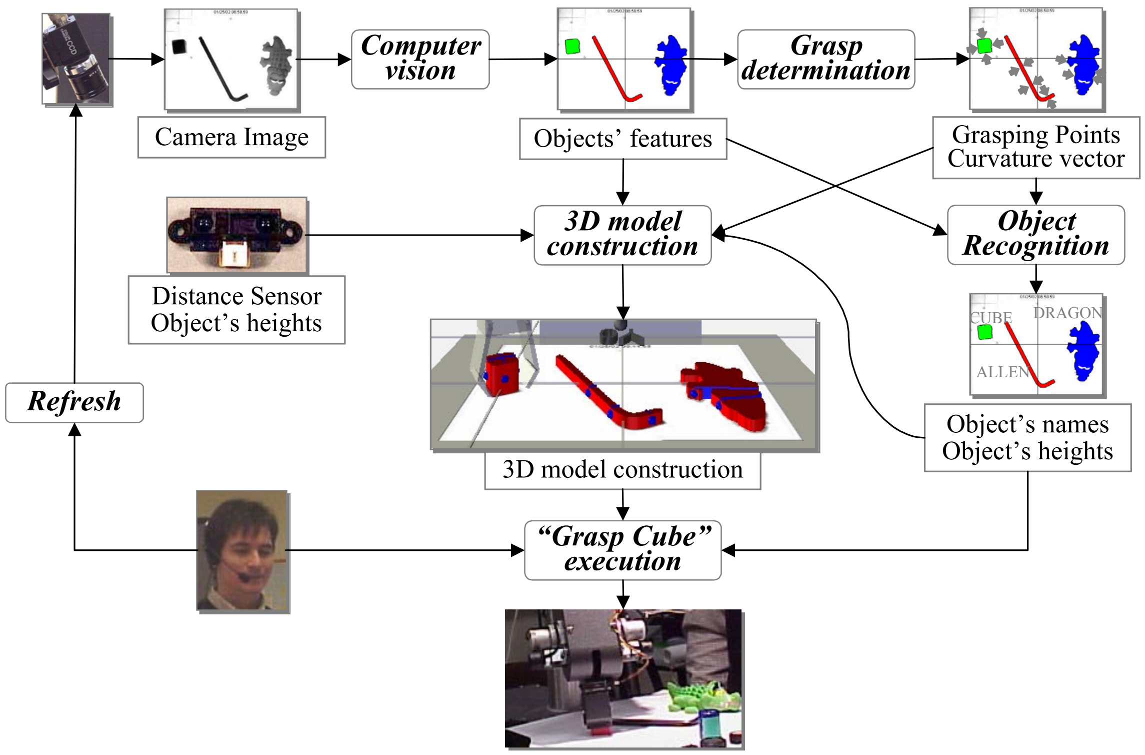


Figure II-12. Relationship between the main project modules.

Once the 3D model is constructed, user is able to program the robot in a predictive manner. Thus, for example, the operator can speak directly to the microphone and command the action "Grasp Cube". The task is first executed on the virtual 3D model, and then, once confirmed by the user, performed by the real robot on the remote scenario.

3. Multiple Ways of Interaction

One of the main important features of the UJI Telerobotic System is the possibility to interact at different levels depending on the actual state of the robot (busy or available) and the current task to be performed (supervised or teleoperated).

First of all, the user interface enables two interaction modes:

- *Simulated (Off-line):* For those situations where the robot is being used by someone else or has been deactivated temporally by the administrator in order to accomplish maintenance operations, the user can still interact with the robot in a simulated manner. Thus, the only requirement is having to provide the user interface with a top camera image (it can be the real working environment) and then the 3D model is constructed with the different objects. The important point is that user can specify a whole assembling task by using this initial state (camera image), which can be applied later on in a single step over the real robot (task specification).

- *Real Robot (On-line):* On the other hand, when the robot is free to be used, the operator can interact with it in a very similar way to the simulated one. The only difference is that once a robot operation has been confirmed, apart from storing it into the task specification field, the operation is launched on both, the real and the simulated robot. Moreover, as introduced in Chapter I, in order to sort out the time delay problem over the Internet, the *on-line* interaction is accomplished by default using a predictive display. It means the user can specify, for example, a grasping task over the simulated interface and then deciding whether applying this operation or not to the real robot. Just note this predictive configuration can be deactivated and then work directly with the real robot in a *Move&Wait* strategy [Sherindan, 1992][1].

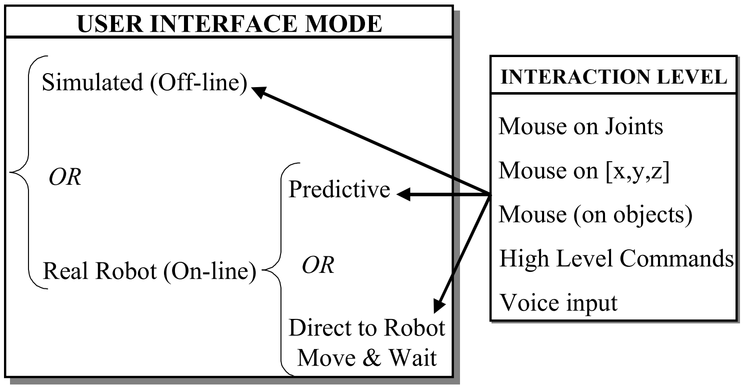


Figure II-13. User interface modes (simulated, real, etc.) and their possibilities of interaction with the robot (Mouse, Voice, etc).

As presented in Figure II-13, for every user interface mode (*off-line*, *on-line and predictive*, *on-line and direct*), we can control the robot at different interaction levels, depending whether the task is to be *supervised* (intelligence at the robot) or *teleoperated* (intelligence at the operator):

[1] [Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

- *Mouse on Joints:* The user can directly perform operations to control the robot joints, gripper translation-rotation, and gripper opening. It defines the most *teleoperated* interaction level (lowest level), due to the fact that operator must provide the whole intelligence to perform the robot command (see Figure II-14).

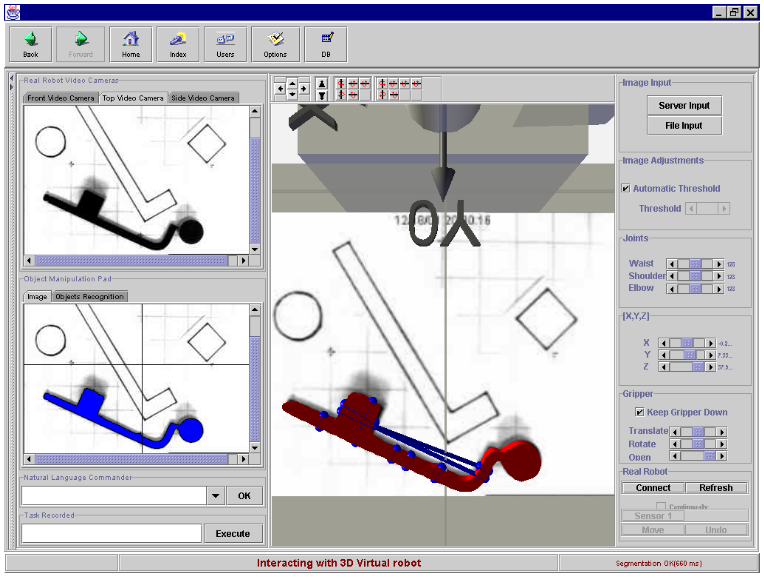


Figure II-14. User interface in simulated mode (off-line) to resolve an assembling task where objects are overlapped. At this point, the user is the one who provides the intelligence (teleoperated). The user can select a given grasping alternative and separate the objects before assembling them.

- *Mouse [x,y,z]:* The second approach is to provide the operator with the robot inverse kinematics already sorted out. It means the user can move the robot in world coordinates instead of the joints. It is very useful, for example, to elevate an object over the *Z*-axis maintaining the gripper orthogonal to the board. In this situation the robot intelligence increases and the operator can focus a little bit more on the task instead of the robot itself.

- *Mouse on objects:* For those situations where the operator detects the computer vision algorithms have performed properly (because of proper illumination and no objects overlapping), a higher level of interaction using the mouse can be accomplished (Figure II-15). Thus, for example, the user can click directly over an object in the scene and select the option "*Grasp*", which means the robot is responsible for selecting the best stable grasping point over the selected object. Besides this, if more grasping possibilities can be performed, the user is able to suggest a particular grasping in order to perform the task. As can be appreciated, at this point the operator interacts in a higher level of programming, which is "*supervised*" instead of "*teleoperated*". The robot takes care of the details, while user focuses on the task itself.

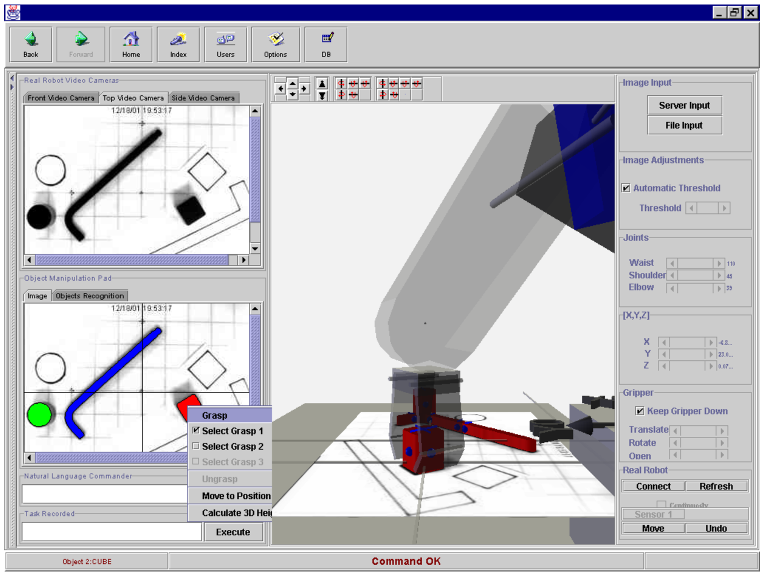
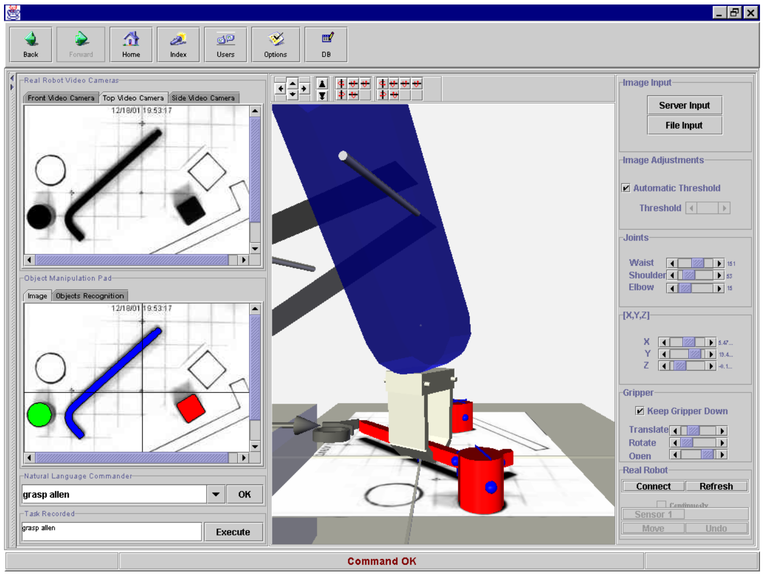


Figure II-15. User interface in simulated mode (off-line) and predictive option. Operator selects picking up the cube using grasping 1 (most stable). The robot has the intelligence (supervised interaction).

- *High level commands:* Apart from mouse operations over the scene objects, it is possible to send commands to the robot as text input, like for example "*Grasp the object 1*", or "*Pick up the allen*". These commands consists of a simplification of the natural language and have access to the whole robot functionality (joints operations, object manipulation, grasping selection, etc.). The main advantage of this approach is the possibility of executing groups of sequential commands that configure a whole task (e.g. assembling two objects). Besides this, as explained in the next chapter, the user interface enables external applications (e.g. a speech recognition program) to connect to the user interface via a TCP/IP socket and then to send text commands in order to control the robot functionality. This kind of architecture is very common in telecommunications and is sometimes known as *northbound interface*.



grasp allen

grasp allen

Command OK

Figure II-16. User interface in simulated mode (off-line) and predictive option. Operator has written (or spoken by the microphone) the command "grasp allen". After that, operator has confirmed the movement, so command is added to the "Task recorded" field and robot picks up the allen key.

- *Voice input:* The highest level of robot programming is accomplished by the execution of high level commands as voice input instead of written text over the user interface. This possibility has many advantages due to the fact that the user can control the robot with a single microphone. For many people exercising remote control over a robot by voice commands at such a distance is impressive. The user is connected over the Internet, which means the user can be hundreds of miles away from the robot. For more details, refer to Chapter VI.

In summary, the user interface offers great flexibility to the user, in such a way that allows the selection of the interaction level required at a particular moment. For those situations where the robot intelligence is insufficient to accomplish the task (e.g. objects overlapping), the operator can interact at a lower level by selecting directly grasping options or even moving the robot in world and joints coordinates (*teleoperation*). On the other hand, when normal situations are presented (e.g. well isolated objects), the robot will be able to respond properly to high level commands like "grasp allen" (*supervised telerobotics*).

4. 3D Models: Virtual & Augmented Reality

At this point, we are going to present some terminology such as immersive virtual reality, non immersive virtual reality, and augmented reality, and the way they can be used in order to improve the human-robot interaction on a web-based telerobotic scenario.

4.1. Immersive Virtual Reality

Today, 'Virtual Reality' is used in a variety of ways and often in a confusing and misleading manner. Originally, the term referred to '*Immersive Virtual Reality*'. In immersive VR, the user becomes fully immersed in an artificial, three-dimensional world that is completely generated by a computer. The head-mounted display (HMD) was the first device providing its wearer with an immersive experience. Evans and Sutherland demonstrated a head-mounted stereo display already in 1965. It took more than 20 years before the first commercially available HMD came up, the famous "EyePhone" system (1989).

A typical HMD houses two miniature display screens and an optical system that channels the images from the screens to the eyes, thereby, presenting a stereo view of a virtual world. A motion tracker continuously measures the position and orientation of the user's head and allows the image generating computer to adjust the scene representation to the current view. As a result, the viewer can look around and walk through the surrounding virtual environment.

To overcome the often uncomfortable intrusiveness of a head-mounted display, alternative concepts (e.g., BOOM and CAVE) for immersive viewing of virtual environments were developed.

The BOOM (Binocular Omni-Orientation Monitor) (see [Fakespace][1]) is a head-coupled stereoscopic display device. Screens and optical system are housed in a box that is attached to a multi-link arm. The user looks into the box through two holes, sees the virtual world, and can guide the box to any position within the operational volume of the device. Head tracking is accomplished via sensors in the links of the arm that holds the box.

The CAVE (Cave Automatic Virtual Environment) (see [Cave][2]) was developed at the University of Illinois at Chicago and provides the illusion of immersion by projecting stereo images on the walls and floor of a room-sized cube. Several persons wearing lightweight stereo glasses can enter and walk freely inside the CAVE. A head tracking system continuously adjust the stereo projection to the current position of the leading viewer.

Apart from these systems, a variety of input devices like data gloves, joysticks, and hand-held wands allow the user to navigate through a virtual environment and to interact with virtual objects. Directional sound, tactile and force feedback devices, voice recognition and other technologies are being employed to enrich the immersive experience and to create more "sensualized" interfaces.

The unique characteristics of immersive virtual reality can be summarized as follows:

- Head-referenced viewing provides a natural interface for the navigation in three-dimensional space and allows for look-around, walk-around, and fly-through capabilities in virtual environments.
- Stereoscopic viewing enhances the perception of depth and the sense of space.
- The virtual world is presented in full scale and relates properly to the human size.
- Realistic interactions with virtual objects via data glove and similar devices allow for manipulation, operation, and control of virtual worlds.

[1] [Fakespace] (http://www.fakespace.com)

[2] [Cave] (http://www.evl.uic.edu/EVL/VR/systems.shtml)

- The convincing illusion of being fully immersed in an artificial world can be enhanced by auditory, haptic, and other non-visual technologies.
- Networked applications allow for shared virtual environments (see below).

4.2. Non-immersive VR

Today, the term 'Virtual Reality' is also used for applications that are not fully immersive. The boundaries are becoming blurred, but all variations of VR will be important in the future. This includes mouse-controlled navigation through a three-dimensional environment on a graphics monitor, stereo viewing from the monitor via stereo glasses, stereo projection systems, and others. Apple's QuickTime VR (see [QuickTime][1]), for example, uses photographs for the modeling of three-dimensional worlds and provides pseudo look-around and walk-trough capabilities on a graphics monitor.

4.3. WWW Non-immersive VR

For our situation, as the UJI telerobotic training system runs over the WWW and the main objective is letting as many people as possible have access to the real robot, the non-immersive VR must be accomplished using a simple mouse interaction.

So now the question is, *which technology can be used on a WWW environment in order to allow the design of a virtual reality scenario?*

The term *Web3d* describes any programming or descriptive language that can be used to deliver interactive 3D objects and worlds across the Internet. This includes open languages such as *Virtual Reality Modeling Language* **(**VRML**)**, *Java3D* and *X3D* (under development) - also any proprietary languages that have been developed for the same purpose come under the umbrella of *Web3d***.** The *Web3D Repository* is an impartial, comprehensive, community resource for the dissemination of information relating to Web3D and is maintained by the Web3D Consortium.

VRML (Virtual Reality Modeling Language) (see [VRML][2]) provides three-dimensional worlds with integrated hyperlinks on the Web. Home pages become home spaces. The viewing of VRML models via a VRML plug-in for Web browsers is usually done on a graphics monitor under mouse-control and, therefore, not fully immersive. However, the syntax and data structure of VRML provide an excellent tool for the modeling of three-dimensional worlds that are functional and interactive and that can, ultimately, be transferred into fully immersive viewing systems. The current version VRML 2.0 has become an international ISO/IEC standard under the name VRML97.

Apart from VRML, there exists the possibility of integrating a virtual reality world on a Java applet by using Java3D, which means the whole Java language can be used as support to the final application. An API (Application Programming Interface) that enables Virtual Reality applications must support generating 3D graphics, handling input trackers, and continuously

[1] [QuickTime] (http://www.apple.com/quicktime/qtvr/index.html)

[2] [VRML] (http://www.web3d.org/vrml/vrml.htm)

integrating that tracker information into the rendering loop. The Java 3D API includes specific features for automatically incorporating head tracker inputs into the image generation process and for accessing other tracker information to control other features. It does this through a new view model that separates the user's environment from the application code. The API also explicitly defines the values returned by six-degrees-of-freedom detectors as primitive Java 3D objects, called Sensors. Together, the new view and input models make it quite easy to turn interactive 3D graphics applications into VR-enabled applications. A detailed description of the API appears in the Java3D specification. Rather than trying to support all possible 6 degrees of freedom input devices, Java 3D defines an InputDevice interface that tracker vendors or developers can use to support a particular 6 degrees of freedom device. The InputDevice interface requires the implementer of a device driver to define nine methods. The nine methods provide device-specific semantics for open, close, and read operations as well as a minimal number of state-setting and state-query methods.

In summary, the UJI Telerobotic Training System uses the Java3D API because it enables the implementation of non-immersive VR that can be converted in a near feature in a fully immersive one by the inclusion of several input devices.

4.4. Augmented Reality

Virtual reality has been a subject of great interest, and increasing attention is being paid to the related field of *Augmented Reality*, due to its similar potential. The difference between Virtual Reality and Augmented Reality is in their treatment of the real world. Virtual Reality immerses a user inside a virtual world that completely replaces the real world outside. In contrast, Augmented Reality lets the user see the real world around him and augment the user's view of the real world by overlaying or composing three-dimensional virtual objects with their real world counterparts. Ideally, it would seem to the user that the virtual and real objects coexisted.

In order to understand the Augmented Reality the key issue is the registration problem. The registration of the object virtual information is overlaid on the information coming directly from the cameras. In typical augmented reality systems developed, head-trackers are used for tracking user's head position/orientation, rangefiner or sonar sensor is used for detecting or tracking the object pose in the world. The problems are lack of accuracy and latency of the system. Most commercially available head- trackers do not provide sufficient accuracy and range. The rangefiner and sonar sensor are not sufficient enough for its speed and accuracy.

For the UJI Telerobotic point of view, the augmented reality capability has been used to improve the way the user interacts with the robot. For example, for those situations where the real robot is below the camera (objects' occlusion), it is necessary to help the user with virtually generated information in order to allow him to continue programming the task. Other features like the measurements of the objects and more can be easily included into the system by using the same technique.

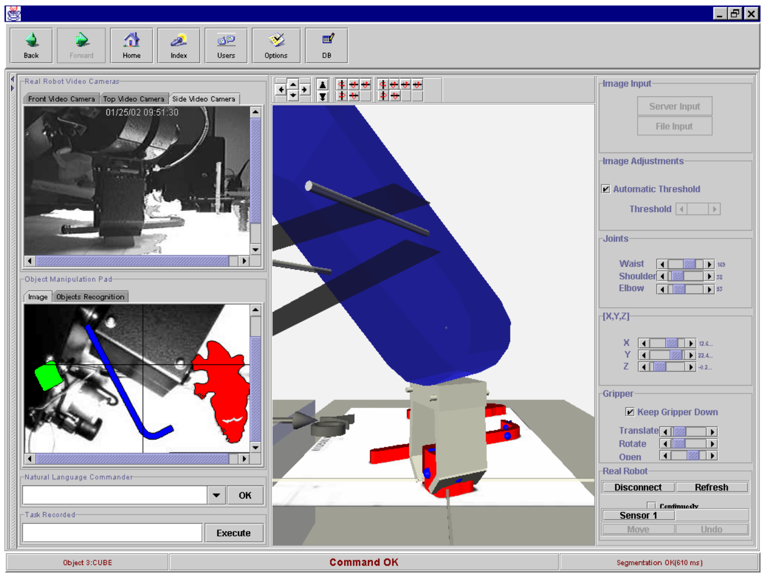


Figure II-17. Augmented reality used as a way to manipulate partially occluded objects. It is combined with a non-immersive virtual reality 3D model implemented with Java3D.

As can be seen in Figure II-17, the user interface presents a non-immersive VR 3D model implemented with Java3D, that enables the user to vary his viewpoint by means of mouse interactions. Besides this, in order to facilitate the robot programming when camera input is partially occluded by the robot, an augmented reality feature is added to the "Object Manipulation Pad" panel.

5. On-line programming

In this section, the virtual and augmented reality capability is presented by means of an example of grasping an object over the board and dropping it into another place (Pick & Place operation). It means the explanation is focused on the "*Active Guided Programming*" that can be accomplished by means of the Java *module "Java 3D virtual & Augmented Reality*" introduced above. Please note we present the situation when the operator has control over the real robot.

As can be seen in Figure II-18, the user interface has 4 main parts. The first one the cameras, that give the user a continuos monitoring to the board scenario from two different viewpoints. Note a Manipulation Pad is included where user can click directly over the objects and send commands to the robot through a mouse interaction. The second the 3D virtual scenario, that monitors the real robot position and renders it over the screen. Note this capability allows user to see the work area from any viewpoint, avoiding occlusions and facilitating the robot programming procedure. The third part refers to the robot movement controls, that allow the user to move the robot to a specific (x,y,z) location or even access directly the degrees associated to every join. The fourth part is the text-input field that allows the user to specify the robot tasks in a more natural manner (e.g. Pick up the cube).

In Figure II-18, the real and simulated robot are situated at the initial position, the scene objects are already recognized (allen, cube and dragon), and the user interface is able to execute a grasping operation. Look how the user has clicked over the cube object in order to select the "Grasp" option.

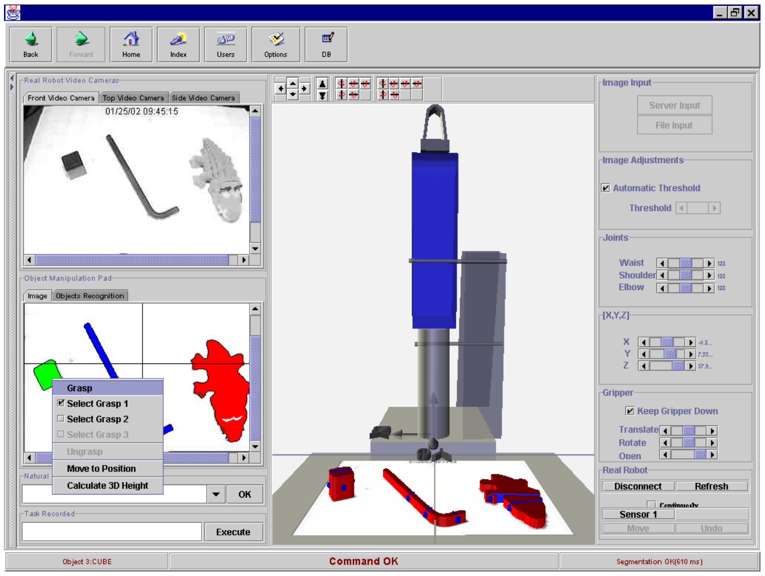


Figure II-18. User interface initial state once connected to the real robot. See how objects are already recognized (cube, allen, etc.) and grasping points over the 3D virtual environment are represented. Note how the user has clicked over the cube in order to select the grasp option.

As can be seen in Figure II-19, in order to manipulate an object, first we have to place the gripper above it. To do so we can move the arm by using the controls at the right side of the screen (*teleoperation*), by clicking with the mouse the appropriate object on the manipulation pad, or by entering the command "Grasp cube" (*supervised control*).

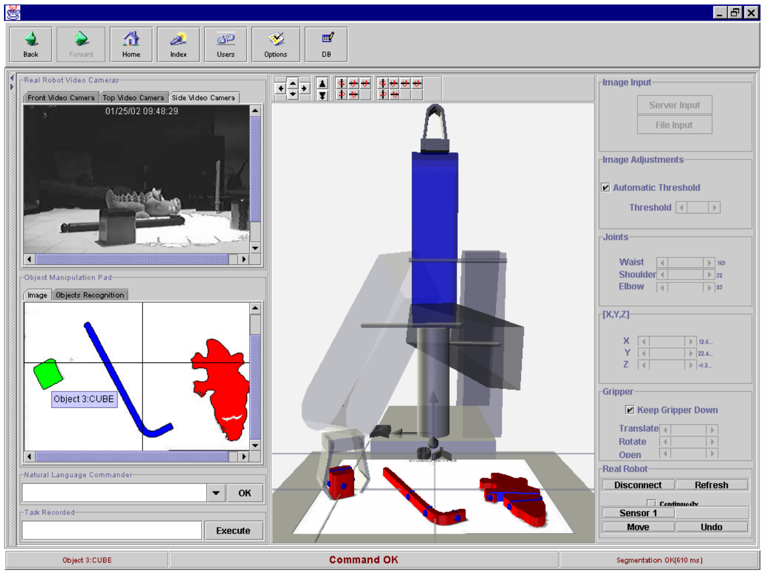


Figure II-19. User selected the "Grasp cube" option and the 3D virtual environment "predicts" the operation before sending it to the real robot. The transparent robot representation means the predicted position and the opaque the real robot situation.

As can be appreciated in Figure II-19, the 3D virtual environment is predicting the "Grasping cube" task. As the operation is not performed over the real robot (see the real image from the cameras), the opaque representation of the robot monitors its current position. At this point, we could confirm the operation on the real robot by selecting the "Move" button, or undo the predicted position by pressing the "Undo" option.

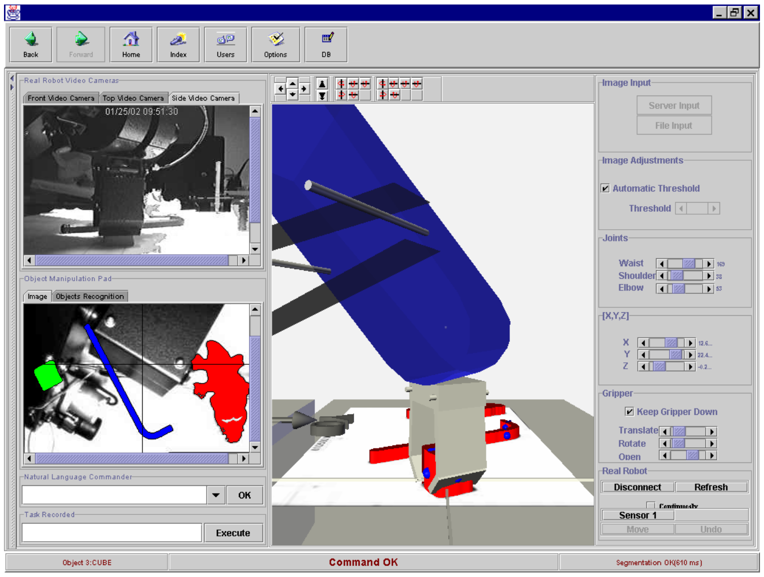
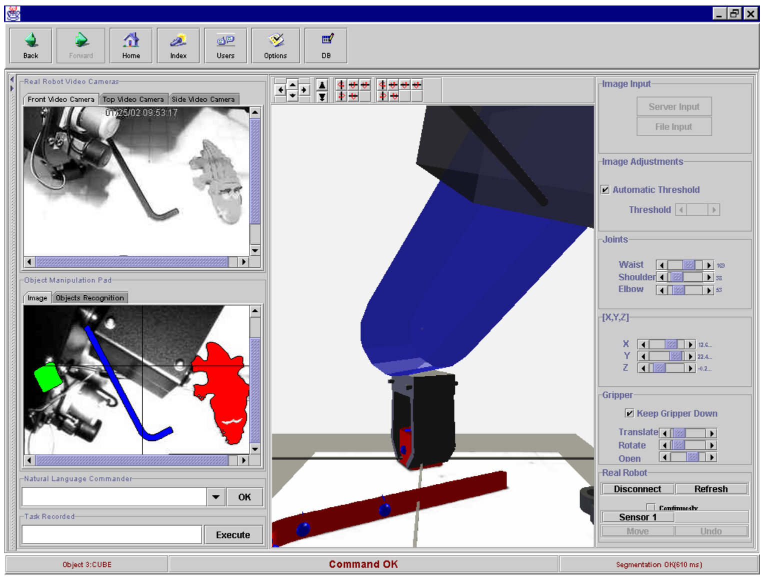


Figure II-20. User confirmed the predicted operation "Grasp cube" by selecting the Move button. Real robot performs accordingly. Augmented reality avoids robot occlusion over the manipulation pad and the 3D virtual environment.

At this point, the user has selected the "Move" button and the robot has performed the "Grasp cube" operation over the Internet to the real robot. As can be appreciated in Figure II-20, the real top camera input is mixed with the object recognition knowledge, providing an augmented reality information. By looking at the camera images, the real position of the gripper over the object can be seen, prepared to execute the next action.



Object 3:CUBE Command OK Segmentation OK(610 ms)

Figure II-21. The same robot position by using another viewpoint and the front camera monitoring.

Note how useful are the projections of the gripper over the board in order to get a better understanding of the position of the arm into the world (augmented reality). Besides this, note how interesting is the possibility of moving the user viewpoint in order to better specify a given task (virtual reality). See in Figure II-20 and Figure II-21 two different user viewpoints of the same robot position. This feature allows the user to navigate through the robot world and interact with it by using any point of view. The navigation buttons through the 3D virtual environment are situated on top of the 3D representation. There are four preprogrammed positions (front, left side, right side, top) by using two possibilities, closed view and normal.

The "Grasp cube" task has been accomplished in a simple user interaction. It means the user did not take care of the grasping points to be used or the approximation procedure. The user occupies a supervision role instead of teleoperating every robot movement, and then waits to confirm whether it was successful or not. Thus, for example, user could have selected the second grasping possibility over the cube, which means the robot would have performed accordingly.

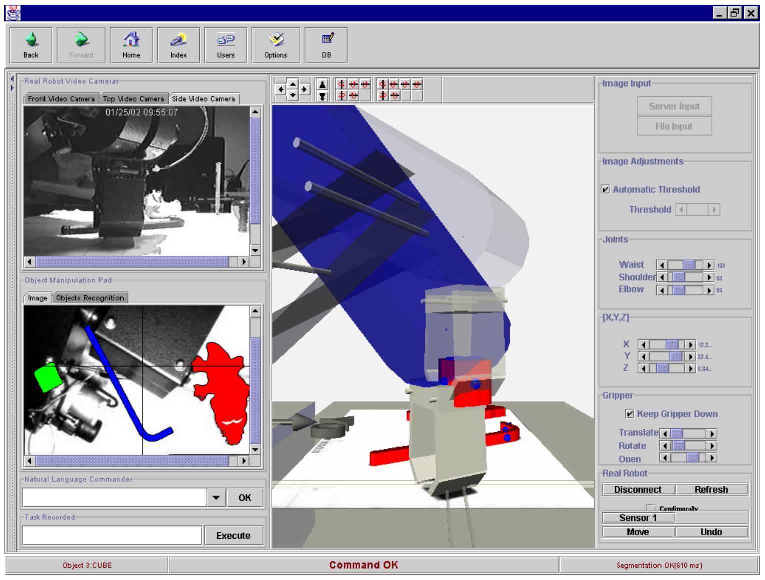


Figure II-22. Operator elevates the cube and the user interfaces predicts the movement before sending it to the real robot. The prediction is performed on both, robot position and objects.

As seen in Figure II-22, next step consists of elevating the cube by moving the gripper in world coordinates over the *Z*-axis. See how the user interface predicts the movement by showing an almost transparent robot, which represents the predicted position. Note that prediction is accomplished on both, virtual robot movement and constructed 3D object being manipulated.

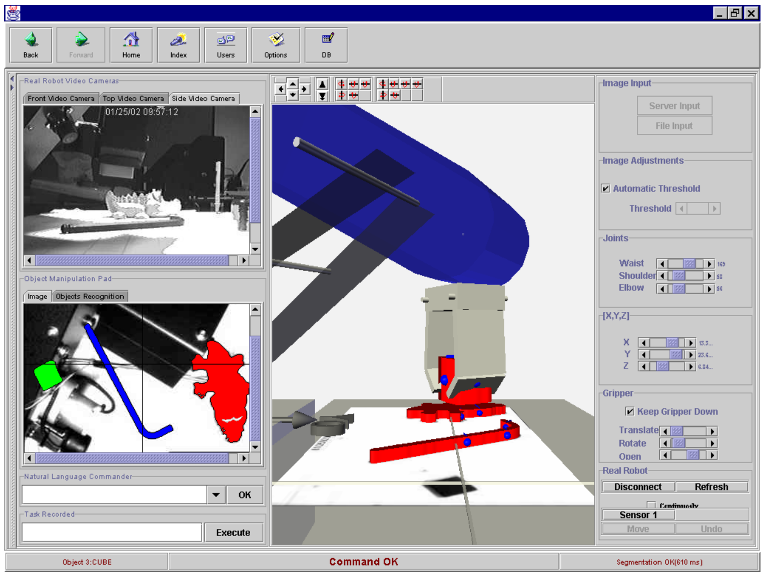


Figure II-23. Elevating the cube using the real robot

At Figure II-23 it can be seen how user has confirmed the predicted position from Figure II-22, and then the real robot has performed the action accordingly. Next step consists of dropping the object at a given position into the board, so first this would be situating the gripper over the dropping point, which is performed in Figure II-24 and Figure II-25.

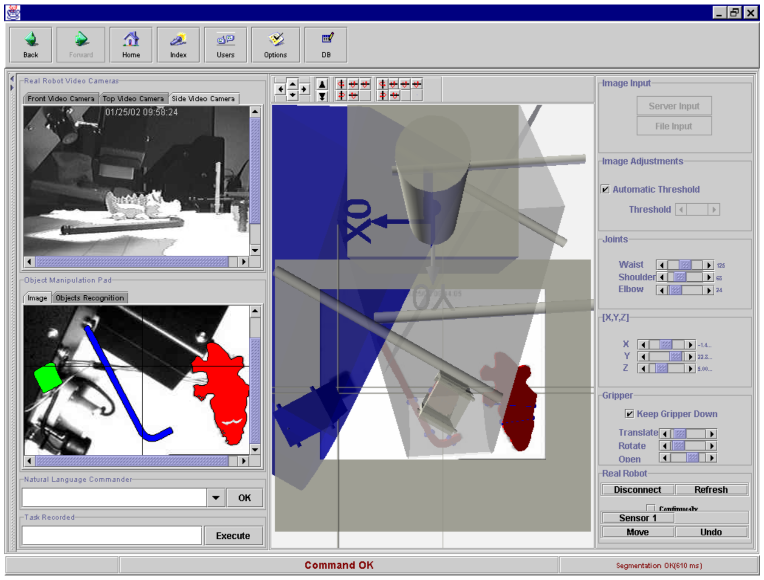


Figure II-24. Predicting next dropping position by using the top view option.

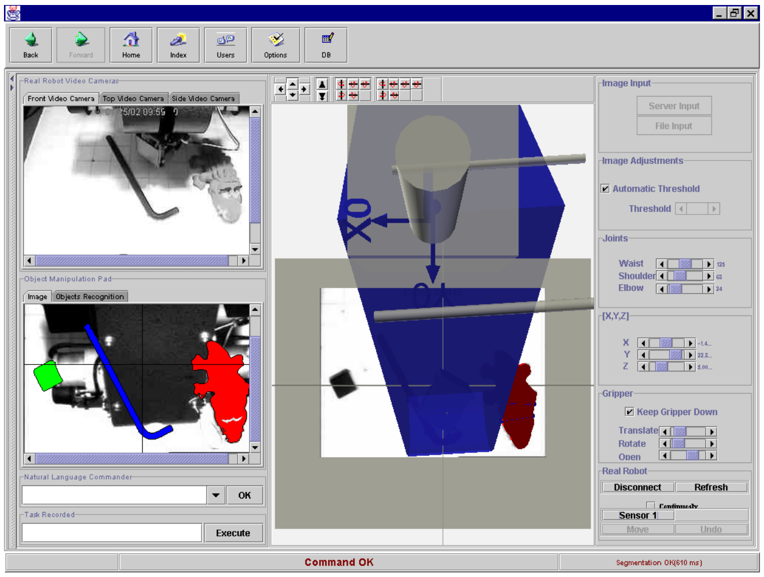


Figure II-25. Moving the real robot to the last predicted position in order to perform an un-grasp operation.

Now the real robot is located several centimeters over the board, having the cube grasped with the gripper. The next step consists of moving down the gripper over the *Z*-axis before ungrasping the cube.

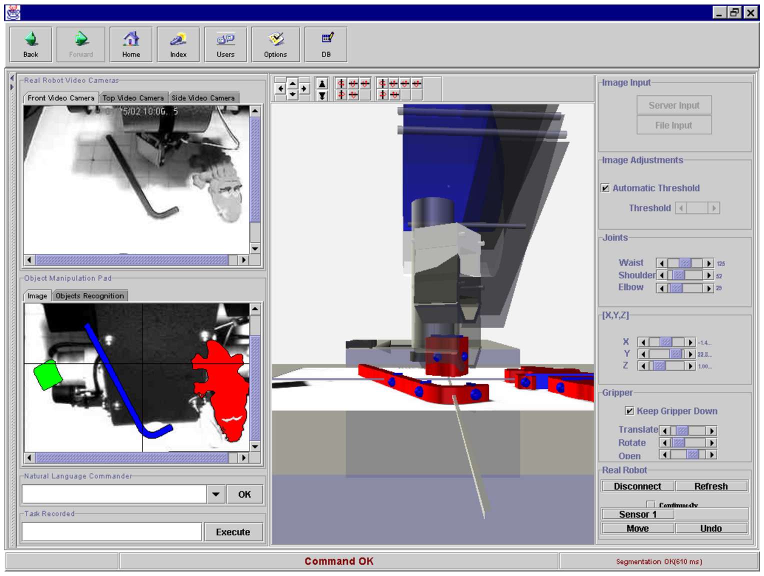


Figure II-26. Programming next position by moving down the gripper in order to perform a proper ungrasping operation.

In Figure II-26 can be seen the prediction operation to move down the gripper in order to ungras the object. Then, once the predicted position has been confirmed by the user the next robot position is shown at Figure II-27.

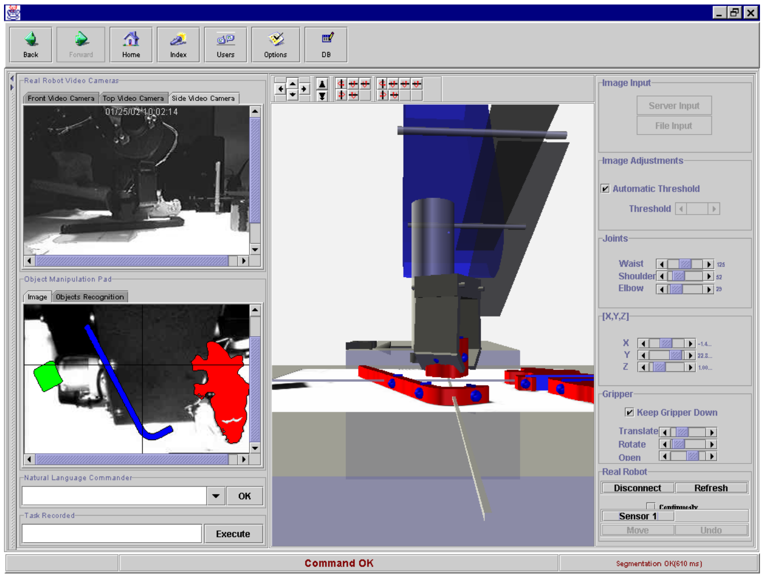
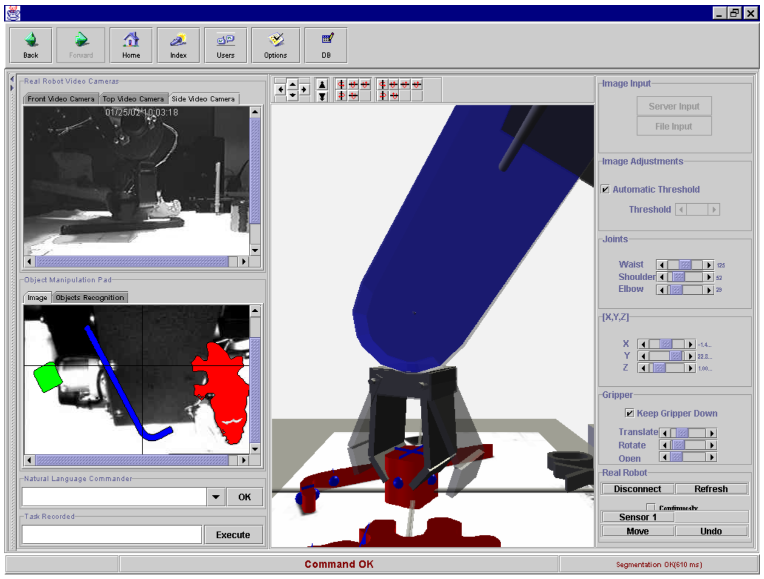


Figure II-27. Confirming the gripper movement over the real robot

Again, in order to drop the object we can open the gripper the let the simulated object drop onto the board.

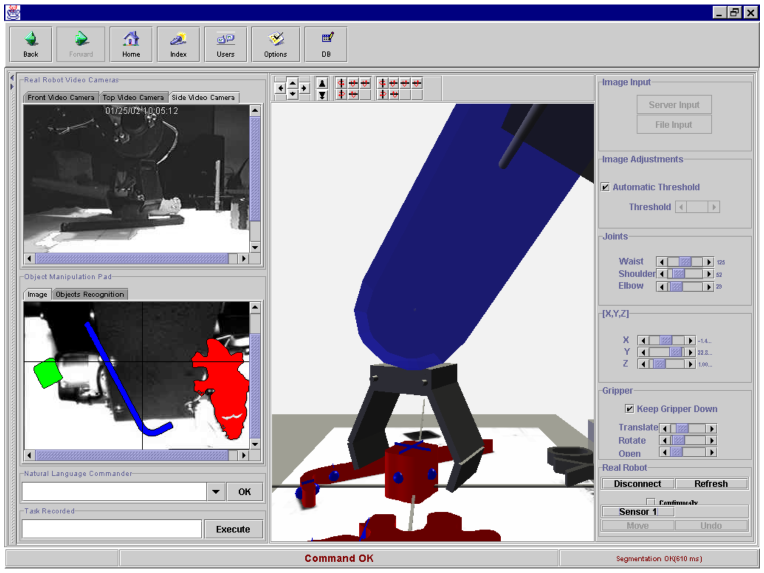


Command OK

Segmentation OK(610 ms)

Figure II-28. Opening the gripper

In Figure II-29 the robot has finally dropped the cube onto the board and the user can confirm the operation as successful by using the camera input. Not how the manipulating pad is not updated accordingly yet, due to the fact that the 3D model state is considered predicted until the robot comes back to the initial position and reconstructs again the model from the real top camera input. It means the prediction feature has a limit in time of an object movement on the table. After that, system must be refreshed, so robot goes to the initial position, new object's position recalculated and the 3D construction reevaluated.



Command OK

Segmentation OK(610 ms)

Figure II-29. Gripper opening confirmed and simulated 3D object position is predicted on the board. The real images from the cameras show the same situation.

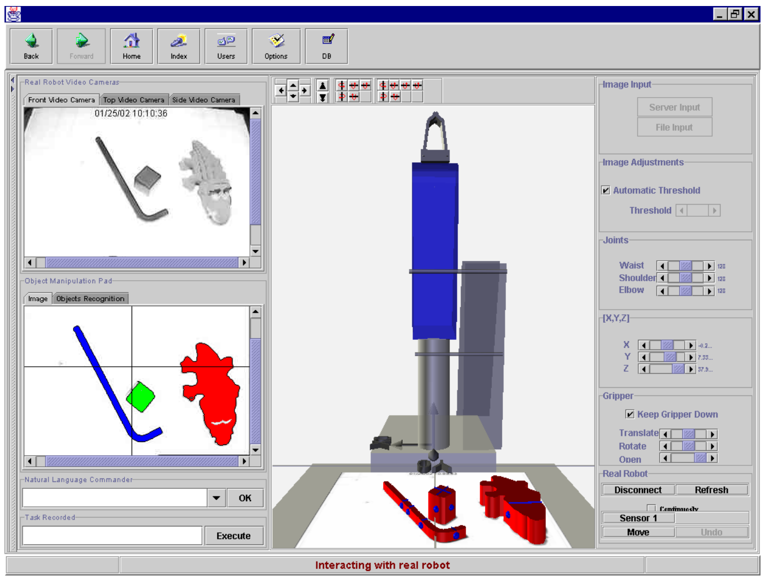


Figure II-30. Bringing the robot to the original position, recognizing the new scene configuration and updating the 3D model according to the real situation.

Once the robot comes back to the initial position by pressing the button "Refresh", the manipulation pad state and the 3D virtual environment is updated accordingly.

6.Off-line programming

For those situations where the robot is being accessed by someone else on the Internet, the system allows users to program the manipulation activities in an off-line manner, by using the 3D model representation of the robot scenario as a task simulation tool. Then, once the task is programmed, its specification is stored into a set of high level commands that can be kept and executed later on over the real robot. The idea is providing students and researchers with a tool that enables interaction with the robot depending whether the device is actually accessible or not. This is another project contribution to the web telerobotics domain, where the majority of available systems make users wait until the robot is free to be used (e.g., Telegarden and Australian Telerobot).

This off-line programming can be applied in many environments. For example, in an industrial domain this kind of technique is very convenient, due to the fact that while the robot is being programmed to perform a given task it can be accomplishing an important activity on the production line. The robot programming does not mean stopping the production activity. Moreover, in the education and training environment (like ours), letting fifty students wait until one of them frees the robot control is obviously not very convenient. In fact, our experience has demonstrated that students can be very motivated programming into a virtual environment if the user interface is designed using proper techniques (virtual and augmented reality, 3D model construction, task prediction, etc.).

6.1. *Pick & Place example*

At this point we are going to see an example of programming a simple task of picking an object and placing it onto a selected board position. The problem can be observed in Figure 32, where the circle has to be placed on its corresponding position, which is its mould (left side of the figure).

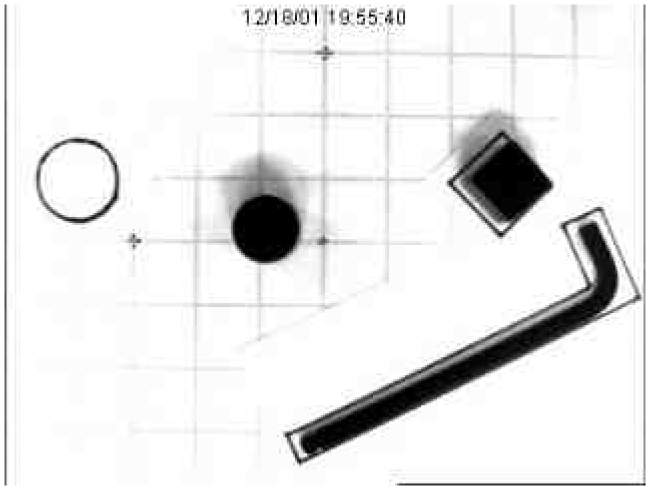


Figure II-31. Problem of placing the circle into its corresponding position.

The first step consists of specifying the image below as initial state in order to program the task in off-line mode. To do so select the file by pressing the button "File Input". The result can be observed at Figure II-32.

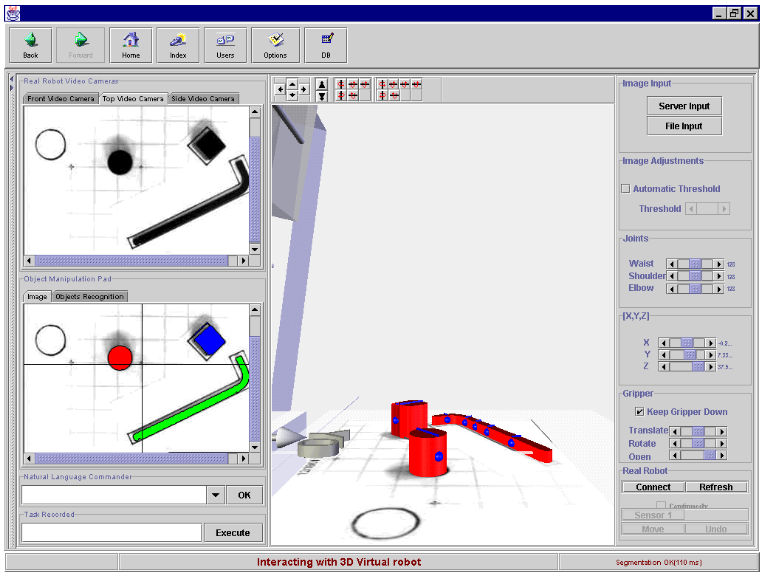
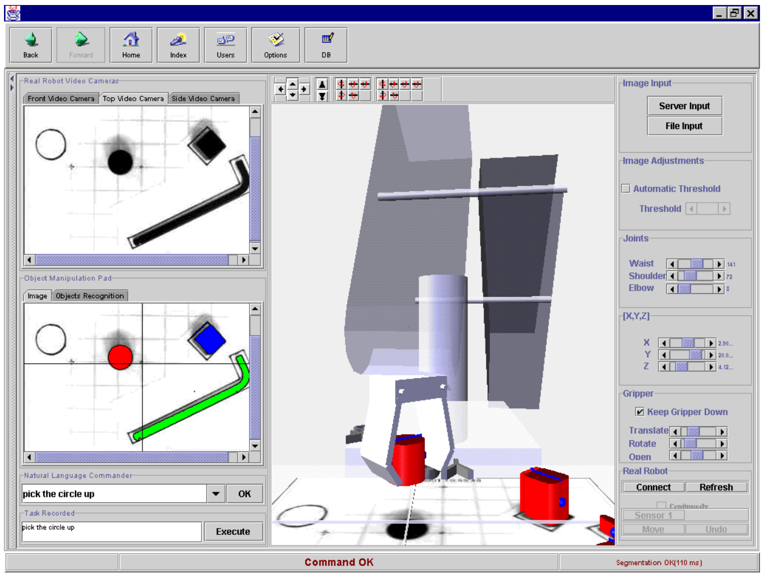


Figure II-32. Problem of placing the circle into its corresponding position.

The next step consists of using any of the robot commands (mouse, text, etc.) to begin the task execution. For example, a good alternative would be to pick up the circle by using the command "Pick the circle up". The result is observed at Figure II-33:



pick the circle up

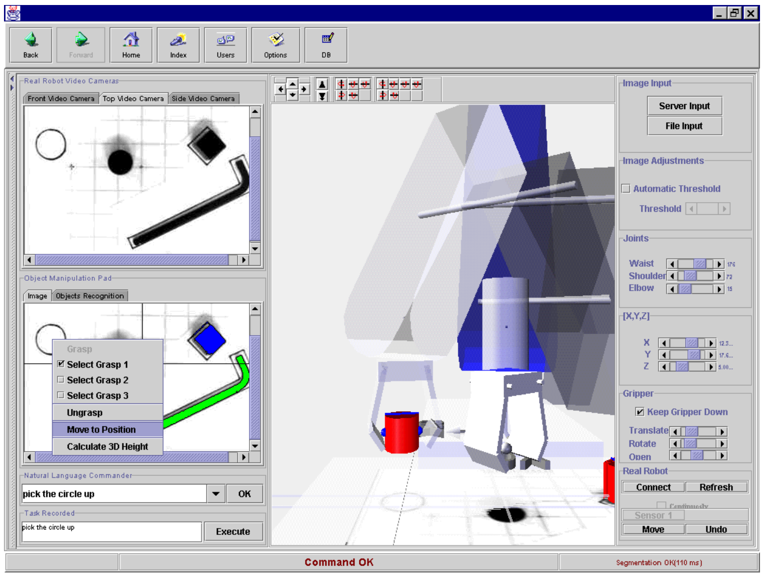
pick the circle up

Command OK

Segmentation OK(110 ms)

Figure II-33. Executing the command "Pick the circle up" and saving into the task specification field.

As the robot already knew the grasping points associated with the circle object, it just executed the only grasping alternative and elevated the object over the *Z*-axis. Look how the command has been stored in the Task specification field once confirmed by the user by pressing the "Move" button.



pick the circle up

pick the circle up

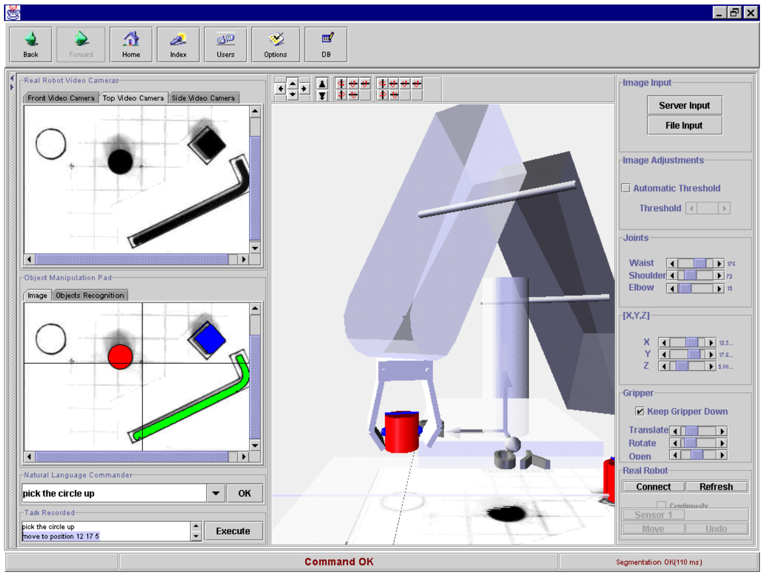
Command OK

Segmentation OK(110 ms)

Figure II-34. Specifying the placing point by selecting the "Move To Position" option

As observed in Figure II-34, the following action consists of selecting the exact point where the object is to be placed on the board. A possible alternative could be, for example, clicking the placing point into the manipulation pad, and then selecting the "Move to Position" option.

Then, if the position selected is confirmed by the user, the results are shown in Figure II-35. The opaque robot is moved to the corresponding position and the action is saved in the task specification file.



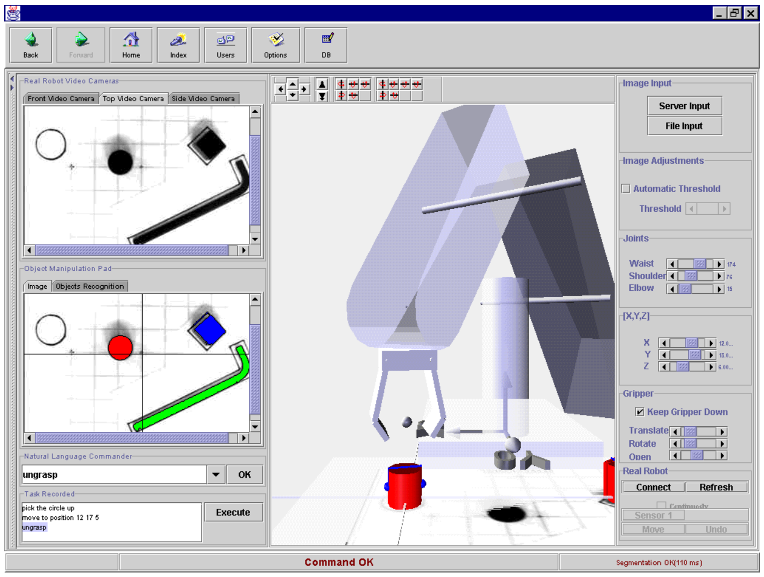
pick the circle up

pick the circle up
move to position 12 17 5

Command OK

Segmentation OK(110 ms)

Figure II-35. Confirming the moving operation and saving the action on the task field specification field.



ungrasp

pick the circle up
move to position 12 17 5
ungrasp

Command OK

Segmentation OK(110 ms)

Figure II-36. Selecting the ungrasp option.

And finally, as shown in Figure II-36, by selecting the "ungrasp" option on the manipulation pad the robot places the circle on the board. Look how the action has also been recorded in the task file.

Finally the task programmed is the following:

```
pick the circle up

move to position 12 17 5

ungrasp
```

Now, the task could be executed again and again over the off-line environment or even on the real robot once user takes control of it.

As explained later, in order to execute a programming task over the real robot, it should be taken into account whether or not the initial state is the same on both, the real and the simulated robot. Depending on the way the task has been defined, it will be dependent or independent of the initial object's position.

6.2. Available programming commands

As it has been explained in Figure II-13, the user interface permits the user to control the robot at different interaction levels: (1) Mouse on Joints, (2) Mouse on [x,y,z], (3) Mouse (on objects), (4) High Level Commands, and (5) voice input.

Once introduced into the system by using any interaction level, the user has the possibility to store it as a written high level command into the task specification field. The final result will be having a text file that identifies a robot task to be accomplished by a robot, by using both, on-line and off-line possibilities.

The present section summarizes the whole set of available text commands that can become part of a more general task like the one explained in the previous section. By combining several commands, different tasks can be programmed.

Command	Description
Grasp the {object name}	Given a scene with objects, it executes the most stable grasping alternative on the object {object name}
Grasp the object {number}	Given a scene with several objects, it executes the most stable grasping alternative on object {object number} numbered from top to down and from left to right.
Move ahead	It moves the TCP ahead 1 cm
Move down	It moves the TCP down 1 cm
Move left	It moves the TCP to the left 1 cm, taking the manipulation pad orientation as the reference
Move right	It moves the TCP to the right 1 cm, taking the manipulation pad orientation as the reference
Move to position {x} {y} {z}	It moves the TCP to the world coordinates position [x, y, z]
Move up	It moves the TCP up 1 cm
Move to the sector {number}	The scenario is divided into 16 sectors, which are numbered from 1 to 16. The command moves the robot to the {number} sector.
Open gripper to {number}	It opens the gripper a {number}% of its capacity
Pick the {object name} up	It executes the grasping operation over the object labelled as the {object name}, and then elevates it 5 cm over the board
Pick the object {number} up	It executes the most stable grasping on object {number} and then elevates it over the board 5 cm (by default)
Place it at position {x} {y}	Having an object grasped in the gripper, it places it at position {x, y} in image coordinates
Place it over the {object name}	Having an object grasped in the gripper, it places it on top of the {object name}
Place it over the object {number}	Having an object grasped in the gripper, it places it on top of object {number}
Refresh	It brings the robot to the initial position, captures the scene image, recognizes the objects, and constructs the 3D model.
Rotate gripper to {± number}	It rotates the gripper {± number}%
Ungrasp	Having an object grasped in the gripper, it places it at the current robot position.
Ungrasp at position {x} {y}	Having an object grasped in the gripper, it places it at position {x, y} in image coordinates
Ungrasp over the {object name}	Having an object grasped in the gripper, it places it on top of the {object name}
Ungrasp over the object {number}	Having an object grasped in the gripper, it places it on top of object {number}

Figure II-37. Available robot commands

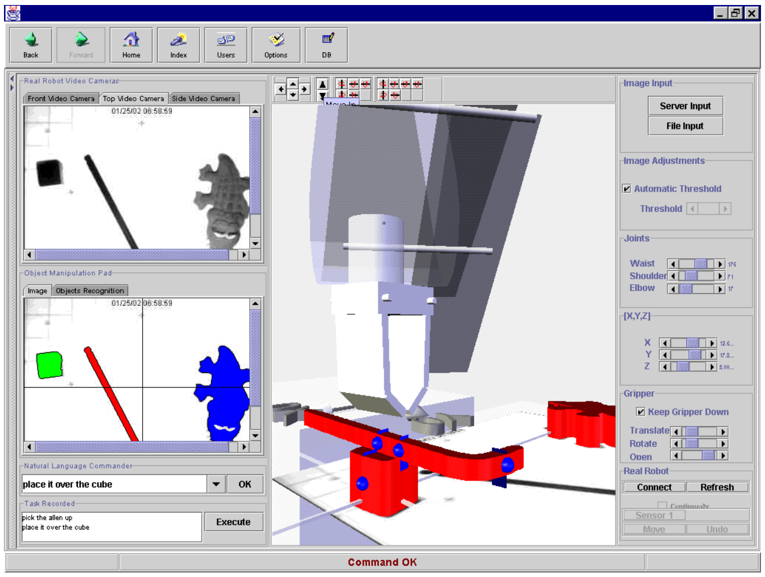
6.3. Position independent commands

In section 6.1 we have seen an example of using the available robot commands (see Figure II-37), in order to program a particular robot task (grasping the circle and dropping it into the corresponding place). Then, once the user gets control over the real robot, the question is the following: *How can we assure the initial conditions (object positions in the scene) are the same as the ones used to program the task in off-line mode? Would it be possible to define more general tasks that work properly independently of the initial conditions?*

The answer is yes, as long as the commands used for the task are position independent (e.g "Pick the allen", "Place it on the cube").

First of all, in order to identify the object to be manipulated, the user has the possibility to program tasks by using object names instead of object numbers. Obviously, if the object does not exist on the scene the task will generate an error and the robot will pause. The set of commands that can be used to accomplish such an action are: "Grasp the {object name}" and "Pick the {object name} up". The grasp action should be followed by a second "Move up" command (if needed).

At this point the object is grasped and the gripper elevated several centimeters over the board, ready to accomplish the *placing* action. Thus, second step consists of selecting the point over the board where object has to be placed. A good way to do this would be partitioning the board in quadrants and specifying the one to be used in the placing command (e.g "Place it on quadrant 12"). System could incorporate some collision prediction features in order to avoid placing the object on an already occupied portion of the board. In fact, there exist a lot of techniques for real-time collision avoidance that could be implemented (see [Gupta el al., 1998][1] and [Pobil et al., 1995][2]). Another alternative would be using commands like "Place it at the left side of the allen", which permit a very high level of interaction an let the robot deduce the exact portion of the board to be used in order to place the object. These two alternatives are being incorporated (at writing time) and open a very interesting research field related to the qualitative spatial reasoning. Some of our next research work will be focus at this domain.



01/25/02 06:58:59

Natural Language Commander: place it over the cube

Task Recorded:
pick the allen up
place it over the cube

Command OK

Figure II-38. Using the cube as target to place the allen

What we have prepared to partially resolve this problem is incorporating the command "Place it over the {object name}". This action is very interesting because permits user to specify a placing task where the target can vary depending of the situation. At Figure II-38 can be seen a possible application of this command.

[1] [Gupta et al., 1998] K. Gupta, A. P. del Pobil (ed.), "Practical Motion Planning in Robotics: Current Approaches and Future Directions", John Wiley & Sons, New York, 1998.

[2] [Pobil et al., 1995] A. P. del Pobil, M. A. Serna, "Spatial Representation and Motion Planning", Springer-Verlag, Berlin, 1995.

Figure II-38 shows the result of executing two sequential commands: "Pick the allen up", and "Place it over the cube”. These types of actions have applications in many fields, education and training etc., and more particularly in an industrial domain where objects should be classified accordingly at their respective places. Thus, a direct extension of this problem would be incorporating into the scene the border patterns where objects should be placed (see Figure II-39). Then, the robot can be trained to recognize these patterns, so for example, the one associated with the allen could be called "allenpattern", and the one related to the cube could be "cubepattern". After this, the robot could respond to actions like "Place the allen over the allenpattern", which means executing in a single step an invariant to position task that has application in many robotics areas.

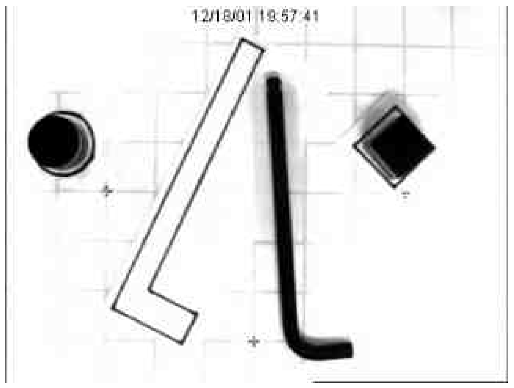


Figure II-39. Using objects patterns on the scene as task's objectives where objects has to be placed

Note the Imin and Imax axis (see chapter IV) for the pattern and its corresponding object are identical, so the robot is able to know the objective orientation as well as its real position in the scene.

7.Education & Training

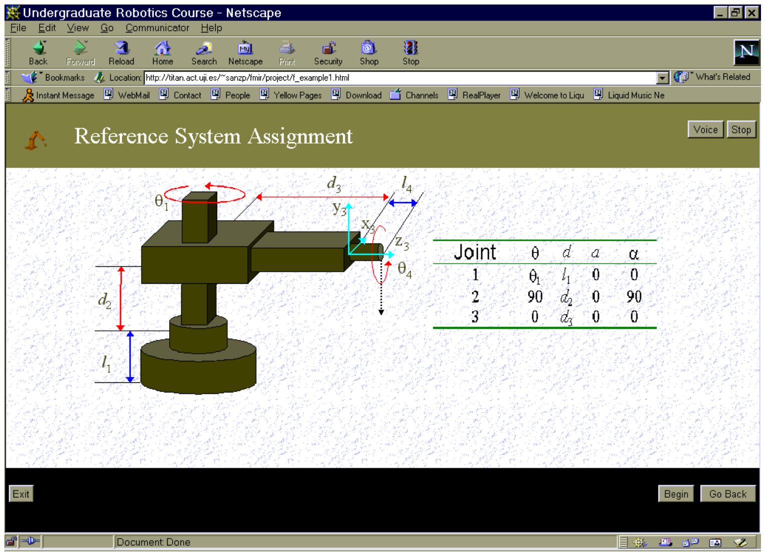
As explained in Chapter VII, the project is being used as an educational product for teaching-learning basic concepts in the robotics subject to every user connected to the web. In a first stage the system has been used in the classroom for undergraduate students within their computer science curricula in our university.

The idea is to allow the student to have a tool that helps them to learn the difficult robotics’ concepts and complements traditional theoretical lessons. One of the major problems in education is the different knowledge level that every student has. While some students may find the teacher’s explanation slow and boring others may find the same explanation fast and difficult. Thus, the multimedia training system takes into account those differences and acts accordingly. In this case, it is the student who imposes the speed and not the teacher. Students will establish their own learning process speed. Besides this, as the system has the ability to model in real time the user knowledge level, the lessons are presented accordingly to the kind of user that we are interacting with. Moreover, when using a computer oriented tutorial, students sometimes still need the advice of the teacher. Our objective is to supply a method that allows the student to get online-supervised by the teacher or another advanced student when necessary [Marín et al., 2001][1].

Bearing all this in mind, the Telerobotic System introduced above has been complemented with some alternative functionality that helps enormously the student in order to elaborate the lab exercises: (1) Robotics Tutorial, (2) Online Teaching Assistance, (3) Robotics Lab Complementary Exercises, and (4) Industrial Robots Simulation.

7.1. Robotics Tutorial

Since 1999 the Multimedia Research Lab at the Jaume I University has been implementing several tutorials in computer science, and particularly in the Robotics domain [Sanz et al., 1998][2][Marín et al., 1997][3]. The already existing Robotics Tutorial has been improved during this period in order to allow students to learn as many concepts as possible by using the Web.


Undergraduate Robotics Course - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://titan.act.uji.es/~sanzp/tmir/project/f_example1.html What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels RealPlayer Welcome to Liqu Liquid Music Ne
Reference System Assignment
Voice Stop
θ_1
d_3
l_4
y_3
x_3
z_3
θ_4
d_2
l_1
Exit
Begin Go Back
Document: Done

Joint	θ	d	a	α
1	θ_1	l_1	0	0
2	90	d_2	0	90
3	0	d_3	0	0

Figure II-40. Robotics Tutorial to access the "Reference System Assignment" theoretical concepts

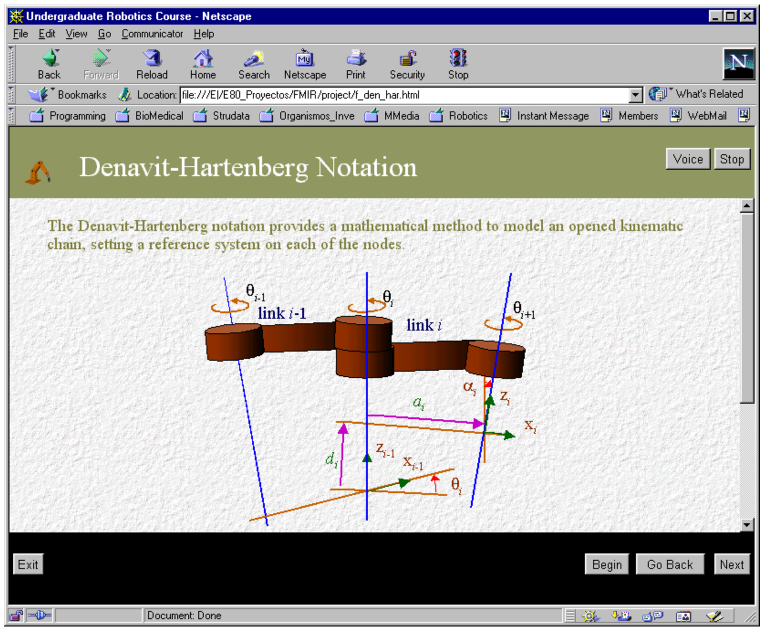
The last version of this HTML tutorial on Robotics has been incorporated as part of the UJI Telerobotic Training System, allowing the students to access online the theoretical concepts necessary to accomplish the practical sessions (see Figure II-40 and Figure II-41). Note the Robotics tutorial is not just a static HTML document, since it allows some kind of interaction through JavaScript code that offers the student a certain degree of interactivity.

[1] [Marín et al., 2001] R. Marín, P.J. Sanz. *The UJI Telerobotic Training System. 1st EURON Workshop on Robotics Education and Training (RET2001),* Alicia Casals, Antoni Grau, Grup Artyplan-Artymprès, S.A., Weingarthen, Germany, 2001.

[2] [Sanz et al., 1998] Sanz P.J., Adell S. An undergraduate Robotics Course via Web. Teleteaching'98 Conference, a part of the 15th IFIP World Computer Congress. Distance Learning, Training, and Education. Austrian Computer Society (book series of). Edit. by Gordon Davies, pp. 859-867, Viena, Austria. 1998.

[3] [Marín et al, 1997] Marín R, Sanz P.J., Coltell O., et al. *Student-teacher communication directed to computer-based learning environments.* Displays, Elsevier Science. Special Issue on Displays for Multimedia (17) pp. 167-178. 1997. Abstract available on http://www.system-concepts.com/displays/

One of the most difficult concepts that students have to learn in the Robotics classroom is, for example, the kinematics of a generic robot arm. As seen at Figure II-40 and Figure II-41 the Robotics Tutorial treats these concepts specially, with the aim to let student have as much information as possible in order to get expertise in the "Reference System Assignment" and the "Denavit-Hartenberg Notation".

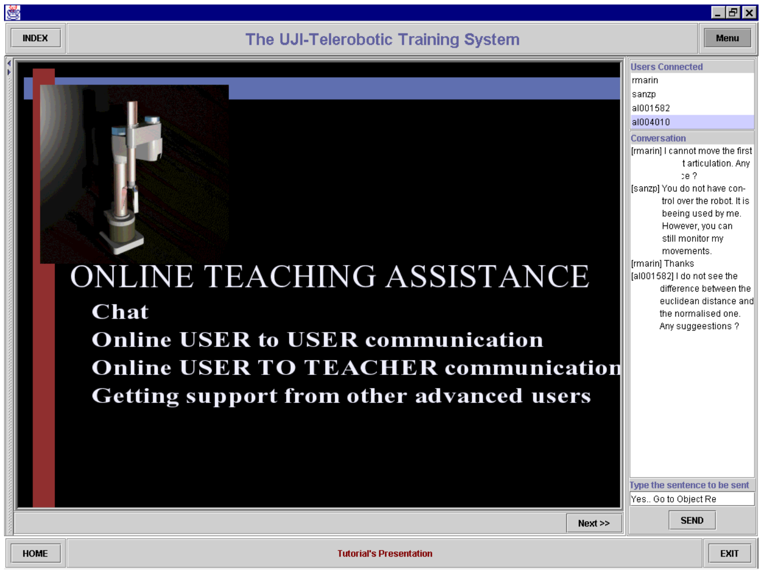

Undergraduate Robotics Course - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///E|/E80_Proyectos/FMIR/project/f_den_har.html What's Related
Programming BioMedical Strudata Organismos_Inve MMedia Robotics Instant Message Members WebMail
Denavit-Hartenberg Notation
Voice Stop
θ_{i-1}
link i-1
θ_i
link i
θ_{i+1}
α_i
z_i
a_i
x_i
d_i
z_{i-1}
x_{i-1}
θ_i
Exit
Begin Go Back Next
Document: Done

The Denavit-Hartenberg notation provides a mathematical method to model an opened kinematic chain, setting a reference system on each of the nodes.

Figure II-41. Robotics Tutorial to access the "Denavit-Hartenberg Notation" subject

7.2. *Online Teaching Assistance*

Basically, this capability allows the teacher or other advanced users to assist online when needed, by means of a chat access by every user connected to the training system at a particular moment. In Figure II-42 can be seen a snapshot of the Online Teaching Assistance feature.



ONLINE TEACHING ASSISTANCE
Chat
Online USER to USER communication
Online USER TO TEACHER communication
Getting support from other advanced users

[rmarin] I cannot move the first t articulation. Any ce ?
[sanzp] You do not have control over the robot. It is beeing used by me. However, you can still monitor my movements.
[rmarin] Thanks
[al001582] I do not see the difference between the euclidean distance and the normalised one. Any suggeestions ?

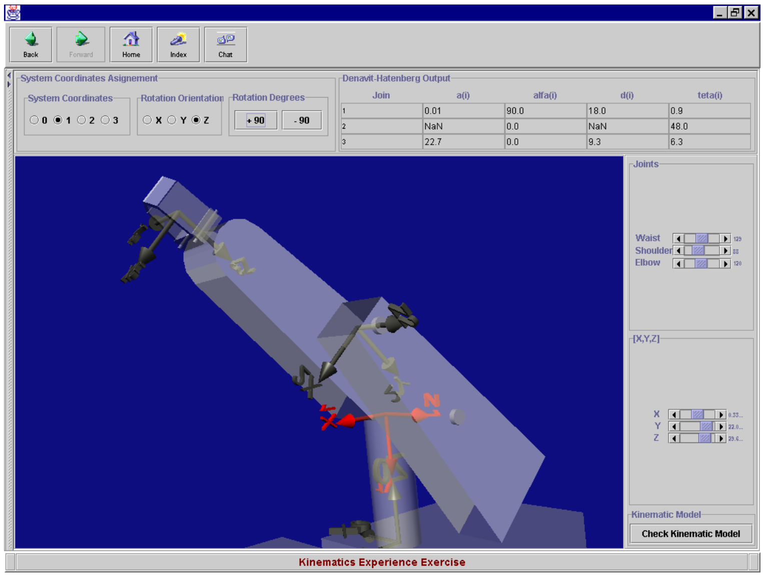
Yes.. Go to Object Re

Figure II-42. Online Teaching Assistance capability

The Chat functionality is very useful for the students who can in this way share ideas and even compare solutions to the “Pick and Place” exercices proposed at the laboratory sessions (see Chapter VII). Moreover, as people have the opportunity to know who is connected to the training system at any given time, it gives them sense of belonging to a classroom, which has prooved a highly motivating factor for the students.

7.3. *Robotics Lab Complementary Exercices*

It consists of several specific practices for the undergraduate Robotics course that introduce concepts like “Kinematics”, “Image Capturing”, “Image Processing”, etc. At the moment, two complementary exercises are proposed "Kinematics Experience", and "Image Capturing and Segmentation " (see Figure II-43 and Figure II-44). The first one allows the user to practice the calculation of several kinematics for the robot and see if they work properly in the off-line virtual 3D environment. The second one treats several aspects related to the construction of a Visually Guided Robot. In Figure II-44, for example, a window can be seen that allows the student to learn the difficulties associated to illumination and binarization, in order to accomplish a proper segmentation of the scene. This window permits the adjustment of several factors like binarization threshold, contrast and brightness in order to experiment in real time the effects of these items on a real vision guided robot. Besides this, some mathematical calculations like the image histogram are offered, that allow the student to relate his practical experience with the theoretical concepts explained in the classroom.



System Coordinates Asignement

System Coordinates: ○ 0 [✓ ● 1] ○ 2 ○ 3

Rotation Orientation: ○ X ○ Y [✓ ● Z]

Rotation Degrees: + 90 | - 90

Denavit-Hatenberg Output

Join	a(i)	alfa(i)	d(i)	teta(i)
1	0.01	90.0	18.0	0.9
2	NaN	0.0	NaN	48.0
3	22.7	0.0	9.3	6.3

Joints

Waist	129
Shoulder	88
Elbow	120

[X,Y,Z]

X	0.33...
Y	22.0...
Z	29.6...

Kinematic Model

Check Kinematic Model

Kinematics Experience Exercise

Figure II-43. Kinematics experience

As explained before, one of the major problems that students find when studying the Robotics course is the Kinematics. Some kind of interactive tool was required that shows the student the way reference systems are assigned to every joint in order to obtain the Denavit-Hartenberg parameters.

On the other hand, by providing the student with such a tool that enables him to program a robot by using a very high level of interaction, it means they do not realize the set of interesting concepts that must be taken into account. For example, the image segmentation is one of them, and letting the student see what happens when selecting a different binarization threshold has increased the way they understand computer vision techniques.

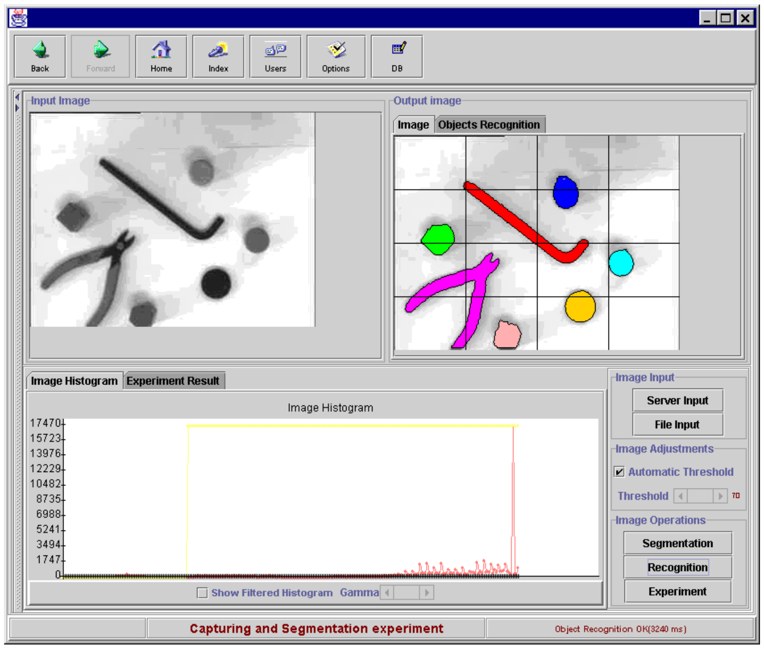
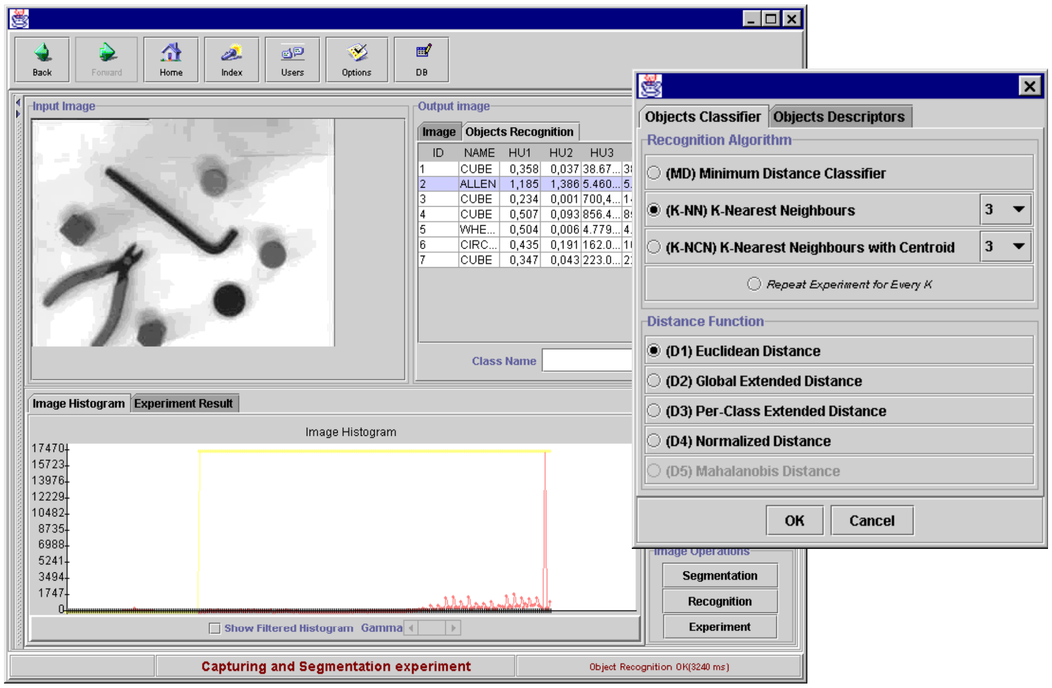


Figure II-44. Image capturing and segmentation exercises



ID	NAME	HU1	HU2	HU3
1	CUBE	0,358	0,037	38.67...
2	ALLEN	1,185	1,386	5.460...
3	CUBE	0,234	0,001	700,4...
4	CUBE	0,507	0,093	856.4...
5	WHE...	0,504	0,006	4.779...
6	CIRC...	0,435	0,191	162.0...
7	CUBE	0,347	0,043	223.0...

Recognition Algorithm
- (MD) Minimum Distance Classifier
- [✓ (K-NN) K-Nearest Neighbours] 3
- (K-NCN) K-Nearest Neighbours with Centroid 3
- Repeat Experiment for Every K

Distance Function
- [✓ (D1) Euclidean Distance]
- (D2) Global Extended Distance
- (D3) Per-Class Extended Distance
- (D4) Normalized Distance
- (D5) Mahalanobis Distance

Figure II-45. Configuring the Object Recognition properties in order to enhance its accuracy for the given scene

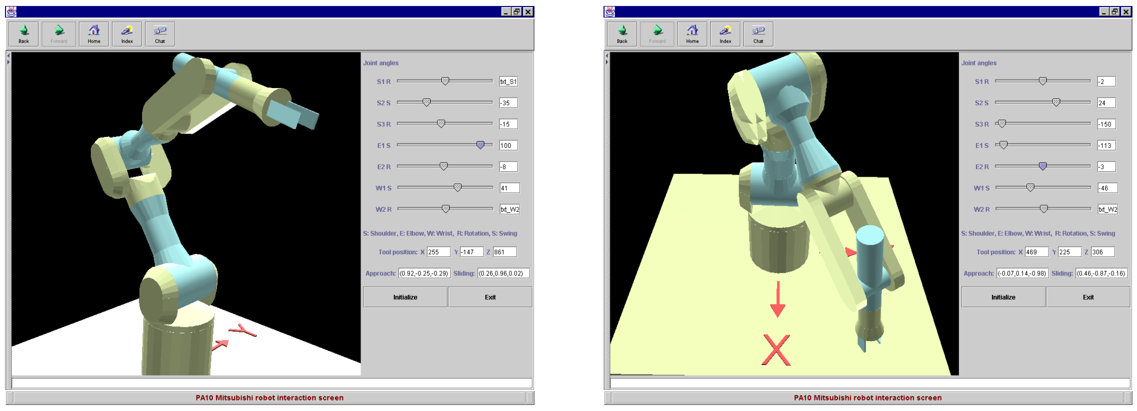
Finally, as shown in Figure II-45 once the image is segmented the student is able to execute the Automatic Object Recognition procedure in order to see if the illumination parameters affect or not the proper recognition of the objects in the scene. Besides this, by pressing the Properties button, the user is able to select among several Recognition Algorithms, Distance

Functions and even the Object Descriptors, that the user wants to use for his interaction with the teleoperated robot.

7.4. Industrial Robots Simulation

The UJI Telerobotic Training System offers the students the possibility of interacting with an educational Mentor robot through the web, using both an off-line and on-line connection. After that, we considered it would be very interesting for the students to have access to industrial robots too. Obviously, the on-line interaction with these arms could not be offered due to the fact that they were being used in other research projects. However, we could provide to the student with the possibility of interacting with several industrial robots in a simulated manner.

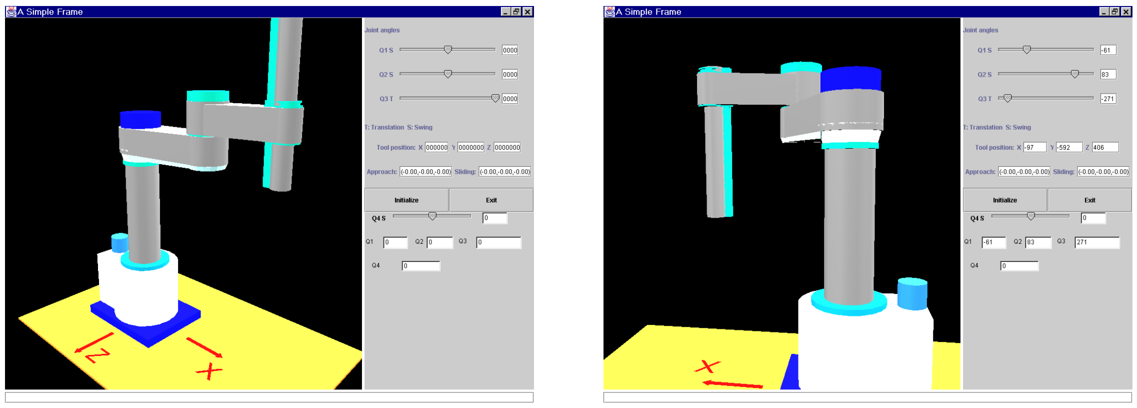
And that is what we did! We prepared 3D virtual environments for the Mitsubishi PA-10 robot and the AdepOne Scara manipulator. With them, students have been able to understand the differences between educational and industrial robots, in terms of the manipulation possibilities, work area, and multiple degrees of freedom (see Figure II-46 and Figure II-47).



Joint angles
S1 R txt_S1
S2 S -35
S3 R -15
E1 S 100
E2 R -8
W1 S 41
W2 R txt_W2
S: Shoulder, E: Elbow, W: Wrist, R: Rotation, S: Swing
Tool position: X 255 Y -147 Z 861
Approach: (0.92,-0.25,-0.29) Sliding: (0.26,0.96,0.02)

Joint angles
S1 R -2
S2 S 24
S3 R -150
E1 S -113
E2 R -3
W1 S -46
W2 R txt_W2
S: Shoulder, E: Elbow, W: Wrist, R: Rotation, S: Swing
Tool position: X 469 Y 225 Z 306
Approach: (-0.07,0.14,-0.98) Sliding: (0.46,-0.87,-0.16)

Figure II-46. 3D virtual environment for the Mitsubishi industrial robot



Joint angles
Q1 S 0000
Q2 S 0000
Q3 T 0000
T: Translation S: Swing
Tool position: X 000000 Y 0000000 Z 0000000
Approach: (-0.00,-0.00,-0.00) Sliding: (-0.00,-0.00,-0.00)
Q4 S 0
Q1 0 Q2 0 Q3 0
Q4 0

Joint angles
Q1 S -61
Q2 S 83
Q3 T -271
T: Translation S: Swing
Tool position: X -97 Y -592 Z 406
Approach: (-0.00,-0.00,-0.00) Sliding: (-0.00,-0.00,-0.00)
Q4 S 0
Q1 -61 Q2 83 Q3 271
Q4 0

Figure II-47. 3D virtual environment for the AdeptOne industrial robot

8. Summary

The robot scenario has been presented, which allows the manipulation of real objects on a board by means of an educational robot. The whole system functionality has been described by means of the user interface. Thus, the way the user interacts with the system using a 3D virtual and augmented reality interface has been described.

Moreover, the importance of a predictive configuration has been justified, which avoids time delay effects over the Internet. In fact, users can define a whole assembling task on the 3D virtual environment and then sending the whole specification to the real robot in a single step.

As the system allows different levels of manipulation (joints, world coordinates, high level commands, voice input, etc), it means the user is able to select the most convenient for a given situation. For example, for normal situations where objects are isolated on the board, in order to acquire an object a single command like "Grasp allen" would be enough. It means the user interacts in a supervised form, avoiding having to control every robot movement at any time. On the other hand, if the robot intelligence is not great enough to accomplish a particular task (e.g. grasping occluded objects), the user has the opportunity to program the robot in a more teleoperated manner. It means the operator can control directly the joints, gripper position, etc., in order to perform the task.

After that, examples of commands applied to the execution of complete Pick & Place actions have been shown. In particular, several invariant to position situations have been presented, which allow the execution of the same task independently of the objects' distribution on the scene. These kinds of situations open the doors to new interesting and challenging research fields, like for example, the qualitative spatial reasoning.

Finally, as the environment has been applied to the Education and Training domain, several complementary screens have been implemented that help students to understand the difficult concepts that enable programming a robot in a friendly manner. First, the "HTML Robotics Tutorial" introduces the theoretical concepts, then the "Online Teaching Assistance" makes students feel as if he (or she) belongs to a class, moreover some complementary exercices help to introduce the aspects necessary to implement a vision guided robotic system (i.e. "Image Processing", "Kinematics", etc.), and finally the 3D virtual environments for industrial robots allow the students to interact with more sophisticated manipulators in a simulated manner.

References

[Adept] (http://www.adept.com/main/products/robots/index.html)

[Alecop] (http://www.alecop.es)

[Cave] (http://www.evl.uic.edu/EVL/VR/systems.shtml)

[Dvantech Sonar] (http://www.acroname.com/robotics/parts/R93-SRF04.html)

[Fakespace] (http://www.fakespace.com)

[GP2D15] (http://www.acroname.com/robotics/parts/R49-IR15.html)

[Gupta et al., 1998] K. Gupta, A. P. del Pobil (ed.), "Practical Motion Planning in Robotics: Current Approaches and Future Directions", John Wiley & Sons, New York, 1998.

[Marín et al, 1997] Marín R, Sanz P.J., Coltell O., et al. *Student-teacher communication directed to computer-based learning environments*. Displays, Elsevier Science. Special Issue on Displays for Multimedia (17) pp. 167-178. 1997. Abstract available on http://www.system-concepts.com/displays/

[Marín et al., 2001] R. Marín, P.J. Sanz. *The UJI Telerobotic Training System. 1st EURON Workshop on Robotics Education and Training (RET2001),* Alicia Casals, Antoni Grau, Grup Artyplan-Artymprès, S.A., Weingarthen, Germany, 2001.

[PA10] (http://www.mitsubishitoday.com/1/PA10.htm)

[Pobil et al., 1995] A. P. del Pobil, M. A. Serna, "Spatial Representation and Motion Planning", Springer-Verlag, Berlin, 1995.

[Polaroid Sonar] (http://www.acroname.com/robotics/info/articles/sonar/sonar.html)

[QuickTime] (http://www.apple.com/quicktime/qtvr/index.html)

[Sanz et al., 1998] Sanz P.J., Adell S. An undergraduate Robotics Course via Web. Teleteaching'98 Conference, a part of the 15th IFIP World Computer Congress. Distance Learning, Training, and Education. Austrian Computer Society (book series of). Edit. by Gordon Davies, pp. 859-867, Viena, Austria. 1998.

[Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

[Sowizral et al., 2001] H. A. Sowizral, M. F. Deering, Projects in Virtual Reality: The Java 3D API and Virtual Reality, Sun Microsystems (http://java.sun.com), 2001.

[VR Lab Michigan] University of Michigan, Virtual Reality Laboratory (http://www-vrl.umich.edu/)

[VRML] (http://www.web3d.org/vrml/vrml.htm)

[Web3D] The Web3D Consortium Repository (http://www.web3d.org/vrml/vrml.htm)

Chapter III SYSTEM ARCHITECTURE

This Chapter describes the high and low level distributed architecture for the UJI Telerobotic Training System. First of all it focuses on the hardware architecture, and the way multiple clients are able to control multiple servers (robots), via the web.

Secondly the Chapter presents the low level architecture, showing the different modules present on both, client and server side, as well as the way they interact with each other (sockets, RMI, CORBA, JDBC, etc).

Finally, we will focus on the multirobot architecture configuration. We will see which parts of the system should be duplicated in order to control different types of robots (Mentor, Adept, Mitsubishi) by using the same user interface, and which parts can be shared (i.e. the robots knowledge).

Topics:

1. Hardware architecture (Multiple clients / Internet / Multiple servers).
2. Software architecture.
   - Cameras Server
   - Robot Server
   - Database Server
   - Web Server
   - Users Manager Server
   - Computer Vision Module
   - Object Recognition Module
   - Speech Recognition Module
   - Natural Language Commander Module
   - Virtual 3D environment Module
   - Task Specification Module
3. Multirobot configuration

1. Introduction

In recent years, the approach used for most application development has shifted from client/server to three-or n-tier applications. In a multi-tiered design, the application is usually broken down into three components or "tiers": a presentation/user interface component, a business logic component, and a data storage and retrieval component, typically represented by a database. This type of architecture is usually more flexible and reusable than a similar client/server design, and obviously it does increase the complexity of the application. One of the issues that must be addressed is how to allow the user interface and business logic components to communicate with one another, since it is common (and not necessary) for those two components to reside on separate machines. In addition, it is sometimes desirable for different parts of the business logic (or parts of the database) to reside on separate machines [Spell, 2000][1].

When different pieces of an application reside in separate physical machines, the program is known as a *distributed application*, and a *distributed object* is simply an object that allows its methods to be called by processes running on different machines, or communicating across different process spaces.

When designing applications to be run over the World Wide Web, several considerations must be taken into account. First of all, the user interface component technology must be selected, which can belong to one of the following groups:

- **Server-based technology**: This approach means the whole user interface (HTML page) is generated on the server side depending on the current state of the system and the current user input. It means that after the user has selected an option (or pressed a button) on the web page, the operation is sent to the server-side and the updated HTML page to be shown to the client is reconstructed accordingly. For this situation, it is the server side which knows the user interface component logic, and the client side simply shows the results acquired from the server. An example of this approach is the CGI (Common Gateway Interface), or more advanced ones like PHP and Servlets. Some Internet robots (e.g. Telegarden) and many interactive web pages are based on these technologies.

- **Client-based technology**: Another approach consists of implementing the whole user interface component logic on the client side, by using more advanced alternatives like Java. For these situations the architect is able to select which part of the system runs on the client, and which others on the server. It means Java combined with some other communication technologies like RMI (Remote Method Invocation), CORBA (Common Object Request Broker Architecture), and even TCP/IP (sockets), allows the design of real distributed applications over the World Wide Web.

Obviously the second alternative has many advantages, like for example:

[1] [Spell, 2000] B. Spell, Professional Java Programming, Wrox Press Ltd, EEUU, 2000.

1. *Application latency*. Instead of having to interrupt the server every time, the user has selected an option on the user interface, client-server communication is only established when information is needed (e.g. the real robot position coordinates have changed). It means the network bandwidth required for a given application is greatly diminished, and obviously the overall system performance is increased enormously.

2. *Server load*. As many operations are already accomplished on the client side, it means the server is able to perform its tasks more rapidly, and even give service to more users at the same time. Basically, the server side hardware requirements are diminished.

3. *System reliability and flexibility*. For those situations where a server side service is down (e.g. robot already used by someone else), the client side can still accomplish other kinds of alternative services (e.g. simulated 3D virtual environment). Besides this, when using a distributed approach, it is easy to move components from a computer to another, depending on the actual system requirements. For example, if we realize the server side is having many connections at the same time and it has the risk of being overloaded, we still have the possibility to move some logic to another machine (or even the client side).

4. *Java advanced APIs*. Due to the important success of Java, many interesting APIs have been included into the language. For example, the Java3D API allows the implementation of full virtual reality models that enhance greatly the user interface. On the other hand, Java Sound and Java Advanced Imaging allow the design of great Multimedia interfaces.

On the other hand, some drawbacks are still present, like for example:

1. *Program Launching*. Once connected to the web page, the whole Java applet (compiled code) must be downloaded to the client side. For example, the UJI Telerobotic Training System Java code has a total size of 954Kb. Taking into account some external libraries, it makes a global size to be downloaded of almost 2Mb of information (considering the Java3D plugin is already installed). Thus, it means some initialization delay is introduced into the system depending on the particular bandwidth (3 seconds on campus and even minutes when using modem connections).

2. *Java Maturity*. Since the Java programming language was launched at 1995, many versions and improvements have come up since then. It means many browsers do not yet have the lastest Java Virtual Machine installed. Thus, users are required in many situations to update their plugins in order to access the contents of a given web page.

In our opinion, the advantages justify enormously the drawbacks. First of all, the program launching effect can be reduced by using a greater bandwidth, or simply by using some software techniques like compressing the Java byte codes. In any case, having to wait once at first is much better than waiting all the time. Moreover, Java is gaining more maturity as time goes by. In fact, the JDK 1.3.1 Java compiler has given a robustness, performance, and flexibility to the language that makes the use of Java useful as a way to implement research and development projects.

2. Hardware Architecture

The hardware architecture (see Figure III-1) is based on two main parts, the server and client side. The physical connection between them is accomplished through the Internet by means of the TCP/IP protocol (particularly RMI). As explained at Chapter I, the Internet has the advantage of allowing public participation. It means anybody with a computer and a modem would be able to access the real robot, independently of the person location.

The client side holds the "remote controller application" and allows the user to invoke commands in a combination of a simplified natural language specification (i.e. "Grasp the screw") and mouse controls. Note the user has the option of using a microphone and speakers on the client side, in order to program the robot by means of voice commands. See Chapter VI for more details.

The server side is composed of two servers, the Mentor and the Mitsubishi server. The first one consists of an educational Mentor robot, a distance sensor, three LCD cameras, a remote controlled switch (illumination control), and a Java based Database that holds the already learnt objects' descriptors (see Chapter IV). Moreover, the Mitsubishi server consists of an industrial Mitsubishi robot, and a stereo camera pair input held on top of the gripper. The Mentor server devices are directly controlled by using a Pentium III 450/128 Mb Windows 98 PC, and the Mitsubishi server uses a Pentium II 400/128Mb Windows NT PC.

Please note the Mitsubishi and AdepOne robots are not fully accessible on-line from the Internet at the moment, because some implementation works have yet to be completed. However, the general multirobot architecture is already defined and that is why is being presented in this Chapter.

Basically, the server side architecture is organised by using a dedicated server computer for every robot being accessed remotely. For example, if we decide to connect an AdeptOne industrial robot to the Internet (we are actually doing it at this moment), we would need an AdeptOne robot server with the different cameras and sensors controllers connected to it. Obviously, this server computer must have access to the Internet itself.

On the other hand, there are some modules that can be shared by every robot server. The objects database is held in a single computer, and shared by any robot belonging to the UJI Telerobotic Training System. It means whenever a new object is learnt by the interaction of a user with any of the robots (e.g. Mentor), this new knowledge is passed on to the Mitsubishi and the AdeptOne robots too. This is due to the fact that the object descriptors utilized (HU, Rv, Lv, Tr, etc.) are invariant to scale, rotation and translation, so they are also invariant to different cameras calibrations, as long as they use the same configuration from the top of the scene. Moreover, there are some aspects that can affect the proper recognition of the same object from two different robots, like, for example, illumination variations or even noise introduced by the digitizing process. In Chapter IV some experiments on object recognition that take into account these situations are presented.

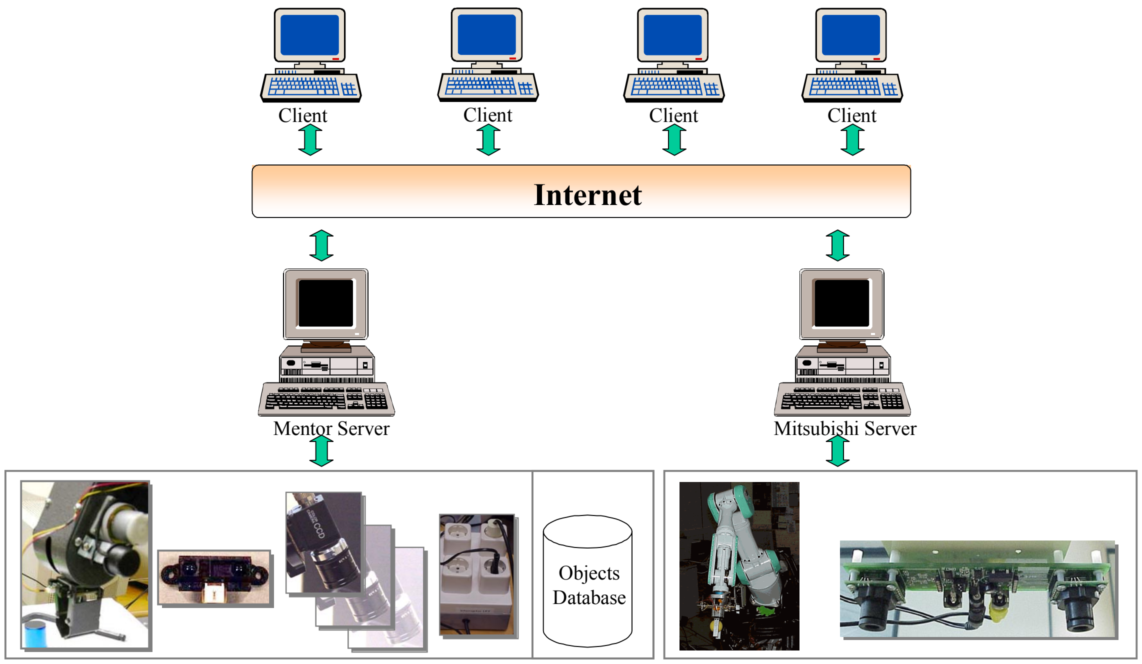


Figure III-1. Hardware architecture: Two robot servers, the Mentor and the Mitsubishi server. The Objects Database is shared and held on the Mentor server.

The distance sensor presented in Figure III-1, is connected directly to the robot interface, and then controlled by the Mentor Server using the same controller circuit as the robot itself. The Mentor robot scenario is provided with four remotely operated switches in order to illuminate the scene whenever a user is going to control the robot. These switches are connected to the Mentor server through the parallel port, and can be programmed by writing directly into that port specifying the switches that we want to activate. At the moment, two lamps are programmed remotely, the idea is (in the future) to allow the connection of other devices like the cameras, and even the robot (depending on the circuitry characteristics).

3. Software Architecture

As we have said, the telerobotic system allows the manipulation of objects on a board by means of mouse interactions on the 3D virtual reality environment and also by using a simplification of the natural language. Thus, the system can respond to commands like "Pick up the scissors" and "Grasp object 1". This kind of interaction is possible thanks to an optimised object recognition CORBA module that processes the camera images and returns every objects name. As the program is able to learn new object characteristics through the user interaction, the system becomes more robust as time goes by. As introduced above, such a capability has not been reported in a Telerobotic system yet [Goldberg et al., 2001][1][Taylor, 2000][1][Marín, 2002b][2]. In Figure III-2 the telerobotic system's software architecture is presented.

[1] [Goldberg et al., 2001] K. Goldberg, Roland Siegward, Beyond Web Cams: An introduction to Online Robots, MIT Press, Massachusetts, 2001.

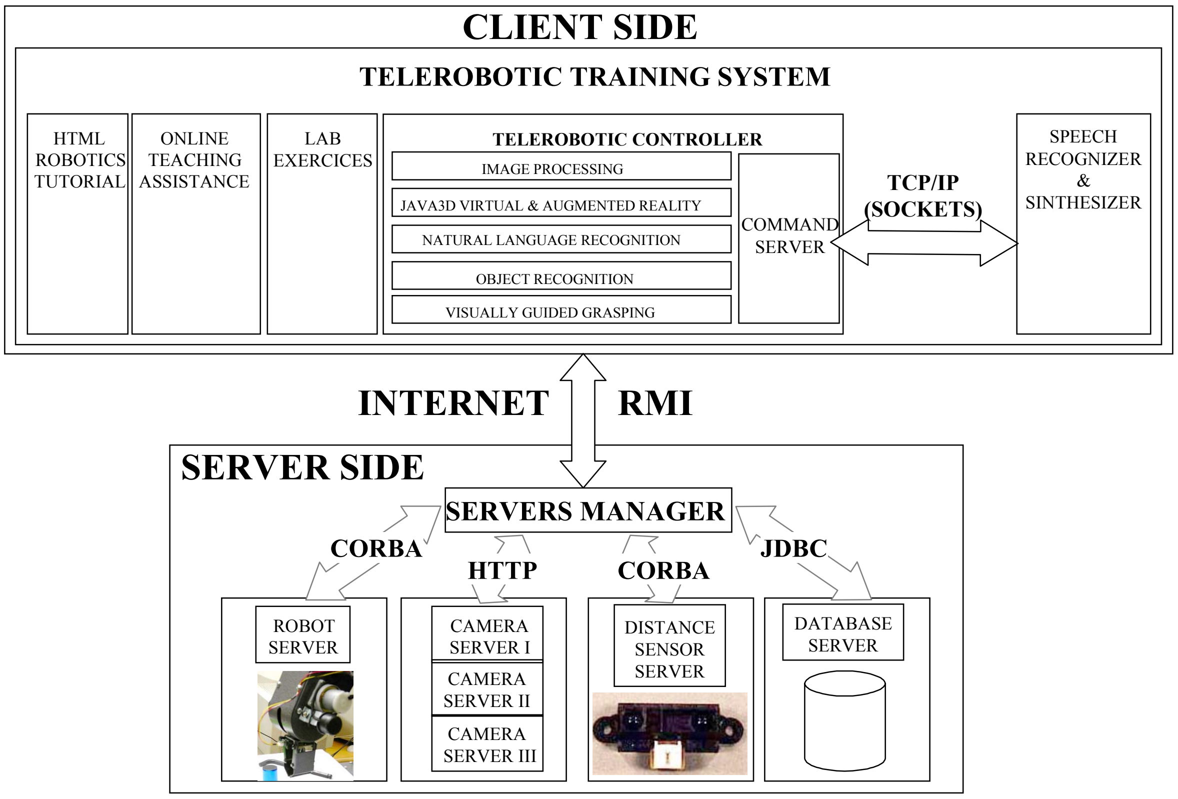


Figure III-2. Software architecture

Note the software architecture is organized in several modules connected through several protocols: CORBA, RMI, HTTP and JDBC. The CORBA standard is the most used and makes easier the integration of already existing software, implemented in different programming languages and running over distinct platforms.

The system is structured on both the client and server side. The client side consists of a single process implemented in Java and running through a web browser, and an optional executable file that allows the user to interact with the robot by means of the voice. If we prefer to avoid the applet downloading time, another possibility is installing the Java application into the client machine and running it directly from the command line. This second possibility significantly increases the client performance, and requires a previous installation procedure on the client computer before using the application. The server side consists of several concurrent processes running on the server machine and interacting through the CORBA, JDBC and HTTP standards.

[1] [Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[2] [Marín et al., 2002b] R. Marín, J.S. Sanchez, P.J. Sanz. Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System. In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

3.1. The client Side

The client side is organized in five modules: the HTML Robotics tutorial, the online teaching assistance (chat), the laboratory exercices (robots kinematics, computer vision, etc.), the telerobotic controller, and the speech recognizer and sinthesizer module. The speech recognizer and synthesizer are organized on a single Windows-based executable program that can be used optionally by the user.

3.1.1. HTML Robotics Tutorial

The first module is the "*HTML Robotics Tutorial*", which consists of a set of HTML pages that includes a well-classified description of the main topics of the Robotics subject. For example, details to calculate the direct and inverse kinematics of a generic manipulator are given, which are necessary concepts to comprehend the practical laboratory lessons.

3.1.2. Online Teaching Assistance

The "*Online Teaching Assistance*" allows the chat interaction between the different users connected to the Training System. Thus, students can share ideas by using the same training system, and even ask the teacher (system administrator) questions referring to the laboratory exercises. This is an interesting way of allowing collaborative learning in the education and training applications.

3.1.3. Laboratory exercices

The "*Lab Exercises*" consists of several screens that enable students to practice some advanced features like "Kinematics", "Image Segmentation", and "Object Recognition", allowing them to discover in the 3D virtual Robotic environment the consequences of their mathematical decisions. For example, the Object Recognition screen allows student to appreciate the effect of selecting a given object descriptor (e.g. Tr) in the recognition process. The illumination intensity effect on the image segmentation can be distinguished by using the Image Segmentation laboratory screen.

3.1.4. Telerobotic controller

The "*Telerobotic Controller*" offers access to the virtual and real robot by using the advanced interaction explained before. Thus, move and place tasks can be programmed using both the online and offline user interaction, by having the particularity of implementing augmented reality features and predictive techniques, which are help enormously to avoid the Internet latency effect.

As can be seen in Figure III-2 the Telerobotic Controller is divided in six submodules. "*Image Processing*", "*Java 3D Virtual & Augmented Reality*", "*Natural Language Recognition*", "*Object Recognition*", "*Visually Guided Grasping*", and "*Command Server*".

The first one, "*Image Processing*", gives some services for capturing and segmenting images. It obtains a camera image as input and gives to the output a list of the objects belonging to the scene, as well as their mathematical representations (perimeter, contours, object descriptors, etc.). See Chapter IV for more details.

The second, "*Java 3D virtual & Augmented Reality*", implements a 3D virtual environment of the robot scenario, which allows a human-robot communication through a non-immersive VR interface, as well as some augmented reality capabilities that enhance the information gotten from the cameras with computer generated information. Thus, for example, the projection of the GRP over the board is included into the virtual world, as well as some capabilities that allow the user to interact with the objects even though the robot is occluding the objects' scenario. Another contribution of this module is the capability of constructing 3D model representations of the scene objects from the top camera input. In order to calculate the heights, it is necessary to use a proper sensor on the gripper or simply ask the automatic object recognition module to provide the height. See Chapter II for more details on the 3D model construction.

The "*Natural Language Recognition*" module consists of several JAVA classes that are able to interpret a simplified natural language command from the user, specified in a text manner. It translates this command into the appropriate sequence of actions from within the remote controller. Examples of commands that can be interpreted are: "*Grasp object 1*", "*Grasp allen*", "*Refresh*", etc.

The "*Command Server*" offers a way for external applications to send commands to the Natural Language Recognition module. It means we are able to implement a Speech Recognition module that uses the Command Server as interface to control the robot by spoken commands. This kind of feature is very common on TMN (Telecomunications Management Networks), and in that context, they are well known as *northbound interface* [Marín et al., 1998][1][Marín et al., 1999][2]. Thus, we can consider the Command Server as a northbound interface that allows access to the robot control from third-party applications. This interface has been implemented by using TCP/IP sockets, which means the Telerobotic Controller opens a port on the client side machine (port 7745) in order to service robots movements specified externally. Please note in order to allow a Java applet to open a port on the client machine, it is necessary to have a certified applet, and then let user accept the certification as a trusted source.

The "*Object Recognition*" module is able to identify the different objects present in the scene in a fast and robust manner. It gets the Image processing results as input, and then compares the existing values on the actual image with an already learnt database of objects, providing finally the set of objects' names on the scene (e.g. screw). As explained at Chapter IV, the object recognition is one of the novel contributions to the web Robotics domain.

And finally, the "*Visually Guided Grasping*" is another novel contribution that has the particularity of calculating from the image processing output the set of possible grasping points for every object, that assures the grasping process will be carried out under stability constraints. For more information about this module, please refer to Chapter VI.

[1] [Marin et al., 1998] "Aplicaciones Distribuidas Orientadas a Objetos como Medio para conseguir la Calidad Total: Standard CORBA", Proceedings of the "VII Jornadas Técnicas de Calidad en Tecnologías de la Información (CIECAT´98)", *Telefónica I+D, Madrid, 1998.* (http://www.tid.es/calidad/ciecat/contra.html)

[2] [Marin et al., 1999] "Gestores de Red basados en CORBA: Un caso real", Proceedings of the IX Telecom I+D Congress, Madrid (Spain), 1999. (http://www.telecomid.com)

3.1.5. Speech Recognizer and Synthesizer

At this point, we have a Java applet capable of executing robot actions from a text command taken as input. The idea now is to allow the generation of these commands from voice input. It would make the way users interact with the system more sophisticated. As explained at Chapter VII, the Speech Recognizer and Synthesizer program runs on the client-side, and connects to the robot controller through the northbound interface (port 7745 on client machine). Thus, when user says the command "*Grasp object 1*", the Speech Recognizer converts it into a string of characters and sends it to the robot controller by means of the northbound interface.

3.2. The Server Side

At the beginning, the connection between the client side and servers were accomplished under the CORBA standard, through the web, by tunneling the distributed objects along the HTTP port (80). This configuration allowed, for example, the connection to the robot of people being connected under a firewall. The situation was managed by using the CORBA HTTP tunneling offered by "Visibroker Gatekeeper". As every information to be transmitted had to be previously tunneled on the common 80 port, the performance of the client/server connection went down considerably (four times slower).

After that, we spent some time improving the client/server communication procedure, and we realized that by limiting the number of connections through the web we could shorten the response time. This was better than allowing the client-side to connect directly to the remote CORBA servers. Thus, as seen in Figure III-2, the Server Manager module was created, which has the ability of to manage the whole servers' services through a single connection. Besides this, in order to allow more flexibility, we decided to implement this module in Java, allowing the communication with the client side by means of the RMI standard. The Remote Method Invocation is a simple and quick way to implement a client/server communication, where both client and servers are implemented in Java. Another advantage of using RMI is that it is already included in the Java runtime on the client machine. When using CORBA, the communications libraries must be downloaded as well, so the Java applet initialization process increases notably. Moreover, for those situations where another programming languages has to be used, RMI has the alternative of communicating to CORBA services too, by using IIOP (Interoperability Protocol). Some other telerobotic systems such as UWA use RMI as well.

In the server side, there are several modules: The "Robot Server", "Cameras Servers", "Distance Sensor Server", and "Database Server".

3.2.1. Robot Server

The first one is the "*Robot Server*", that accepts CORBA requests to move the real robot to a given world position (x,y,z), managing directly the values for the joints, as well as controlling the opening of the gripper. This CORBA server is used by the Servers Manager in order to allow clients to control the real robot movements.

3.2.2. Camera Server

The second one is called "*Camera Server*", and consists of a commercial product called "WebCam32" that offers a HTTP interface to the server cameras. An instance of this program must be launched for every camera connected to the system.

3.2.3. Distance Sensor Server

The distance sensor server connects to the robot digital controller in order to read the actual sensor value. Then, by using a calibration table, it generates to the output the distance from the gripper to the object. This information is very valuable for calculating objects' heights as well as avoiding possible collisions.

3.2.4. Database Server

And finally, as the system offers an object recognition capability so that it can accept simplified natural language commands, it is necessary to setup a database storing a mathematical description of every object already learned by the system. This database represents the robot knowledge, and is accessed by the multiple Java clients running over the Internet. It means the robot knowledge is common to the multiple users and, besides this, it is *robot independent*. Thus, once the database is trained for a given robot, it can be shared by other robots as long as the camera configuration used is from the top.

4. Multirobot Architecture

In the previous sections, we saw how to access a single robot by several remote controllers. We also discussed the difficult situations that would be created by several clients accessing the same robot at the same time, so just an operator can have control on the real robot while the others are programming the virtual environment.

As an alternative to this problem (having a unique robot and several users), a possible solution would be allowing access to more than one robot. In fact, in our laboratory we have three Mentor robots, which could be easily connected to the Internet in order to allow more students to program them.

The idea is well described in Figure III-3 and would consist of selecting from the remote interface, the robot we want to interact with. Obviously, it would not be possible for two controllers to access the same robot at the same time.

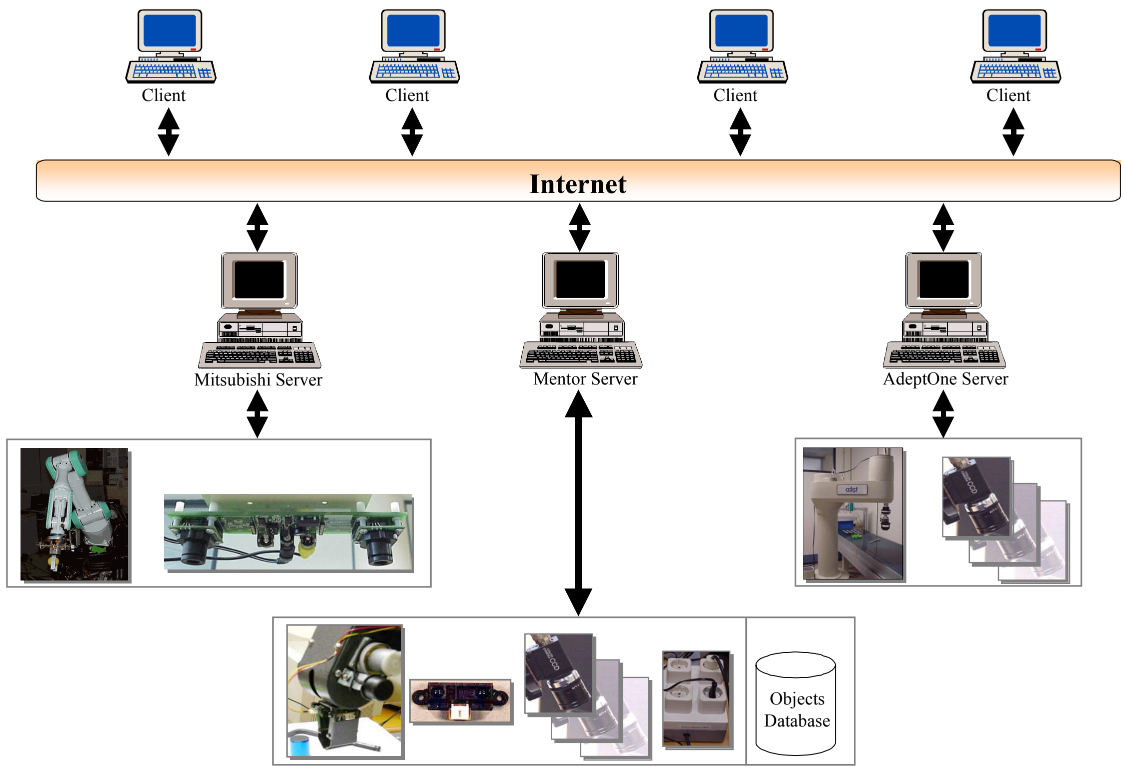


Figure III-3. Multirobot configuration

The important point in this configuration is the use of a single knowledge source. Remember that the objects database is shared between the clients and it is independent of having 3 or 4 dozens of robots in our system. *When a robot learns a new object it, is fully seen by the other ones in the following iterations, because all of them use the same database.*

5.Summary

The chapter has described the hardware and software architecture that has allowed the implementation of such a distributed telerobotic system. Many advanced techniques like CORBA, RMI, JDBC, and JAVA3D were introduced, which have been very important tools for the design of such a complex project.

In this chapter it has been proved that web based telerobotics is robust and powerful enough to control not only a single robot, but also multiple robots that are able to share some components like the "Robot Knowledge".

References

[Goldberg et al., 2001] K. Goldberg, Roland Siegward, Beyond Web Cams: An introduction to Online Robots, MIT Press, Massachusetts, 2001.

[Marin et al., 1998] "Aplicaciones Distribuidas Orientadas a Objetos como Medio para conseguir la Calidad Total: Standard CORBA", Proceedings of the "VII Jornadas Técnicas de Calidad en Tecnologías de la Información (CIECAT´98)", T*elefónica I+D, Madrid, 1998.* (http://www.tid.es/calidad/ciecat/contra.html*)*

[Marin et al., 1999] "Gestores de Red basados en CORBA: Un caso real", Proceedings of the IX Telecom I+D Congress, Madrid (Spain), 1999. (http://www.telecomid.com)

[Marín et al., 2002] R. Marín, P.J. Sanz., J.S. Sanchez, A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality. In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Marín et al., 2002b] R. Marín, J.S. Sanchez, P.J. Sanz. Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System. In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Spell, 2000] B. Spell, *Professional Java Programming*, Wrox Press Ltd, EEUU, 2000.

[Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

Chapter IV **AUTOMATIC OBJECT RECOGNITION**

For those situations in which the user wants to interact with the system by using, for example, voice commands, it would be convenient to refer to the objects by their names (e.g., "cube") instead of other types of interactions (e.g. "grasp object 1"). Automatic object recognition is the first step in order to acquire a higher level of interaction between the user and the robot.

Applying Object Recognition techniques when the camera images are being transmitted through the web is not an easy task. In this situation images can not have a very high resolution, which affects enormously the recognition process due to the inclusion of more errors while digitalizing the real image. In our case, images consist of (320x240) pixels, in gray scale, which means that an average of 4Kb has to be transmitted through the web.

First of all, we need to select a set of mathematical descriptors that will enable the system to decide to which class of objects a real sample (as detected by the camera) belongs. The idea is that the values of these descriptors are to be quite similar when the samples belong to the same class (e.g. different views of an Allen key), and quite different when the samples belong to distinct classes (e.g., a screwdriver and a scissors). The selected descriptors are Invariant Hu Descriptors (both surface and borders alternatives), Thinness ratio (Tr), Shape Elongation (Lv), Spreading (Rv), and Compactness (C). Secondly, we will introduce several classifiers and distance measures that can be used to decide which object class has to be associated with a given sample of scene object (camera input).

Results will show which combination of descriptors, algorithms and distances is more appropriate to our purpose, in terms of both effectiveness and computing time.

And finally, in order to allow the system to learn new objects representations as time goes by, the automatic incremental learning capability is presented. This is important because it permits to refine the object's representation database in an automatic manner, improving the recognition capabilities (training set editing), as well as maintaining or even reducing the computing time (training set condensing).

Topics:

1. Why do we need automatic object recognition?
2. Computer Vision: *Binarization, Segmentation, and Feature extraction.*
3. Object Descriptors: *Hu Descriptors (Surface and Borders approaches), Thinness ratio (Tr), Shape Elongation (Lv), Spreading (Rv), Compactness (C).*
4. Classifiers: Object Recognition Algorithms and Distances
5. Automatic Incremental Learning
6. Comparative Results

Never regard study as a duty, but as the enviable opportunity to learn to know the liberating influence of beuaty in the realm of the spirit for your own personal joy and the profit of the community to which your later work belong (Albert Einstein).

1. Introduction

The UJI Online Robot allows the manipulation of objects located on a board by means of mouse interactions and also by using a subset of the natural language (e.g., "*Grasp cube*"). As the program is able to learn new objects' characteristics through the user interaction, the system becomes more robust as time goes by. As introduced above, such a capability has not been reported in the frame of web robots, yet [Taylor et al., 2000][1][Goldberg et al., 2000][2].

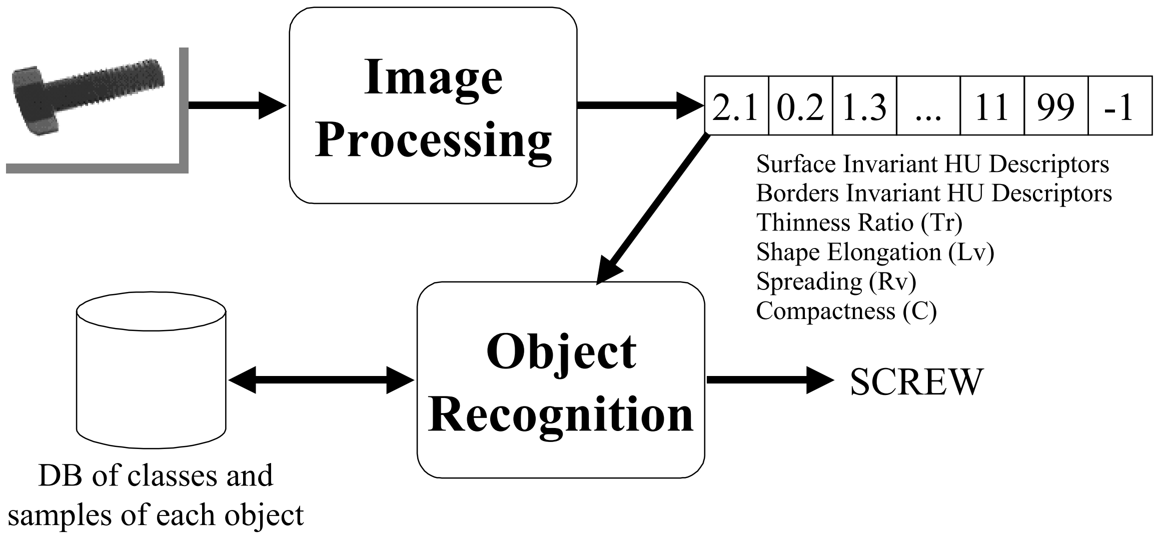


Figure IV-1. Simple Object Recognition procedure, from the image acquisition to the object classification using an already trained database

As can be seen in Figure IV-1, the simplest automatic object recognition paradigm works as follows: first of all, an image is captured with the camera. In this situation, the image is obtained from the top, so the object classification is based on 2D information. After this, the Image Processing module binarizes and segments the image in order to identify (isolate) the objects on the scene, and then it calculates a series of mathematical descriptors for every one of them. In the following section, the descriptors are explained in a more extensive manner. Once we have the mathematical representation of an object, we apply the object classification algorithms to compare this mathematical representation with the already learnt information stored on the database. Finally, the result will be the name of the class associated with our scene's object, in this case a "screw".

First of all, the chapter describes the computer vision algorithms here implemented in order to obtain mathematical object information from a bimodal image captured from a camera. It focuses on the binarization process, which has the property of being invariant to the illumination variations. Secondly, the chapter describes two of the main features of the web-based telerobotic system, object recognition and incremental learning. Within this context, the pres-

[1] [Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[2] [Goldberg et al., 2000] K. Goldberg, S. Gentner, C. Sutter, and J. Wiegley, "The Mercury project: a feasibility study for Internet robots: design and architecture of a public teleoperated system accessible by anyone on the Internet at any time", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

ent chapter investigates the feasibility of applying several distance-based classifiers to remote object recognition tasks. On the other hand, incremental learning [Natarajan, 1991][1] enables the system to maintain a representative set of past training examples that are used together with new data to appropriately modify the currently held knowledge. This constitutes a significant property because it allows a system to continually increase its knowledge and accordingly, to enhance its performance in recognition as it learns. And also, because this capability allows to detect an incorrect or anomalous employment of the specific remote system.

Hereafter, this chapter is organized as follows. Section II explains the computer vision methodology, which enables binarizing and segmenting a bimodal camera image in order to distinguish the existing objects on the scene. Moreover, it shows the mathematical representation of the moments associated with each object, which will be used later on by the object recognition procedure in order to calculate the final object descriptors (features). Section III provides an overview of the different set of descriptors that have been used to identify an object sample on a camera image. Some of them (e.g., Hu descriptors) are based on the previous calculated moments, while others (e.g. *Tr*) use computer vision information like the object area and perimeter. Section IV describes some classification algorithms applied to object recognition tasks. Section V presents a set of distance functions employed with the classification approaches. Section VI introduces an incremental learning procedure that allows the system to continually increase the knowledge utilized by the object recognition module. Experiments in Section VII evaluate the performance of the classification models combined with different metrics. Finally, conclusions and further work are outlined in Section VIII.

2. Image Processing

From the time an image is captured from the camera till the object recognition module can be applied there is still a lot of work to be done. This work consists of analyzing and interpreting a camera image in order to acquire a mathematical representation for every object present in the scene. *How can this be accomplished?*

The computer vision problem is that of devising a manner for a computer to be able to interpret pictures in a useful way. Naturally, the interpretation will depend on the current application. Simple applications may require only those areas in the image to be classified as light or dark, or that motion be detected. More involved applications demand that the precise three-dimensional coordinates of objects recognized in the image be obtained. Whatever the application, the first step in its solution is to characterize the problem as a numerical one (adapted from [Parker, 1992][2]).

Another problem with constructing algorithms for vision is that, although there are many practical, working vision systems to be found (most animals for example), it is not known how they operate. We appreciate that we can recognize the faces of our parents, but we do not have any idea of how we do it. Of course, years of research on animals and humans have provided a large body of knowledge about biological vision system, but this has not yet resulted in a high

[1] [Natarajan, 1991] B. Natarajan, *Machine Learning: A Theoretical Approach*, Morgan Kaufmann, 1991.

[2] [Parker, 1994] J. R. Parker, "Practical computer vision using C", E.E.U.U, John Wiley & Sons Inc, 1994.

level of visual sophistication in computers, and it would be impractical to wait until we know everything.

For our situation, in order to facilitate the image interpretation procedure, we assume that several situations are present:

1. In order to identify the existing objects, the illumination of the scene must be as constant as possible. It does not matter if the light varies from one hour to another, the algorithm is robust enough to deal with this. There would however, be a problem if any shadows were projected onto the scene as they might be interpreted as objects.
2. To facilitate the border extraction, we suppose there are no contiguous or overlapping objects. In the case of overlapping objects, as will be explained in Chapter VI, although the vision system is not adequate to deal with the difficulty, the robot manipulation can solve the problem.
3. The images must be as bimodal as possible, which means their colors should be organized in two big groups, the ones belonging to the background, and the ones belonging to an object. At the moment, the algorithm requires dark colors for objects, over a white background.

Once these three situations are accomplished the process correctness can be guaranteed.

In Figure IV-2 we can appreciate the three well-organized steps in order to extract object characteristics by interpreting an image.

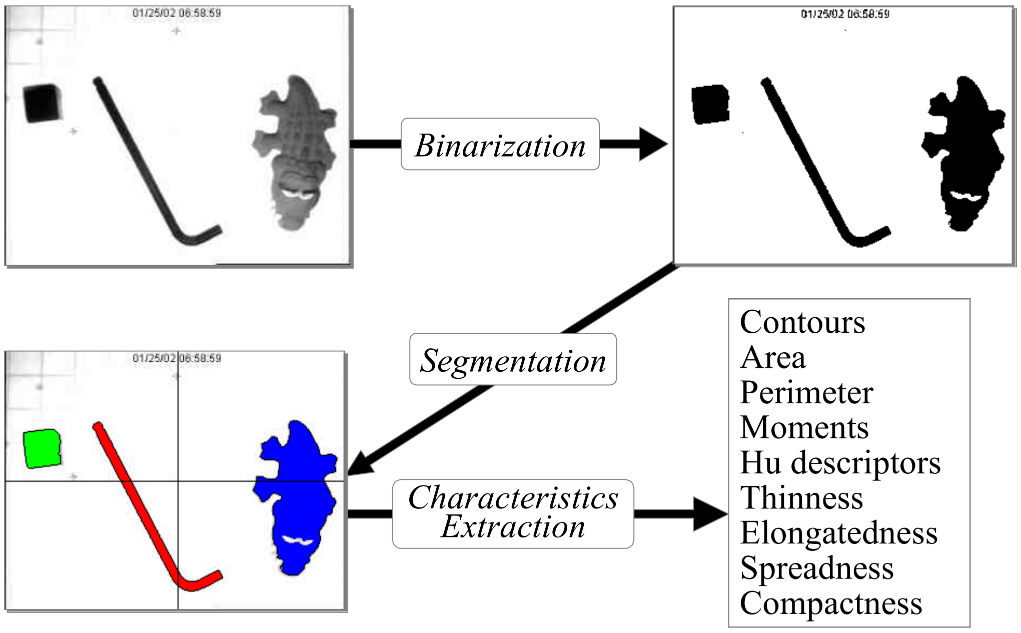


Figure IV-2. Computer Vision applied to extract object characteristics on an image

First of all, we get as input a bimodal image. In our case, gray level images are used in order to make the process work as fast as possible. Then we apply a *binarization* process that converts the gray scale image into a binary one (two levels, black and white). In this case, the resulting binary image consists of black pixels for the objects and white for the background. After that, the *segmentation* process gets as input the binary image and identifies the objects on the scene (labeling) and their contours. Finally, the *feature extraction* process allows the system to obtain the whole set of descriptors that are to be used by other modules like object recognition or grasp determination (Area, Perimeter, Moments, etc).

2.1. Binarization

The binarization process takes as input a gray scale image, and generates as output the corresponding binary one. For our case, the resulting pixels are black when corresponding to an object and white otherwise.

This procedure can be called *thresholding* and involves looking at each pixel and deciding whether it should be made white (255) or black (0). Comparing the numeric pixel value against a fixed number called a *threshold* generally makes the decision. If the pixel value is less than the threshold, the pixel is set to zero; otherwise it becomes 255. The problem to be solved in thresholding is to select a good value for the threshold.

Although it is a simple matter to convert an image that has many gray levels into one that has only two, it is much harder to do it in such a way that the important features in the image are still visible. The problem is that image quality is a subjective issue, and since there are many binary images that can be produced from the same gray-level image, which one is correct? Clearly, the one that most correctly retains the objects of interest; but this depends on the user and the application. This argument leads to the conclusion that the user ought to select the threshold, and sometimes this is in fact done (*though not in our case*).

If the gray-level histogram of an image is bimodal, meaning that it has two obvious peaks, then selecting the threshold is apparently simple: one merely has to choose the gray level representing the lowest point between the two peaks. Unfortunately, relatively few images have such an obvious bimodal appearance.

For histograms that are also bimodal there are problems involved with locating the peaks automatically. It is rare for the largest two values in the histogram to belong to separate peaks. More often they belong to the same peak, the largest one. In addition, the histogram is not a smooth curve but contains jagged sections that can easily be mistaken for peaks.

In practice, it is rare to encounter an obviously bimodal histogram, and although it is possible to force a histogram to have an arbitrary shape, doing so will alter the relationships between regions and levels, and will not always result in a good threshold. *So, what can we do apart from letting the user select the appropriate threshold?*

If the histogram is not of any assistance in threshold selection other techniques still work, and these usually involve either a *search* or some statistical measure. Search methods select a number of thresholds and accept or reject them based on some "goodness" measure. Statistical methods compute a threshold based on some set of measured properties of the image.

2.1.1. Iterative Selection

One example of a search method is called iterative selection (adapted from [Parker, 1994][1]). The idea is to provide an estimate of the average gray level of both the background (*Tb*) and

[1] [Parker, 1994] J. R. Parker, "Practical computer vision using C", E.E.U.U, John Wiley & Sons Inc, 1994.

the objects (*To*), and to use the average of these two levels as the threshold: *T=(Tb+To)/2* . Initially, these values are guesses based on known properties of the image. If it is known that objects are dark and the background is lighter, then the initial values could be *To=0* and *Tb=255*. Sometimes values from the four corners are assumed to be background pixels. It is even possible to use the overall mean gray level as the initial threshold *T* and then produce a guess for *Tb* and *To* on the next iteration, which is done in our project. In fact, the values are determined, they are initially just a guess.

The next step is to refine the values of *Tb* and *To* using the threshold *T*. Assuming that dark regions are objects, *To* is recalculated to be the mean value of all pixels less than *T*. Similarly, the new value of *Tb* is the mean value of all pixels with a value greater than *T*. This process should produce a better estimate of the mean levels, and these in turn should produce a better estimate of the threshold, which is now recomputed, as before, as *T=(Tb+To)/2*, using the new values for *Tb* and *To*.

This entire process is repeated until the same threshold value *T* is produced on two consecutive iterations, at which point *T* is presumed to be a good threshold for the image. This method has the advantage of being simple to implement and often yields good thresholds. The binarization procedure at Figure 2 has used this method.

```
Algorithm IterativeSelection(img: image): Threshold
Begin
  Tb, To, No ⇐ 0
  N ⇐ img.size
  For every pixel (i,j) in img do To ⇐ To + img[i,j]
  Tt ⇐ (To/ N)        //Initial threshold set to the overall mean gray level
  Exit ⇐ false
  While (not exit) do Begin
    Tb, To, No, Nb ⇐ 0
    For every pixel (i,j) in img do Begin
      If (img[i,j] >= Tt) Then Begin
            To ⇐ To + img[i,j]
            No ⇐ No+1
      Else
            Tb ⇐ Tb + img[i,j]
            Nb ⇐ Nb+1
       EndIf
    End For
    If (No = 0) Then No ⇐ 1
    If (Nb = 0) Then Nb ⇐ 1
    T2 ⇐ (Tb/Nb + To/No)/2
    If (T2 = Tt) Then exit ⇐ true
    Tt ⇐ T2
  End While
  Result ⇐ Tt
End
```

Figure IV-3. Iterative Selection Algorithm: Searching the appropriate binarization Threshold

Iterative selection can be understood as a binary search of all the possible gray levels for a reasonable threshold. However, there is no evaluation of the threshold for its suitability; it is simply assumed that when the procedure stops, the resulting *T* will be acceptable.

In fact, this procedure has been working for more than a year on the UJI Telerobotic Training System, under some lightning variations, and has given very good results. Sometimes, when the illumination caused some shadows on the scenes, the algorithm could take the shadows as belonging to the object. However, since these situations were improved, the binarization has been always accomplished at a very high quality level.

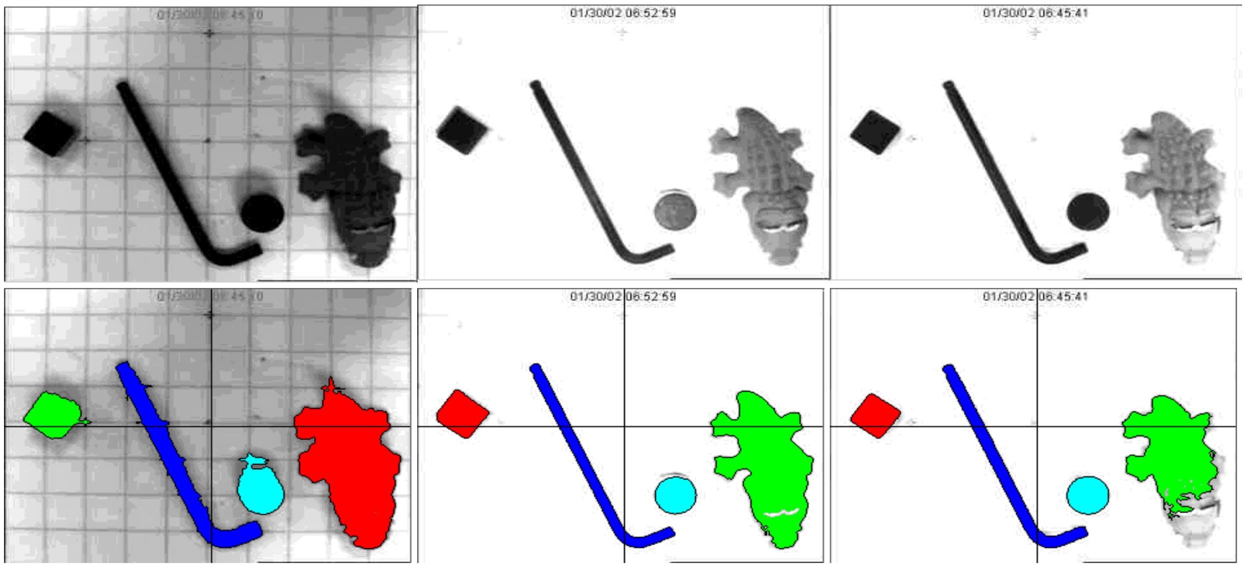


Figure IV-4. Applying the Iterative Selection Algorithm under illumination variations

At Figure 4, different levels of illumination are shown. The top row represents the camera input, and their corresponding segmentation using the Iterative Selection thresholding is at second row. As we can see, even when the image to be treated is quite dark, the segmentation works quite well. The same happens when there is an excess of light.

Although the binarization seems to be quite robust, in order to obtain a good mathematical object representation, the illumination factor should be maintained as constant as possible. Some significant lightning variations could incorporate shadows to a "*circle*" and make it appear to the object recognition algorithm as e.g. a "*cube*" (or viceversa). When very different objects are used, the problem does not occur (e.g., "*allen*" and "*dragon*"). However, for other situations (e.g. "*wheel*" and "*circle*"), the illumination control should be really taken into account.

2.2. Segmentation

Although the segmentation term is used in some literature to consider the whole image interpretation (binarization, segmentation and feature extraction), we are considering it as the process needed to identify objects and their contours from a binarized image. Once this information is extracted, the next step (feature extraction) could be performed.

Thus, three points are here treated:

1. The labeling procedure, which enables assigning a unique object identification to every pixel belonging to that object. For example, if a two-object scene is computed, every pixel of the first object would be labeled as "1". For the second object, the label "2" would be used, and finally "0" for the background.

2. In order to accomplish the labeling procedure, it is necessary defining when two contiguous pixels are or not considered as neighbors. This is commonly known as connectedness criteria.

3. Finally, in order to enable some feature extraction, it is necessary getting a representation of the external contour of an object.

2.2.1. Labeling Objects

Two points in a binary image are considered connected if a path can be found, along which the characteristic function is constant (e.g. every point in path belongs to object ***a***) (adapted from [Horn, 1993][1]).

A connected component (*scene object*) of a binary image is a maximal set of connected points, that is, a set, such that, a path can be found between any two of its points and all connected points are included.

One way to label the objects in a discrete binary image is to find a pixel *(i,j)* belonging to that object (seminal or initial point). Then we assign a label (e.g. the object number) to this point and to its neighbors. Next, label all the neighbors of these neighbors (except those that have already been labeled), and so on. When this recursive procedure stops, one object will have been labeled completely, and we can continue choosing another seminal point for the next object. To find new places to start from, we can simply scan through the image in any systematic way (e.g., from top to down and left to right), starting a labeling operation whenever an unlabeled point is found belonging to a new object (black pixel). When we have tried every cell in this scan, all the objects in the binary image will have been assigned a label.

In addition, in order to avoid image noise or isolated points being mistaken for existing objects (e.g. calibration lines or image acquisition time), the algorithm could be improved by establishing a minimum number of pixels per object. Once an object has been labeled, if the number of pixels (area) is less than a certain factor (experimentally obtained), the object is considered as background and therefore, the labeling discarded.

Algorithmically, this procedure could be expressed as follows:

[1] [Horn, 1993] B. K. P. Horn, "Robot Vision", Cambridge (E.E.U.U), McGraw-Hill, 1993.

```
Algorithm LabelingObjects(img: Binarized image): Labeled images
Begin
 CurrentLabel ⇐ Initial object identifier
 For every pixel (i,j) in img do Begin
   If (img[i,j] = 1) Then Begin //Pixel belongs to an object
      If (img[i,j] is not labeled) Then Begin //Pixel has not been labeled yet. Seminal Point
          Img[i,j] ⇐ currentLabel
          Label every connected pixel to (i,j) using currentLabel
          If (area of object currentLabel < MINIMUM_AREA) Then Begin
              Discard currentLabel labeling
          EndIf
      EndIf
   EndIf
 End For
End
```

Figure IV-5. Labeling objects procedure

2.2.2. Connectedness

At this point, we are going to define exactly what neighborhood means when applied to a digital image (bidimensional array of pixels). If we are dealing with a square tessellation, we should presumably regard the four picture cells touching a given cell on the edges as neighbors. But, what about the four cells touching on the corner? The are two possibilities:

1. Four-connectedness: Only edge-adjacent cells are considered neighbors.

2. Eight-connectedness: corner-adjacent cells are considered neighbors, too.

These alternatives are shown at Figure 6 (adapted from [Gonzalez et al., 1993][1]):

[1] [Gonzalez et al, 1993] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. EEUU: Addison-Wesley, 1993.

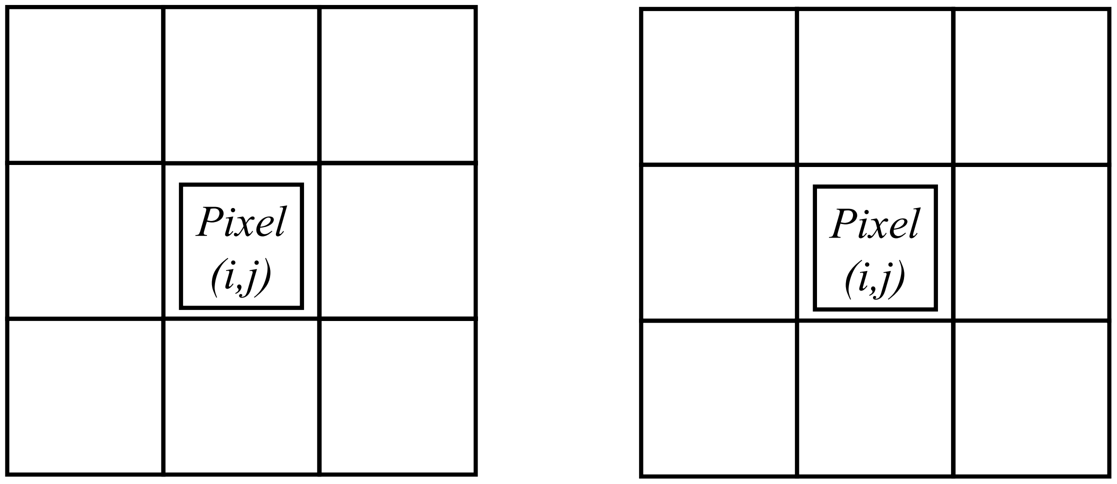


Figure IV-6. Four and Eight connected neighbors to pixel (i,j)

The four-connectedness is faster and easier to implement. On the other hand, the eight one is necessary in some given situations, for example, when obtaining the contour representation of a given object that has been labeled through the four-connectedness approach.

2.2.3. Contour representation

In order to calculate important object characteristics like, e.g. the perimeter, it is necessary to obtain a representation of the boundary. In fact, this contour information is applied later to calculate the grasping points of an object, a part from being a useful source to obtain invariant object descriptors too.

Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. Typically, this representation is based on four- or eight-connectivity of the segments. The direction of each segment is coded by using a numbering scheme such as the ones shown at Figure 7.

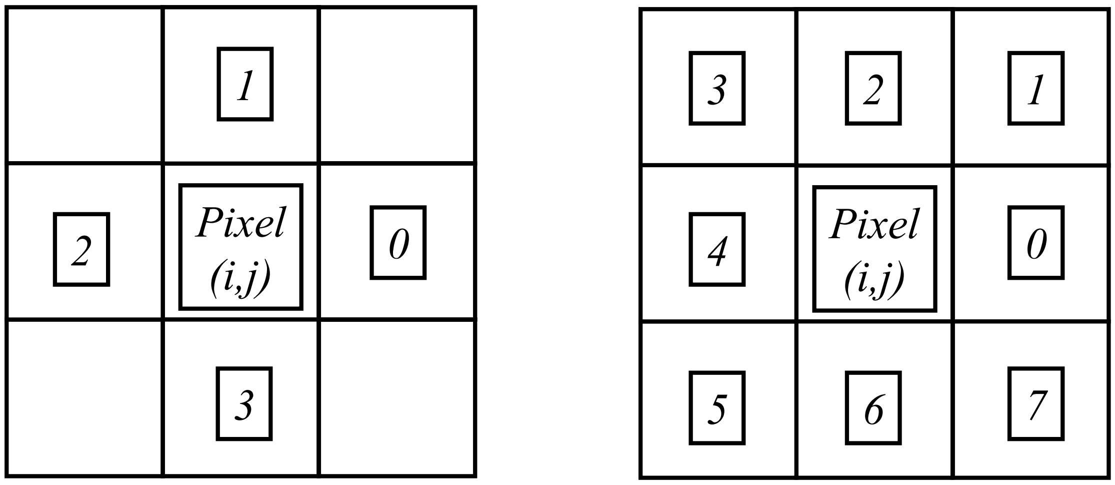


Figure IV-7. Chain code representation using Four and Eight connected neighbors to pixel (i,j)

Thus, for example, the chain code "00332211" would represent the borders for a single cube, starting from top to down and left to right (see Figure 8).

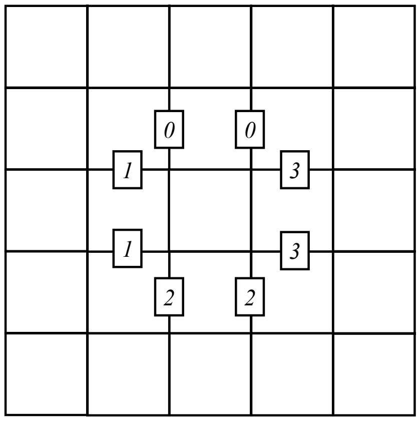


Figure IV-8. Chain code "00332211" assigned to a cube image

Obviously, the chain code of a boundary depends on the starting point. However, the code can be normalized and made invariant to orientation, as explained in [Gonzalez, 1993]. In fact, the invariant version of a chain code could be used as a object descriptor to the automatic object recognition procedure.

2.3. Feature extraction

At this point, we are going to see how the information obtained from the segmentation procedure can be used to extract some important object characteristics, which will be further used to obtain the final invariant object descriptors.

2.3.1. Moments

For a 2-D continuous function *f(x,y)*, the moment of order *(p+q)* is defined as [Gonzalez et al, 1993][1]:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y)\, dx\, dy \qquad \text{for } p,\ q = 0, 1, 2, \ldots$$

A uniqueness theorem [Papoulis, 1965][2] states that if *f(x,y)* is piecewise continuous and has nonzero values only in a finite part of the *xy* plane, moments of all orders exist and the moment sequence (m_{pq}) is uniquely determined by *f(x,y)*. Conversely, (m_{pq}) uniquely determines *f(x,y)*. The *central moments* can be expressed as

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q f(x,y)\, dx\, dy \qquad \text{for } p,\ q = 0, 1, 2, \ldots$$

where

[1] [Gonzalez et al, 1993] R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* EEUU: Addison-Wesley, 1993.

[2] [Papoulis, 1965] A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* New York: McGraw-Hill, 1965.

$$\bar{x} = \frac{m_{10}}{m_{00}} \qquad \text{and} \qquad \bar{y} = \frac{m_{01}}{m_{00}}$$

Please, note that $\bar{x}$ and $\bar{y}$ define the *Centroid* of an object, and m_{00} its *Area*.

For a digital image, the *central moments* can be expressed like:

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

The central moments of order up to 3 are

$$\mu_{10} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^0 f(x,y) = m_{10} - \frac{m_{10}}{m_{00}}(m_{00})$$

$$\mu_{11} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^1 f(x,y) = m_{11} - \frac{m_{10}m_{01}}{m_{00}}$$

$$\mu_{20} = \sum_x \sum_y (x-\bar{x})^2 (y-\bar{y})^0 f(x,y) = m_{20} - \frac{2m_{10}^2}{m_{00}} + \frac{m_{10}^2}{m_{00}} = m_{20} - \frac{m_{10}^2}{m_{00}}$$

$$\mu_{02} = \sum_x \sum_y (x-\bar{x})^0 (y-\bar{y})^2 f(x,y) = m_{02} - \frac{m_{01}^2}{m_{00}}$$

$$\mu_{30} = \sum_x \sum_y (x-\bar{x})^3 (y-\bar{y})^0 f(x,y) = m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2 m_{10}$$

$$\mu_{12} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^2 f(x,y) = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2 m_{10}$$

$$\mu_{21} = \sum_x \sum_y (x-\bar{x})^2 (y-\bar{y})^1 f(x,y) = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01}$$

$$\mu_{03} = \sum_x \sum_y (x-\bar{x})^0 (y-\bar{y})^3 f(x,y) = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2 m_{01}$$

In summary,

$$\mu_{00} = m_{00} \text{ (Area)} \qquad \mu_{11} = m_{11} - \bar{y}m_{10}$$
$$\mu_{10} = 0 \qquad \mu_{30} = m_{30} - 3\bar{x}m_{20} + 2m_{10}\bar{x}^2$$
$$\mu_{01} = 0 \qquad \mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2 m_{10}$$
$$\mu_{20} = m_{20} - \bar{x}m_{10} \qquad \mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01}$$
$$\mu_{02} = m_{02} - \bar{y}m_{01} \qquad \mu_{03} = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2 m_{01}$$

Please, note that the central moments are invariant to object translation.

The *normalized central moments*, denoted by η_{pq}, are defined as

$$\eta_{pq} = \frac{\mu_{10}}{\mu_{00}^{\gamma}}$$

where

$$\gamma = \frac{p+q}{2} + 1 \qquad \text{For } p+q = 2,3,...$$

The normalized central moments are important because they become constant when the object is scaled. They are good characteristics to define more powerful invariant object descriptors (e.g., Hu).

2.3.2. Perimeter

In a binarized image, the perimeter of a region consists of the set of pixels that belong to the object and that have at least one neighbor that belongs to the background. In other words, the perimeter of an object could be understood as the length of its boundary.

In fact, there exist many ways to calculate an approximation to the real perimeter through a digitized image. A very easy approach would be just taking the perimeter as the number of points in a contour. As we can appreciate it makes the perimeter dependent on the selected connectivity approach.

This situation is clearly expressed by the Freeman approximation:

$$L_F = n_p + \sqrt{2} \cdot n_i \qquad \textit{Freeman approximation}$$

where n_p is the number of transitions in the chain code in a four connected way, and n_i the ones in diagonal.

For those situations in which performance is the most important, some alternatives to the Freeman operator could be used. This is the case for the L_{m3} operator, which gives a very good perimeter approximation consuming three times less computing time (refer to [Proffit et al., 79][1] for further information). Please note that the L_{m3} factor is the one incorporated to the telerobotic training system implementation.

[1] [Proffit et al., 1979] D. Proffiet, D. Rossen, "Metrication errors and coding efficiency of chain-encoding schemes for the representation of lines and edges", Comp. Graph. Image Processing, 10: 318-32, 1979.

3. Object Descriptors

Basically, representing a scene object involves two choices:

1. Represent the object in terms of its external characteristics (its boundary)
2. Represent it in terms of its internal characteristics (the pixels comprising the region).

Choosing a representation scheme, however, is only part of the data useful to a computer. The next task is to describe the region based on the chosen representation. For example, a region may be represented by its boundary (e.g, perimeter), the orientation of the straight line joining the extreme points, and the number of concavities in the boundary.

Generally, an external representation is chosen when the primary focus is on shape characteristics. An internal representation is selected when the primary focus is on reflective properties, such as color and texture. In any case, the features selected as descriptors should be as insensitive as possible to variations such as changes in size, changes in location, and rotation.

In our case, the object recognition procedure is based on shape characteristics. Information such as color and texture have not been studied yet. The set of invariant descriptors that we have initially selected are listed below:

- Hu Descriptors (Surface and Borders approaches)
- Thinness ratio (*Tr*)
- Shape Elongation (*Lv*)
- Spreading (*Rv*)
- Compactness (*C*).

3.1. Hu Descriptors

The Hu descriptors are a mathematical representation of a shape that has the particularity of being invariant to *scale*, *rotation* and *traslation*. Besides this, they are quite simple to obtain.

In order to calculate the invariant HU descriptors associated with an object, it is necessary going through the *central moments* (invariant to the object location) and the *normalized central moments* (invariant to scale). Finally, we obtain the seven HU moments that are invariant to location, scale and rotation of the objects on the scene.

The set of seven invariant moments can be derived from the second and third moments [Bell, 1965][1][Hu, 1962][2]:

[1] [Bell, 1965] E. T. Bell, *Men of Mathematics*, New York: Simon and Schuster, 1965.

[2] [Hu, 1962] M. K. Hu, *Visual Pattern Recognition by Moment Invariants,* IRE Trans. Info. Theory, vol. IT-8, pp. 179-187.

$$\phi_1 = \eta_{20} + \eta_{02}$$
$$\phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2$$
$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$\begin{aligned}\phi_5 = {} & (\eta_{30} - 3\eta_{12})\,(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\ & + (3\eta_{21} - \eta_{03})\,(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]\end{aligned}$$
$$\begin{aligned}\phi_6 = {} & (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\end{aligned}$$
$$\begin{aligned}\phi_7 = {} & (3\eta_{21} - \eta_{03})\,(\eta_{30} - \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\ & + (3\eta_{12} - \eta_{30})\,(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]\end{aligned}$$

As introduced previously, this set of descriptors has the characteristic of being invariant to location, rotation, and scale changes. Moreover, in order to obtain a greater stability of the recognition procedure, fourteen Hu descriptors can be used. The idea is to compute first the Hu descriptors for the pixels belonging to the object surface. It means that *f(x,y)* is considered 1 when the pixel belongs to the surface and 0 otherwise. The rest of descriptors are obtained considering the external contour of the object. For this situation, *f(x,y)* is 1 when the pixel belongs to the border and 0 otherwise.

It is easier to understand this situation by means of an example. Just consider the following set of samples for the same object class ("*allen*") (see Table 1).

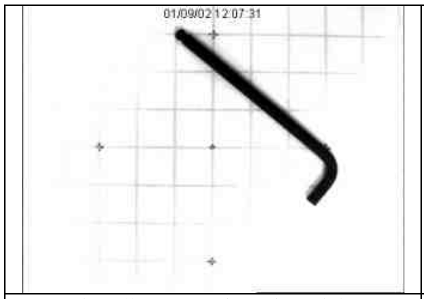
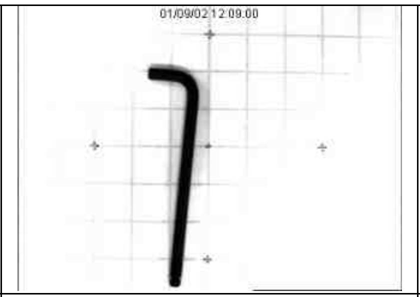
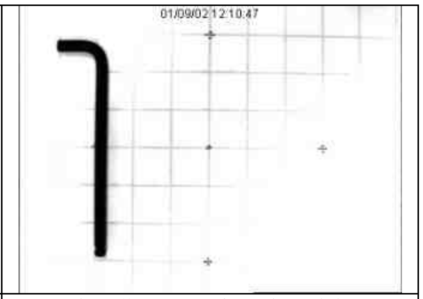
		
Sample "*allen01.jpg*" from the training set	Sample "*allen10.jpg*" from the training set	Sample "*allen19.jpg*" from the training set

Table IV-1: Samples of class "allen": Their rotation, scale and mirroring varies considerably.

The corresponding seven Hu descriptors for the surface are (reported at Table 2):

Surface	Hu1	Hu2	Hu3	Hu4	Hu5	Hu6	Hu7
Allen01	1,622449	2,794189	1,87E+07	1,90E+07	2,01E+14	1,48E+07	-3,15E+14
Allen10	1,714437	2,578057	6386389	6544710	-1,25E+13	6145301	2,50E+13
Allen19	1,661051	2,497977	1296658	1333882	1,75E+12	2085977	1,76E+12

Table IV-2: Hu descriptors using f(x,y)=1 when the pixel belongs the object's surface, 0 otherwise

On the other hand, the calculated Hu descriptors for the border (external contour) are (see Table 3):

Borders	Hu1	Hu2	Hu3	Hu4	Hu5	Hu6	Hu7
Allen01	9,907682	95,92047	1,14E+09	1,16E+09	1,33E+18	-1,51E+08	-1,33E+18
Allen10	8,627161	58,70144	54541,79	7825,882	1,65E+07	-55226,36	1,20E+08
Allen19	8,339341	59,98598	25256,68	1226,679	-4523629	-2527,641	-4113688

Table IV-3: Hu descriptors using f(x,y)=1 when the pixel belongs the object's border, 0 otherwise

At this point, we could think about the following *question: Which of the 14 Hu descriptors is going to be more useful for recognizing an object?* Obviously, those values which are the most *stable* (variance close to 0.0) through the samples of a given class (e.g. "*allen*"). Moreover, being m_{ki} the average of descriptor i for class k, if the variance between m_{ki} for every class k in the training set is high, then the classification hits would be increased enormously.

By looking at Tables 2 and 3, we can see how some of the HU descriptors present more stability than the others. This is the case for Hu1 and Hu2, whose values for a set of samples of the same object present very little variation. Obviously, this should be taken into account when applying a recognition algorithm to the samples, as we will see in the next section.

3.2. Thinness ratio Descriptor

A very useful and frequent descriptor is the measure of thinness, defined as [Duda, 1973][1]:

$$T = 4\pi\left(\frac{S}{P^2}\right)$$ where P is the perimeter and S the area of the object

A famous theorem is that T has a maximum value of 1, which is achieved if the figure in question is a circle. Analogously, from all possible triangles, the equilateral triangle has maximum T (of $T=\pi\sqrt{3}/9$), and from all quadrilaterals, the square has maximum T (of $T=\pi/4$).

Loosely speaking, then, the fatter a figure is, the greater will be the associated thinness ratio; conversely, line-like figures will have a thinness ratio close to zero. Moreover, the thinness ratio is dimensionless and hence depends only on the shape of the figure.

NOTE: The S (area) and P (perimeter) values are real measures that must be approximated with a certain parameter once captured with a camera and digitally processed. In our case, the S parameter is estimated as the number of pixels contained within the object border. Moreover, the P parameter allows multiple types of approximations, some of them explained in the

[1] [Duda, 1973] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis,* New York: John Wiley & Sons, 1973.

previous section. For our study, we have used the L_{m3} approach, which offers a very accurate approximation to the real perimeter with a very low computational cost.

3.3. Elongatedness Descriptor

The elongatedness descriptor (L_v) is derived from the calculation of the best fit ellipse, whose maximum (I_{max}) and minimum (I_{min}) axes length are defined as:

$$I_{min} = \frac{\mu_{20} + \mu_{02} - \sqrt{4\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}}{2}$$

$$I_{max} = \frac{\mu_{20} + \mu_{02} + \sqrt{4\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}}{2}$$

And then, the L_v descriptor is defined as

$$L_v = \frac{I_{min}}{I_{max}}$$

3.4. Spreadness Descriptor

As in the previous case, the spreadness descriptor (R_v) is derived from the calculation of the best fit ellipse too.

$$R_v = \frac{I_{min} + I_{max}}{m_{00}^2}$$

3.5. Compactness Descriptor

Another descriptor based on the perimeter (P) and the area (S) refers to the compactness ratio (C), defined as [Gonzalez, 1993][1]:

$$C = \frac{P^2}{S}$$ where P is the perimeter and S the area of the object

Compactness is a dimensionless quantity, and thus insensitive to scale and translation changes. With the exception of errors introduced by rotation of a digital region, compactness is also insensitive to orientation.

4. Classification Models

Up to now, we have applied the image processing algorithms to the camera input and acquired the vector $\boldsymbol{x}$ of descriptors identifying uniquely the scene object. The aim at this point is

[1] [Gonzalez et al, 1993] R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* EEUU: Addison-Wesley, 1993.

to compare this vector with an already trained set of samples (hereafter training set) in order to identify the scene object as belonging to an object class (e.g., "*allen*").

Among recognition or classification techniques, those based on a form of distance measure probably constitute the most widely known methods. The popularity of these arises in part from their extreme conceptual and implementational simplicity and also in part from the fact that they model adequately a large number of practical situations. Within this context, the Nearest Neighbor (NN) rule [Dasarathy, 1990][1] is one of the simplest non-parametric classification algorithms devised, next only to the Minimum-Distance (MD) approach.

In the telerobotic system described in this work, the object recognition module utilizes a distance-based scheme. With this end, a number of classifiers using several metrics have been tested in order to evaluate their performance when applied to a remote object recognition problem. In particular, the MD and k-NN decision rules, along with a recently proposed classification procedure, namely k-Nearest Centroid Neighbors (k-NCN) classifier, have been used.

4.1. The MD Classifier

The MD classifier is arguably the simplest non-parametric algorithm. Let X be a set of n previously labeled prototypes (namely, training set), and let m_1, m_2, ..., m_c be the means for the c problem classes.

$$m_i = \frac{1}{N_i} \sum_{x \in \omega_i} x$$

where $i = 1, 2, ..., c$ and
N_i is the number of trained samples from class ω_i

One way to determine the class membership of an unknown sample vector $\boldsymbol{x}$ is to assign it to the class of its closest prototype. Then, a new sample x is classified by measuring the distance from x to each of the c means, and assigning x to the class for which the corresponding distance is minimum:

$$\delta_{DM}(x) = \omega_i \Leftrightarrow d(x, m_i) = \min_{j=1,\dots,c} d(x, m_j)$$

where d is the distance measure selected

Algorithmically, the MD Classifier could be represented as:

[1] [Dasarathy, 1990] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, Los Alamitos, CA: IEEE Computer Society Press, 1990.

```
Algorithm MD(x: sample): Object name
Begin
  Minimum ⇐ Infinity
  Name ⇐ “”
  Calculate $m_i$ for every class $i$ in training set
  For every $m_i$ do Begin
    Distance ⇐ distance($x$, $m_i$)
    If (distance < Minimum) Then Begin
      Minimum ⇐ Distance
      Name ⇐ ClassName($m_i$)
    End If
  End For
  Result ⇐ Name
End
```

Figure IV-9. Minimum Distance Classifier represented algorithmically

In practice, the minimum distance classifier works well when the distance between means (m_i) is large compared to the spread or randomness of each class with respect to its mean. Gonzalez [Gonzalez, 1993][1] presents a demonstration that shows that the minimum distance classifier yields optimum performance (in terms of minimizing the average loss of misclassification) when the distribution of each class about its mean (m_i) is in the form of a spherical "hypercloud" in d-dimensional pattern space.

4.1.1. The MD Classifier with Reject

What happens with the MD classifier when a sample of a unknown class (e.g., "*iron*") appears in the camera image? Obviously, the algorithm will assign incorrectly the closest class into the training set (e.g., "*allen*") to the sample of the new class. It is necessary to extend the method in order to detect when an unknown object appears on the scene and then enable some kind of learning procedure in order to incorporate the new class into the training set.

In order to deal with that situation, the "*MD with Reject*" procedure is introduced, which consists of using first the previous MD algorithm and then checking if the distance between the sample and the mean of the assigned class (m_i) is less than a certain threshold (T).

[1] [Gonzalez et al, 1993] R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* EEUU: Addison-Wesley, 1993.

$$\delta_{MD}(x) = \begin{cases} \omega_i & \text{If } \; d(x, m_i) < \mathrm{T} \\ \omega_0 & \text{otherwise (rejected classification)} \end{cases}$$

where the label ω_0 is here used to identify a rejection situation (new class to be learned).

As we can appreciate above, in order to realize if the scene object represented by the vector $\boldsymbol{x}$ belongs to any of the training set classes or, by the other hand, the sample represents a new class to be learned, it is necessary to establish a reject criterion. The idea is, once the Minimum Distance classifier has selected a class as the closest one to the $\boldsymbol{x}$ vector, it must be checked if the *proximity* to the class is less than a certain value (*threshold* in Figure 10). If *true*, the sample $\boldsymbol{x}$ belongs to the Minimum Distance class, otherwise it is considered like a "*doubtful*" sample or even a new class sample. The *T* value is calculated empirically in advance, and it depends on the selected distance measures and the descriptors.

Algorithmically, the MD Classifier with Reject could be described as follows:

```
Algorithm MD-Reject($\boldsymbol{x}$: sample, Threshold:integer): Object name
Begin
  Minimum ⇐ Infinity
  Name ⇐ “”
  Calculate $m_i$ for every class $i$ in training set
  For every $m_i$ do Begin
    Distance ⇐ distance($x$, $m_i$)
    If (distance < Minimum) Then Begin
      Minimum ⇐ Distance
      Name ⇐ ClassName($m_i$)
    End If
  End For
  Proximity ⇐ Minimum
  If (Proximity < Threshold) Then
    Result ⇐ Name
  Else
    Result ⇐ ""
  End
End
```

Figure IV-10. MD classifier with Reject represented algorithmically

Using the MD-Reject implies calculating empirically the *threshold* or *proximity* factor (T), which is dependent on the relation between the means (m_i) for every class in a given training set. This calculation is closely related to the theoretical decision frontier established by the MD classifier [Sánchez, 1998][1]. In our case, the *threshold* (T) is calculated as follows:

Let d_{ij} be the distance between m_i (mean of class i) and mj (mean of class j).

$$d_{ij} = d(m_i, m_j)$$ where class i and class j belong to the TS and ($i \neq j$)

Then, we can calculate the minimum d_{ij} for every ($i \neq j$), and call it factor D.

$$D = \min_{i,j \in TS \text{ and } i \neq j} (m_i, m_j)$$

Finally, the *threshold* (T) can be calculated as:

$$T = \frac{D}{2}$$

4.2. The k-NN Classifier

NN methods have traditionally been used as an important pattern recognition tool. In its classical manifestation, given an input sample x and a training set X, the NN rule assigns any given sample to the class indicated by the label of the closest prototype in the training set. More generally, the k-NN rule maps any sample to the problem class most frequently represented among the k closest neighbors. The reader can refer to [Dasarathy, 1990][2] for a complete survey of NN techniques.

By having a set of prototypes for each class, $\wp_j = \{P_{j,i} \,/\, i = 1, \ldots, N_j\}$, the classification of a new sample x will be based on its k closest prototypes in the training set. For a given training set, $\{X, \Theta\} = \{(x_1, \theta_1), (x_2, \theta_2), \ldots, (x_N, \theta_N)\}$, the neighborhood $V_k(x)$ of a given sample x can be defined as the set of prototypes that satisfies the following conditions:

$$\begin{cases} V_k(x) \subseteq \wp \\ |V_k(x)| = k \\ \forall p \in V_k(x), \quad q \in \wp - V_k(x) \Rightarrow d(p,x) \leq d(q,x) \end{cases}$$

$$\text{where} \quad \wp = \bigcup_{i=1}^{M} \wp_i$$

Now, by defining the distance between a sample and a set of prototypes as:

[1] [Sánchez, 1998] J. S. Sánchez, "Aprendizaje y Clasificación basados en Criterios de Vecindad. Métodos Alternativos y Análisis Comparativo", Castellón (Spain), thesis dissertation, 1998.

[2] [Dasarathy, 1990] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, Los Alamitos, CA: IEEE Computer Society Press, 1990.

$$d_k(x, \wp_i) = k - |V_k(x) \cap \wp_i|$$

then, the k-NN rule can be defined as:

$$\delta_{\text{k-NN}}(x) = \omega_i \quad \Leftrightarrow \quad d(x, \wp_i) = \min_{i=1,\cdots,M} d_k(x, \wp_i)$$

In other words, this rule classifies ***x*** by assigning it to the class most frequently represented among the *k* nearest samples [Duda, 1973][1]. In practice, it is generally considered an odd number of neighbors in order to avoid possible ties among different classes.

Algorithmically the k-NN classifier could be represented as:

```
Algorithm k-NN(x: sample): Object name
Var Window_K: array [1..k] of samples
Begin
  Window_K ⇐ ∅
  Minimum ⇐ Infinity
  Name ⇐ “”
  For every sample y in training set X do Begin
    Distance ⇐ distance(x, y)
    If (window_K is not full) Then Begin
      Insert y into window_K
    Else Begin
      Find z in window_K having maximum distance
      If (Distance is less than distance of z) Then Begin
        Delete z from window_K
        Add x to window_K
      End If
    End If
  End For
  //Now we calculate the most voted class in window_K
  maxVotes ⇐ 0
  maxVotedClass ⇐ ""
  For every class ω_z in window_K do Begin
    Votes ⇐ votes of ω_z in window_K
    If (Votes > maxVotes) Then Begin
      MaxVotes ⇐ Votes
      MaxVotedClass ⇐ ω_z
    End If
  End For
  Knn-Result ⇐ MaxVotedClass
End
```

Figure IV-11. k-NN classifier represented algorithmically

[1] [Duda, 1973] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis,* New York: John Wiley & Sons, 1973.

An important property of NN methods refers to the fact that if the number of training prototypes is large enough, the error probability for the NN rule is asymptotically (that is, in the infinite sample case) at most twice that of the optimal Bayes classifier [Cover, 1967][1]. Furthermore, the asymptotic performance of the *k*-NN rule is even better than that of the simple NN (*k-NN using k=1*) and a number of interesting bounds have been derived [Devroye, 1996][2]. It is only in the limit as *n* goes to infinity that we can be assured of the nearly optimal behavior of the *k*-NN classifier.

An interesting point for this classifier is the selection of the *k* value. In our case, we will present results depending on different values of the *k* parameter, which are obtained empirically.

4.2.1. *k*-NN with Reject

In order to give more robustness and effectiveness to the *k*-NN rule, it is possible to refine the algorithm and identify the situations where a scene object belongs to an unknown class. Moreover, it is convenient to detect those samples that present a degree of uncertainty because they are very close to the decision boundary (doubtful samples). In this case, the classification will be performed when the number of votes for a given class is superior to a specific *threshold*. In other words, if the final votes are not superior to a certain majority (*qualified majority*) for any of the classes, the sample will be rejected. Obviously, one of the purposes of this strategy is to improve the *k*-NN rule, discarding the doubtful classifications.

Let *l* be a positive integer such that $\lceil k/2 \rceil < l \leq k$, then the (*k*,*l*)-Nearest Neighbours ((*k*,*l*)-NN) can be defined [Hellman, 1970] as:

$$\delta_{(k,l)-NN}(x) = \begin{cases} \omega_i & \text{if } \; |V_k(x) \cap \wp_i| \geq l, \quad i = 1,\ldots,M \\ \omega_0 & \text{otherwise} \;\; \text{(classification rejected)} \end{cases}$$

where the label ω_0 is used to identify a reject situation (maybe a new class to be learned).

Moreover, another possibility is allowing different threshold values for each of the *M* distinct classes, l_i, which is known as the (*k*,l_i)-Nearest Neighbors rule ((*k*,l_i)-NN).

$$\delta_{(k,l_i)-NN}(x) = \begin{cases} \omega_i & \text{if } \; |V_k(x) \cap \wp_i| \geq l_i, \quad i = 1,\ldots,M \\ \omega_0 & \text{otherwise} \;\; \text{(rejected classification)} \end{cases}$$

For this situation, the decision to classify a sample or not depends on the number of votes for a given class and the threshold, l_i, associated with that class.

[1] [Cover, 1967] T. M. Cover, P. F. Hart, "Nearest neighbor pattern classification", IEEE Transactions on Information Theory, vol. 13, pp. 21-27, 1967.

[2] [Devroye, 1996] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*; New York: Springer, 1996.

The second alternative to the *k*-NN with Reject rule, consists of using a unique "*absolute majority*" for the number of votes. In this case, the sample will be rejected if the votes for a given class do not overcome the votes for the other classes in a certain "*absolute threshold*".

Let $m \geq 1$ (if $m = 1$, then this rule corresponds to the *k*-NN without reject) positive integer, be the absolute threshold in the number of votes, then the (*k*,*m*)-Nearest Neighbors ((*k*,*m*)-NN) [Luk, 1986] can be defined as follows:

$$\delta_{(k,m)-NN}(x) = \begin{cases} \omega_i & \text{if } (w_i - w_j) \geq m \quad \forall j \neq i \\ & i, j = 1, \ldots, M \\ & \sum_{p=1}^{M} w_p \leq k \\ \omega_0 & \text{otherwise (rejected classification)} \end{cases}$$

where w_i, w_j refer to the number of votes for classes ω_i, ω_j among the *k* nearest neighbors.

Obviously, this absolute majority gives to the *k*-NN rule a higher degree of robustness when classifying. However, it requires using a much bigger (*k*) value in order to assure the absolute threshold is reachable for a reasonable number of situations. Otherwise, just very few samples would be recognized.

In our case, the UJI Online Robot uses the first approach, (*k*,*l*)-Nearest Neighbours ((*k*,*l*)-NN), whose algorithm is represented at Figure IV-12:

```
Algorithm k-NN-Reject(x: sample): Object name
Var Window_K: array [1..k] of samples
Begin
  Window_K ⇐ ∅
  Minimum ⇐ Infinity
  Name ⇐ “”
  For every sample y in Training Set X do Begin
    Distance ⇐ distance(x, y)
    If (window_K is not full) Then Begin
      Insert y into window_K
    Else Begin
      Find z in window_K having maximum distance
      If (Distance is less than distance of z) Then Begin
        Delete z from window_K
        Add x to window_K
      End If
    End If
  End For
//Now we calculate the Most voted class in window_K
  maxVotes ⇐ 0
  maxVotedClass ⇐ ""
  For every class $\omega_z$ in window_K do Begin
   Votes ⇐ votes of $\omega_z$ in window_K
   If (Votes > maxVotes) Then Begin
     MaxVotes ⇐ Votes
     MaxVotedClass ⇐ $\omega_z$
   End If
  End For
  Knn-Result ⇐ MaxVotedClass
  Proximity ⇐ Number of votes in window_K for Knn-Result
  If (Proximity > k/2) Then
   Result ⇐ Knn-Result
  Else
   Result ⇐ ""
  End
End
```

Figure IV-12. k-Nearest-Neighbor Classifier with Reject represented algorithmically

4.3. The k-NCN Classifier

Experience has shown that the theoretical asymptotic performance of the *k*-NN classification rules is not always possible. In practice, the number of samples available is not large enough and then, the error rates can be too far from the expected optimal behavior. In accordance to this fact, many other models have been proposed in the last few years as a way of improving the results of NN techniques on a range of practical problems.

This lack of effectiveness associated with the *k*-NN rule and, more generally, to any distance based classifiers, could be improved by taking into account some additional information when recognizing a scene object (e.g., geometry and spatial distribution of the samples for a given class, etc.) [Sánchez, 1998][1].

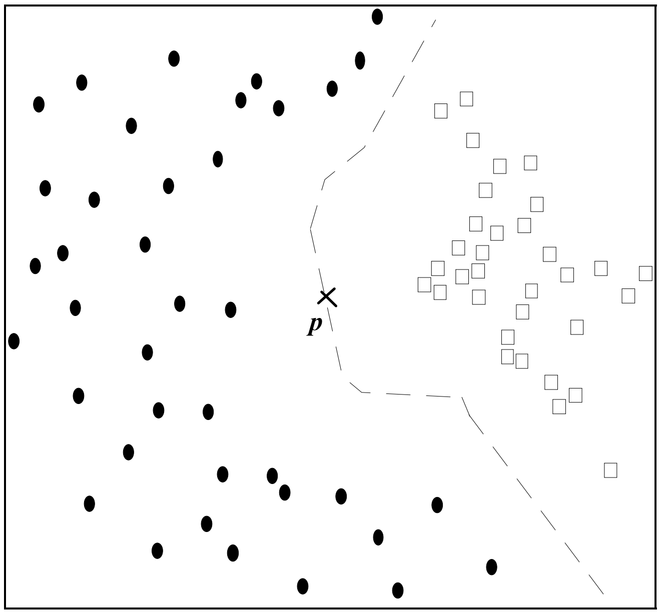


Figure IV-13. Decision frontier for two classes (circles and cubes).

In fact, around the decision boundaries, where there is always more uncertainty than in any other region in the space, the geometrical distribution of the samples can give more relevant information than the distance among them. Thus, for example, by looking at Figure 13, we can appreciate the problem of classifying sample *p* as belonging to the circles or the cubes. By considering distance-based classifications, the sample would probably be determined as belonging to the "cube" class. However, taking into account the geometrical distribution of the samples in the space, we could think that the sample belongs to the "circle" class.

By having all these situations in mind, the *k*-NCN decision rule has been defined [Sánchez, 2000][2] on the lines of complementing the *k*-NN classifiers. This scheme makes use of a neighborhood concept with two complementary constraints. First, the neighbors of a given point *p* should be as close to it as possible. Second, those neighbors should be also located as symmetrically around *p* as possible. In two steps, these NCNs can be obtained as follows [Chaudhuri, 1996][3]:

[1] [Sánchez, 1998] J. S. Sánchez, "Aprendizaje y Clasificación basados en Criterios de Vecindad. Métodos Alternativos y Análisis Comparativo", Castellón (Spain), thesis dissertation, 1998.

[2] [Sánchez, 2000] J. S. Sánchez, F. Pla, and F. J. Ferri, "Surrounding neighbourhood techniques for nearest-neighbour based classification and prototype selection", in *Recent Research Developments in Pattern Recognition*: Transworld Research Network, 2000, pp. 63-76.

[3] [Chaudhuri, 1996] B. B. Chaudhuri, "A new definition of neighborhood of a point in multi-dimensional space", *Pattern Recognition Letters*, vol. 17, pp. 11-17, 1996.

1. The first NCN of a given point p corresponds to its NN, say q_1.
2. The i-th neighbor, qi, $i \geq 2$, is such that the centroid of this and all previously selected NCNs, $q_1,..., q_{i-1}$, is the closest to p.

Algorithmically this procedure could be represented as follows:

```
ALGORITHM Finding the $k$ NCN's ($k$, $X$, $p$)
1. Initialize: $S \leftarrow X$; $T \leftarrow \varnothing$; $j \leftarrow 0$
2. Finding the NN $x_1$ of $p$ in $S$.
        $T \leftarrow T \cup \{x_1\}$; $S \leftarrow S - \{x_1\}$
3. For every $x_i \in S$:
    3.1. Calculate the centroid $c_i$ of every sample in $T \cup \{x_i\}$.
4. Select the prototype $x_i$ that minimizes the distance between $c_i$ and $p$. If two or
   more prototypes have the same distance, select the sample $x_i$ more distant to the
   neighbour selected at the last iteration.
        $T \leftarrow T \cup \{x_1\}$; $S \leftarrow S - \{x_1\}$; $j \leftarrow j + 1$
5. If $j < k$, go to Step 3.
```

Figure IV-14. Finding the k-NCN represented algorithmically

It is worth noting that this iterative procedure clearly does not minimize the distance from the input point to the centroid because it gives precedence to the individual distances instead. On the other hand, proximity of the k NCNs to the given point is guaranteed due to the incremental nature of the way in which they are obtained from the first NN.

This kind of neighborhood can be further used to define the aforementioned k-NCN classifier. Thus, the k-NCN decision rule assigns any given sample to the problem class with a majority of votes among its k NCNs.

Let us suppose a set N of prototypes belonging to M distinct clases, $\{X, \Theta\} = \{(x_1, \theta_1), (x_2, \theta_2), \ldots, (x_N, \theta_N)\}$, and let (x', θ') be the k-NN of a sample x. Let C_k be the set of centroids of grups of k prototypes formed by x' and any $k-1$ prototypes in X. Then, the neighbourhood for the NCN, $VE_k(x)$, can be defined as the set of prototypes that:

1) $VE_1(x) = \{(x', \theta')\}$

2) $VE_k(x) = VE_{k-1}(x) \cup (x_i, \theta_i) / \ d(x, c(VE_{k-1}(x), x_i)) \leq d(x, c(VE_{k-1}(x), x_j))$

$$\forall\ x_i, x_j \in X - \{VE_{k-1}(x)\},\ i \neq j$$

where $c(VE_{k-1}(x), x_i) \in C_k$ refers to the centroid between the $k - 1$ neighbours and the prototype x_i.

If we defined a new distance between the sample x and the set of prototypes in class i, $\wp_i = \{P_{i,j} / j = 1, \dots, N_i\}$, as

$$d_k(x, \wp_i) = k - |VE_k(x) \cap \wp_i|$$

we could represent the k-NCN classification rule as

$$\delta_{k\text{-NCN}}(x) = \omega_i \quad \Leftrightarrow \quad d(x, \wp_i) = \min_{i=1,\cdots,M} d_k(x, \wp_i)$$

The general idea is that the class assigned to sample x will be the most voted among the k NCNs. In practice, as it happens with k-NN, we should consider an odd number of neighbors to avoid possible ties among classes.

Please, note that the *k-NCN with reject* can be defined in the same way as explained for the k-NN rule. The idea is to consider a sample x as classified for class i as long as the number of votes for this class in the k-nearest centroids represents an *absolute majority*. A more detailed description of a number of k-NCN schemes can be found in [Sánchez, 2000][1].

5. Distance Measures

As seen in Section 3, given an object descriptor vector x={θ1, .., θn}, some of its components (θi) present mathematical properties (e.g., stability) that make them more useful in order to classify a sample. In fact, as soon as the components (θi) are more stable (variance close to 0.0) through the samples of a given class (e.g. "allen"), it means that the descriptor can be more useful when recognizing the objects.

The idea for some of the distance measures here proposed is applying some kind of weights to the descriptor components (θi) in order to favour, when calculating the distance, the columns that are more effective for the recognition process.

In this section, the different distance functions chosen for combining with the classification procedures are introduced. In particular, the distance measures here studied corresponds to Euclidean (D1), Mahalanobis (D2), normalized (D3), global extended Euclidean (D4) and per-class extended Euclidean (D5). It is to be remarked that D3, D4 and D5 measures are weighted extensions to the Euclidean distance, as further defined.

5.1. Euclidean Distance

The most significant difference among these metrics refers to the definition of the corresponding weights. Let n be the number of elements that define an object descriptor and let w_i

[1] [Sánchez, 2000] J. S. Sánchez, F. Pla, and F. J. Ferri, "Surrounding neighbourhood techniques for nearest-neighbour based classification and prototype selection", in *Recent Research Developments in Pattern Recognition*: Transworld Research Network, 2000, pp. 63-76.

denote the weight applied to an element, where *i* designates the identifier of a component in the Hu descriptor array. Then, a generic weighted Euclidean distance can be written as follows:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2}$$

Thus, the general Euclidean Distance is defined in terms of equation (1) as:

$$w_i = 1 \quad \forall_{i \in} [1..n]$$

5.2. Mahalanobis Distance

The covariance of two features (descriptors) measures their tendency to vary together, i.e., to co-vary. Where the variance is the average of the squared deviation of a feature from its mean, the covariance is the average of the products of the deviations of feature values from their means [Duda, 1973][1].

To be more precise, consider feature *i* and feature *j*. Let { *x(1,i), x(2,i), ... , x(n,i)* } be a set of *n* examples of feature *i*, and let { *x(1,j), x(2,j), ... , x(n,j)* } be a corresponding set of *n* examples of feature *j*. (That is, *x(k,i)* and *x(k,j)* are features of the same pattern) Similarly, let *m(i)* be the mean of feature *i*, and *m(j)* be the mean of feature *j*. Then, the covariance of feature *i* and feature *j* is defined by

$$c(i,j) = \{ [x(1,i) - m(i)] [x(1,j) - m(j)] + ... + [x(n,i) - m(i)] [x(n,j) - m(j)] \} / (n - 1) .$$

The covariance has several important properties:

If feature *i* and feature *j* tend to increase together, then *c(i,j)* > 0

If feature *i* tends to decrease when feature *j* increases, then *c(i,j)* < 0

If feature *i* and feature *j* are independent, then *c(i,j)* = 0 *

| *c(i,j)* | <= *s(i) s(j)*, where *s(i)* is the standard deviation of feature *i*

c(i,i) = s(i) = v(i)

Thus, the covariance *c(i,j)* is a number between - *s(i)·s(j)* and + *s(i)·s(j)* that measures the dependence between feature *i* and feature *j*, with *c(i,j)* = 0 if there is no dependence. The correspondence between the covariance and the shape of the data cluster is illustrated below.

[1] [Duda, 1973] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis,* New York: John Wiley & Sons, 1973.

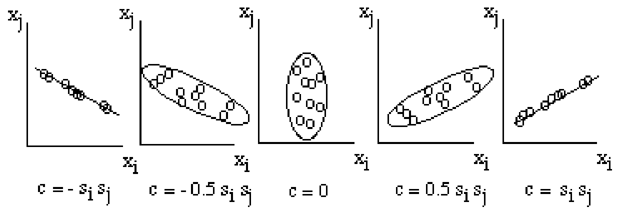


Figure IV-15. Correspondence between covariance and shape of the cluster: Samples drawn from a normal population tend to fall in a single cloud or cluster. The mean vector determines the center of the cluster, and the covariance matrix determines its shape (Adapted from [Duda, 1973]).

All of the covariances *c(i,j)* can be collected together into a covariance matrix *C*:

$$C = \begin{bmatrix} c(1,1) & c(1,2) & \dots & c(1,d) \\ c(2,1) & c(2,2) & \dots & c(2,d) \\ . & . & & . \\ . & . & & . \\ c(d,1) & c(d,2) & \dots & c(d,d) \end{bmatrix}$$

The quantity

$$r^2 = (x-\mu)^t C^{-1} (x-\mu)$$

is called the squared Mahalanobis distance (*D2*) from *x* to μ [Duda, 1973][1].

Obviously, the main problem associated with this distance is the time consumed to calculate the covariance matrix, as we will see later at the results.

5.3. Normalized Distance

The Normalized Distance (*D3*) is an extension of the euclidean distance by taking the following values for weights

$$m_{ki} = \frac{1}{N_k} \sum_{j=1}^{N_k} x_{ji}$$

where m_{ki} is the mean of feature *i* for every sample belonging class *k,* and N_k.is the number of samples in class *k*.

$$s_{ki} = \frac{1}{N_k - 1} \cdot \left(\sum_{j=1}^{N_k} x_{ji}{}^2 - m_{ki} \right)$$

where s_{ki} is the variance of the descriptor component *i* for every sample belonging to class *k*.

[1] [Duda, 1973] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis,* New York: John Wiley & Sons, 1973.

$$w_{ki} = \frac{1}{s_{ki}}$$

being *k* the class identification (e.g. "allen") and *i* the sample descriptors component (*i=1..n*, *n* equals the number of descriptors for a given object).

As we can see in the equation above, the normalized distance takes as weights the inverse of the variance for each class. It uses a different weight depending on the class we are comparing when calculating the distance. It means that we must manage and update at any time the per-class variance in order to apply this distance. And the point is: "*The weights are different for each class*" so, its cost will be higher. When a new sample is learned, the corresponding class weight must be updated accordingly.

5.4. Global Extended Euclidean Distance

The Global Extended Euclidean Distance (*D4*) function has been designed to speed up *D3*, and is based on a previous statistical analysis that defines the constant weights used by this metric [Marín et al., 2001b][1]. The weights to be applied are known in advance and it is not necessary to manage any per-class variance in order to implement the recognition procedure. Besides this, in order to perform the previous statistical study, we must know in advance the set of classes that will be entered into the system. If any sample of class is learned, the weights should be recalculated. The point is "*The weights are equal for every class, and global to the whole recognition procedure, so the cost is lower*".

$$w_i = \left(\sum_{k=1}^{N}\left(\frac{1}{s_{ki}}\right)\right)\cdot\left(\sum_{k=1}^{N}\left(\frac{1}{m_{ki}}\right)\right)$$

where *N* denotes the number of classes, while s_{ki} and m_{ki} are the variance and the mean for the component *i* of the samples belonging to class *k*, respectively.

5.5. Per-Class Extended Euclidean Distance

Finally, the Per-Class Extended Euclidean Distance (*D5*) measure (see equation below) has been designed as a way of including the *D4* capabilities in a system where classes are not well known a priori [Marín et al., 2001b]. Thus, it defines different weights w_{ki} for each class *k* and then, it takes into account the scaling properties of each Hu descriptor as the D4 does.

$$w_{ki} = \frac{1}{s_{ki}} \cdot \frac{1}{m_{ki}}$$

[1] [Marín et al., 2001b] R. Marín, P.J. Sanz, J.S. Sanchez, " *Design and Evaluation of a Telerobotic System with Object Recognition Capabilities* ", In proceedings of the *IASTED International Conference on Robotics and Applications (RA 2001), November 19-22, 2001 Tampa, Florida, USA.*

6. An Incremental Learning Algorithm

The main goal of the present work is to make the object recognition module of the telerobotic system as automatic as possible, meaning that the system has to benefit from the experience obtained when working in the recognition of new examples [Thrun, 1995][1]. This incremental learning capability provides some nice advantages (see Figure 16): first, the recognition module will be more robust because errors in the training set can be corrected during operation and second, it enables the system to adapt to partially-known or dynamic environments. As a consequence, it is expected that the performance will gradually improve over the lifetime of the telerobotic system [Marín et al., 2002b][2].

On the other hand, taking into account that our telerobotic system operates with a web-based interface, an additional problem may be created if anonymous users add erroneous knowledge to the system, which might strongly degrade the performance of the recognition module. Accordingly, the incremental learning procedure implemented in our system has also the power of overcoming this difficulty.

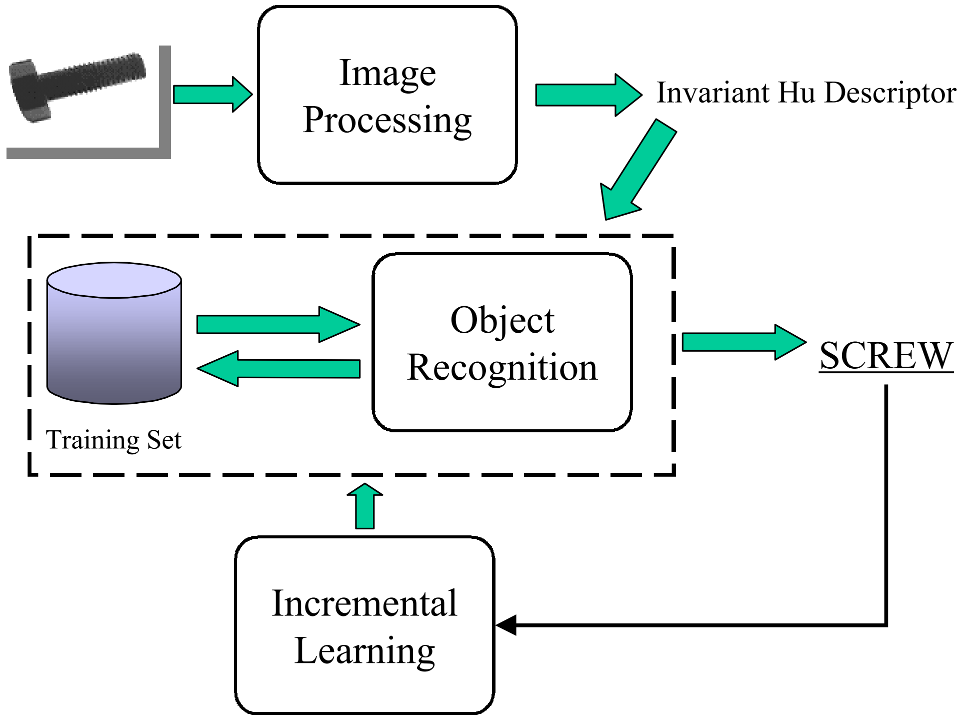


Figure IV-16. Incremental Learning structure: Increasing knowledge of object recognition module

The incremental learning algorithm proposed here covers a broad range of situations: objects belonging to new classes, mislabeled examples, atypical cases and noisy data. Our method

[1] [Thrun, 1995] S. Thrun and T. Mitchell, "Learning one more thing", in *Proc. of the 14th International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, 1995.

[2] [Marín et al., 2002b] R. Marín, J.S. Sanchez, P.J. Sanz. Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System. In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

is based on distance-based classification rules and some related techniques. In summary, the procedure consists of the following steps:

1. Let *X* be the initial training set, which is used by the object recognition module to classify new examples.
2. After recognizing a number of new examples, the system temporarily stops the object recognition module. As new examples are identified, they are stored in a set of candidate examples *Y*.
3. The system begins the learning process, by checking the correctness of examples in *Y* prior to transferring them to *X*.
   3.1. The candidate examples in *Y* are checked by a classifier with a reject option (e.g., the *k*-NCN rule with reject) using the training set *X*: new object classes and mislabeled samples are detected.
   3.2. Insert into X the correct labeled samples and the apparently incorrect ones that have been introduced by a trusted user (e.g., administrator)
   3.3. The Wilson's editing algorithm is employed as a second refinement to remove some of the examples moved from *Y* to *X* during the previous stage.
   3.4. If the number of examples in the current training set *X* is too large, the Hart's condensing algorithm is now applied so that the training set size is reduced.

Go to Step 2.

Figure IV-17. Incremental Learning procedure: Increasing knowledge of object recognition module

As can be seen, our approach to incremental learning focuses on improving the quality of training data, by identifying and eliminating mislabeled examples prior to joining the current training set *X* and the candidate set *Y*. This is accomplished by applying two filters to the set of training examples: a classifier with a reject option and an editing procedure. The former aims at detecting new object classes and also mislabeled examples, whereas the latter deals with the problem of atypical instances. As we can appreciate in Step 3.2, there are situations where samples are rejected in Step 3.1 and must get into the training set *X* because they represent a value information to the correct classification. It normally happens when establishing the initial training set before launching the first incremental learning procedure. For this situations, it is necessary to check if a trusted user has introduced the learning item or not. If true, the sample is added to *X*, otherwise it is discarded.

Distinct reject options have been implemented in many classification models [Devijver, 1982][1] as a way of reducing the misclassification rate of the system. In the specific case of our telerobotic system, the *k*-NN rule with the reject option has been defined as follows: if there is a majority of neighbors belonging to some object class, then the candidate example is assigned to that class and incorporated to *X*. Otherwise, the example is added to the set *X* with its original class label. Note that our reject option does not have the aim of eliminating examples, but finding new classes. Thus, the first filter of our learning algorithm is for detecting mislabeled examples and identifying new possible object classes.

[1] [Devijver, 1982] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall: Englewood Cliffs, NJ, 1982.

The editing stage applied here corresponds to Wilson's editing [Wilson, 1972][1], which consists of applying the *k*-NN classifier to estimate the class label of all examples in the training set and discard those whose class label does not agree with the class associated with the largest number of the *k* neighbors. In such a way, the learning algorithm is now detecting the atypical cases probably incorporated to the training set during the previous step.

Finally, the condensing procedure [Hart, 1968][2] can be applied as a way of reducing the current training set size. In brief, it consists of eliminating from the training set X those examples that are not necessary for correctly classifying the remaining instances. This is a crucial stage in our incremental learning algorithm because the usually vast amount of training examples can prohibit its usage in all but the simplest of telerobotic domains.

6.1. Wilson Editing Algorithm

This corresponds to the first proposal for editing the NN rule. Simply stated, it consists of applying the *k*-NN classifier to estimate the class label of all prototypes in the training set and discarding those samples whose class label does not agree with the class associated with the largest number of the *k* neighbors.

Thus, the Wilson's editing procedure can be written as follows: being $\{X, \Theta\} = \{(x_1, \theta_1), (x_2, \theta_2), \ldots, (x_N, \theta_N)\}$ a training set with N prototypes and M possible classes, and k the number of nearest neighbors to determine for each prototype, the Wilson's algorithm can be expressed in the following way:

```
Algorithm Wilson's Editing ($X$, $k$)
1. Initialization: $S \leftarrow X$
2. For each prototype $x_i \in X$:
    2.1. Find the $k$-NN of $x_i$ in $X - \{x_i\}$.
    2.2. If $\delta_{k\text{-NN}}(x_i) \neq \theta_i$, do $S \leftarrow S - \{x_i\}$.
```

Figure IV-18. Wilson editing algorithm

As we can see, this method is very simple to implement and comprehend. However, the computational cost associated is $O(N^2)$, which may originate performance problems when using large training sets.

[1] [Wilson, 1997] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions", *Journal of Artificial Intelligence Research,* vol. 6, pp. 1-34, 1997.

[2] [Hart, 1968] P. E. Hart, "The condensed nearest neighbor rule", *IEEE Trans. on Information Theory*, vol. 14, pp. 515-516, 1968.

6.2. Hart Condensing Algorithm

The Hart's method [Hart, 1968] was the first condensing approach for the *k*-NN rule appeared to be published. At this point, it is necessary to define the "*consistency*" term: a prototype set *S* is consistent with respect to another set *X* if, when using *S* as design set, *X* can be classified correctly. Thus, a condensed set must be both, reduced and consistent.

```
Algorithm Hart's Condensing ($X$)
1. Inicialization: $S \leftarrow \varnothing$
2. Repeat until no more prototypes are eliminated or $X$ is empty:
    2.1. For every prototype $x_i \in X$:
        2.1.1. Find the NN of $x_i$ in $S$.
        2.1.2. If $\delta_{NN}(x_i) \neq \theta_i$, eliminate $x_i$ from $X$ and assign it to $S$.
```

Figure IV-19. Hart condensing algorithm

As we can appreciate, the Hart's condensing method eliminates from the training set those prototypes that are not necessary for the correct classification of the rest of the samples by using the NN ($k = 1$) rule. The idea behind it, is the following: a sample is classified incorrectly when it is close to the decision boundary, and then it must not be discarded from the training set.

The algorithm is simple and performs rapidly, which means it is possible to obtain a proper condensed set in a few iterations. Moreover, it is important to recall that the size of the condensed set results, in many cases, considerably smaller compared to the original, provided that this has been previously edited in order to avoid the overlapping among distinct class regions.

7. Experimentation

In this section, a comparative analysis of the classification algorithms described earlier using the metrics proposed in Section V is presented. This study focuses on finding the object recognition scheme that provides the highest overall performance, that is, efficiency and recognition accuracy. Moreover, the idea is to select the best combination of object descriptors (Hu, T_r, etc) that will be used in order to facilitate the classification process.

Basically, the main objective of the present study is to select the best combination of distance, recognition algorithm, *k* value, and feature vector that accomplishes the process as fast as possible and with a high quality, in order to make it useful through the web. *Please, note that the experiments have been reproduced for every combination of classifier algorithm, distance function, k value, and feature vector. Results here shown are the most important conclusions from these values.*

The idea of using the system through the Internet means that computing time must be taken into account, so it means we must void using images that have a very high resolution. This affects enormously to the recognition process, due to the inclusion of errors while digitalizing the real image. In our case, the image is of (320x240) pixels, in gray scale, which suppose an average of 4Kb size.

7.1. Experiment I: Well differentiated objects under ideal conditions

The first experiment consists of selecting a set of well differentiated objects and evaluate the conduct of the whole set of classifiers and distances by using simply the 14 Hu descriptors (surface and borders) [Marín et al., 2001b][1]. The question we are trying to answer is the following: *Are the Hu features qualified enough to classify well-differentiated objects? Under which conditions?*

The experiment has been carried out over a database generated from 120 images corresponding to six different objects (20 samples per class), which determine the six problem classes. The different samples for a given class are obtained by rotating, scaling and moving the same object in the scene. As can be seen in Figure 20, the images are almost binarized in order to avoid as many object segmentation errors as possible. It represents the ideal situation for a robotic system.

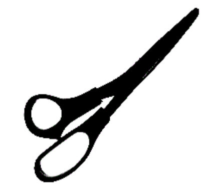
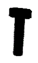
					
Allen Key	Screwdriver	Tweezers	Pliers	Scissors	Screw

Figure IV-20. Well-differentiated objects for Experiment I

In order to acquire more realistic results, the whole set of 120 samples is divided over 2 randomly generated partitions. Thus, the experiment has been applied in two steps:

1. First, the database is trained using the first partition (training set). Then, the whole set of algorithms is applied by using the second partition as test set. Computing time and error rate are obtained (R_1).

2. Secondly, we trained the system using the second partition (training set), and tested it by means of the first partition (test set). Corresponding results (R_2) are also obtained.

Finally, the averaged computing time and error rates are reported in the present section, which represents a more realistic situation. Moreover, note that none of the samples belonging to the training set belongs to the test set, which would obviously suppose a extremely well rated object recognition system.

From each trial, computing time and error rate are calculated. The former gives a direct measure of the computational cost associated with each alternative. On the other hand, the error rate provides a check on the ability of the algorithms to accurately recognize objects. In order to assess the performance relative to these two competing goals simultaneously, a combined performance measure in terms of the normalized Euclidean distance between each (time, error) pair and the origin (0 sec, 0% error) has also been defined.

[1] [Marín et al., 2001b] R. Marín, P.J. Sanz, J.S. Sanchez, " Design and Evaluation of a Telerobotic System with Object Recognition Capabilities ", In proceedings of the IASTED International Conference on Robotics and Applications (RA 2001), November 19-22, 2001 Tampa, Florida, USA.

For the present experiments, different typical settings of the parameter k, ranging from 1 through 20, have been tested and the one leading to the highest performance has finally been included in this section. In particular, the results here provided correspond to those achieved with k = 10.

As can be seen from the results reported in Table 4, D4 metric (Global Extended Euclidean distance) combined with MD (Minimum Distance Classifier) and k-NCN algorithms gives the lowest error rates, close to D4 used with k-NN rule. In fact, only these particular combinations seem to yield high performance in terms of recognition accuracy. Nevertheless, examining the other factor of interest, namely computing time, the results show that the k-NCN approach is much more computationally intense than MD and k-NN classifiers due to its O(kn) expected complexity [Chaudhuri, 1996]1 to search for the k neighbors of an example in a given set of n points.

Please note that the computing time per sample (Table 4) represents the milliseconds spent classifying a simple scene object, using a Pentium 400Hz and 128Mb RAM, and a remote database running on the server computer which is situated on the same campus as the client. All of the operations involved in the experiment are taking place at the client site over the remote database. As we can appreciate over Table 4, almost all results take less than a second, which is very convenient for our purpose.

Metric + Classifier	Error rate (%)	Computing time Per Sample (msec)
D1 + MD	58.00	317
D3 + MD	46.67	322
D4 + MD	*9.00*	316
D5 + MD	43.66	420
D1 + *k*-NN *(k=10)*	47.67	529
D3 + *k*-NN *(k=10)*	41.67	565
D4 + *k*-NN *(k=10)*	11.33	528
D5 + *k*-NN *(k=10)*	41.67	864
D1 + *k*-NCN *(k=10)*	43.33	1105
D4 + *k*-NCN *(k=10)*	*9.00*	1125

Table 4. Error Rate and Computing Time for Experiment I

Note that D3 and D5 distance measures have not been applied to the *k*-NCN procedure because this approach requires a more exhaustive analysis in order to select the object class representing the actual centroid. Analogously, the performance (that is, error rate and computing time) corresponding to D2 metric has not been included in this section because of the very high computing time provided in all trials.

1 [Chaudhuri, 1996] B. B. Chaudhuri, "A new definition of neighborhood of a point in multi-dimensional space", *Pattern Recognition Letters*, vol. 17, pp. 11-17, 1996.

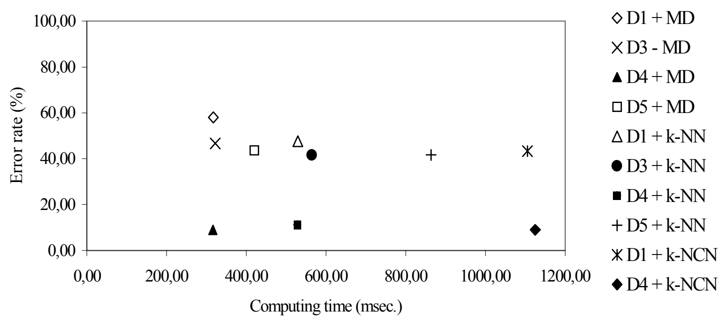


Figure IV-21. Computing time versus Error rate results for experiment I

Exploring both issues of run-time performance and recognition accuracy jointly, the results presented in Figure 21 show that the best alternatives correspond to the use of the D4 measure with MD and *k*-NN models, that is, the closest combinations to the origin (0 sec, 0% error). On the other hand, as is to be expected from the algorithm described in Section III.C, the results for the *k*-NCN classification scheme suffer from the large computational cost for calculating the successive centroids; they obtain a very low error rate (for example, employment of D4 with *k*-NCN provides an error rate of 9%), but also consumes a considerable amount of computing time (about 112 sec). Obviously, this can constitute an important drawback for the practical application of this recognition technique in a web-based environment.

7.2. Experiment II: Not well differentiated objects under normal conditions

In a web-based application with learning capabilities, the usual situation is having many kinds of objects, some of which present many similarities and therefore, are quite hard to classify (e.g., *circle* and *wheel*). This is due to the fact that their decision regions are very close and sometimes even overlapping.

Besides this, the kind of objects that we have selected are more or less easy to segment, due to the fact that they present an almost black superior surface, and are located over a white sheet. Moreover, the lightning is somehow controlled by means of a local lamp that illuminates the robot environment when somebody has its control through the web. However, depending on the time of the day (sunlight coming through the window) and whether people are working or not at the laboratory (lab lights switched on/off), the overall lightning varies accordingly. It means that the object segmentation procedure is affected by this situation, originating some little variations on the real object shape, and therefore over the final automatic object recognition result.

The idea of this experiment is to find the best combination of classifier, distance, feature vector, and *k* value that offers a robust and convenient solution for this situation.

The questions we are trying to answer by performing this experiment are the following: *Are the Hu features robust enough to be applied for not well-differentiated objects under lightning variations? If not, which other features would be useful? Under which circumstances?*

Figure 22 shows samples of the 4 classes used for the present experiment (*Allen* key, *Circle*, *Cube* and Lego *Wheel*). As can be appreciated, the objects are almost binarized thanks to its black superior shape. However, under some lighting situations, small shadows could be understood as belonging to the object and cause, for example, a *cube* to be recognized as a *circle* (it has really happened!). Besides this, the classes *circle* and *wheel* present almost the same shape, which is quite a big challenge (as we will see later) for the Hu descriptors. Note: the number of samples used per class has been twenty.

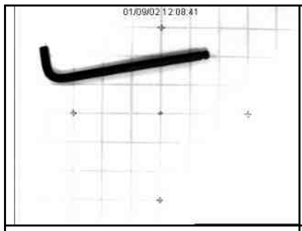
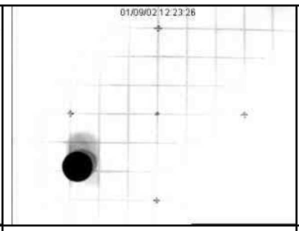
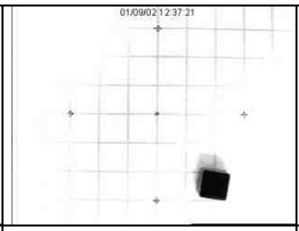
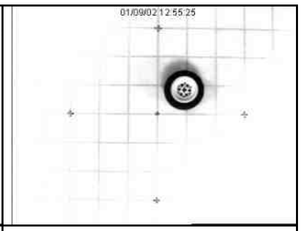
			
Allen Key	Circle	Cube	Lego Wheel

Figure IV-22. Not well differentiated objects' samples

Results for Experiment II are reported in Table 5:

Metric + Classifier	Error rate (%)	Computing time Per Sample (msec)
D1 + MD	62.5	351
D3 + MD	28.75	372
D4 + MD	47.5	350
D5 + MD	50	362
D1 + *k*-NN *(k=1)*	31.25	461
D3 + *k*-NN *(k=1)*	23.75	554
D4 + *k*-NN *(k=1,4)*	*21.25*	451
D5 + *k*-NN *(k=1)*	38.75	702
D1 + *k*-NCN *(k=1)*	31.25	1110
D4 + *k*-NCN *(k=1)*	*21.25*	1052

Table IV-5. Error Rate and Computing Time for Experiment II: 14 Hu descriptors

From Table 5, we can see that computing time has been maintained at the same rates (less than a second for most of the cases). However, the error rates obtained are almost unacceptable.

At Figure 23, we can appreciate the best results for "D4-*k*-NCN" and "D4-*k*-NN", and they still present a high error rate value (21.25%).

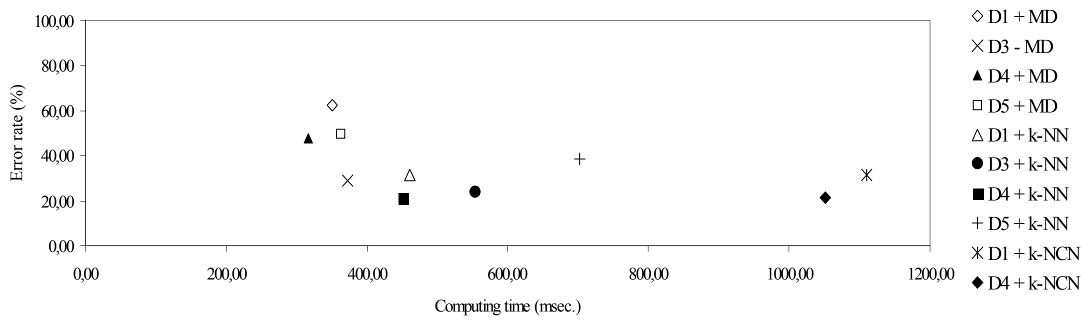


Figure IV-23. Computing time versus Error rate results for experiment I: 14 Hu descriptors

The conclusions for these results are that the Hu descriptors are not good enough as a general way to model a scene object. They work well on ideal situations, where illumination problems do not come up, or when the classes give very distant decision regions.

A possible solution to this problem consists of complementing (or replacing) the Hu descriptors information with other features, such as thinness ratio (Tr), elongatedness ratio (Lv), and spreadness ratio (Rv). The compactness ratio (C) presents almost the same response as the thinness ratio, due to the fact that they depend on the same variables (Surface and Perimeter), so it has been discarded from the final results reported here.

Moreover, it must be considered that the best fit ellipses for the "circle" and the "cube" are identical, which means that features Lv and Rv do not give information to distinguish among them. Besides this, The relations between Area and Perimeter for classes "allen" and "wheel" have some similarities as well, so for this case, the Tr feature needs to be complemented by Lv and Rv in order to perform properly.

The experiment has consisted of applying the test for every possible combination of descriptors, every distance, algorithms, and different values of k. Finally, the combination of descriptors with the highest number of hits corresponds to Tr, Lv and Rv, which means discarding the Hu values. The results for these features are reported in Table 6:

Metric + Classifier	Error rate (%)	Computing time Per Sample (msec)
D1 + MD	15	303
D3 + MD	53.8	372
D4 + MD	18.7	321
D5 + MD	33,7	310
D1 + *k*-NN *(k=1)*	*13.7*	456
D3 + *k*-NN *(k=1)*	28.7	560
D4 + *k*-NN *(k=1)*	20	421
D5 + *k*-NN *(k=1)*	33.75	521
D1 + *k*-NCN *(k=1)*	*13.7*	803
D4 + *k*-NCN *(k=1)*	32.5	915

Table IV-6: Error Rate and Computing Time for Experiment II: Tr, Lv and Rv

As we can appreciate from Table 6, the D1 function with *k*-NN and *k*-NCN algorithms present the best results. In this situation, the weighted approaches of the distance measures do not give any additional improvement.

It must be noted that using the 14 Hu descriptors with the Tr feature has given a 85% of hits (15% error rate) by using the k-NN and k-NCN algorithms with distance D4. From this, we could say the Hu descriptors can be useful to complement some features information, and should never be used as the only way to identify objects.

7.3. Experiment III: Not well differentiated objects under normal conditions using 50 samples per class

The next step consists of experimenting if the number of samples trained for a given class affects or not its the recognition capabilities and the computing time response. The questions we are trying to answer are the following: *Is the recognition algorithm more effective when using 50 samples per class in the training set? If so, how is the computing time affected?*

Again, the experiment has consisted of applying the test for every possible combination of descriptors, every distance, algorithms, and different values of k. Finally, the combination of descriptors with the highest number of hits corresponds to Tr, Lv and Rv (same results as before), which means discarding the Hu values. The results for these features are reported in Table 7:

Metric + Classifier	Error rate (%)	Computing time Per Sample (msec)
D1 + MD	13	321
D3 + MD	30	324
D4 + MD	22	336
D5 + MD	45	341
D1 + k-NN (k=18)	*7*	*814*
D3 + *k*-NN *(k=2)*	22.5	860
D4 + *k*-NN *(k=2)*	12.5	1085
D5 + *k*-NN *(k=1)*	25.5	1232
D1 + k-NCN (k=1)	*11.5*	*1413*
D4 + *k*-NCN *(k=1)*	22.5	1428

Table IV-7: Error Rate and Computing Time for Experiment III: Tr and Lv

As we can appreciate from Table 7, the D1 function with *k*-NN and *k*-NCN algorithms present the best results. Using 10 samples per class in the training set (see experiment II) resulted over in a 13% error rate, on the other hand, using 50 samples the error rate becomes 7%, which is perfectly acceptable for our purposes.

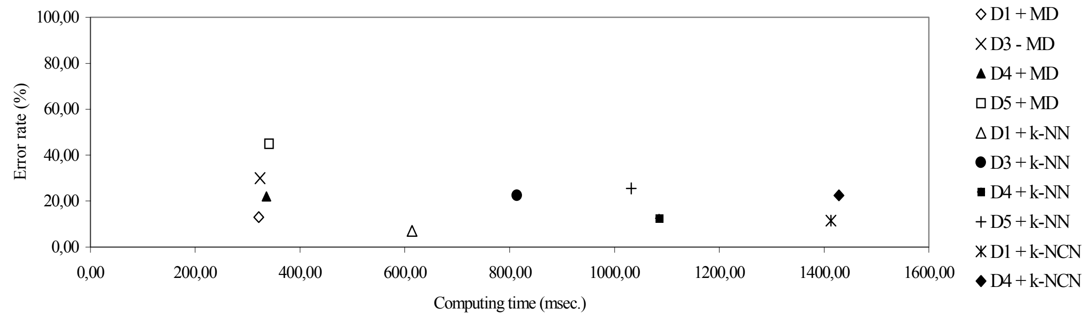


Figure IV-24. Computing time versus Error rate results for experiment III: Tr, Lv and Rv

The computing time is obviously affected when using the *k*-NN and *k*-NCN algorithms, due to the fact that they must compare the given object with every sample existing in the training set. In fact, the *k*-NN classifier offers an average recognition time of less than a second, which for our purposes can be considered good enough.

8.Conclusions and Further Work

This chapter describes the application of pattern recognition and machine learning techniques to a web-based telerobotic system. Accordingly, the focus of the present paper is two-fold: improving the performance of the object recognition module by means of different classification techniques, and proposing an incremental learning algorithm that continually increases the knowledge of the telerobotic system.

With respect to the recognition task, a number of conventional and novel distance-based classification schemes, along with several metrics have been tested in the experimental section, searching for the one with lowest computing time and highest accuracy. From the set of experiments carried out in Section 7, some preliminary conclusions can be drawn. Firstly, the *k*-NCN classification algorithm generally achieves high recognition accuracy but is computationally too expensive. It is to be admitted that the use of this classifier without an algorithmic improvement is for all practical purposes of little value, specially in these specific kinds of applications. Nevertheless, it seems that the performance of this model is more consistent than that of the MD and *k*-NN classification techniques.

On the other hand, the combination of D4 (that is, the extended Euclidean distance) with the MD classifier generally produces the best results in terms of balancing computing time for implementation purposes with classification accuracy.

The incremental learning algorithm proposed here, makes use of multiple filters in order to deal with different situations. Firstly, the *k*-NN classifier with a reject option is employed as a way of identifying new object classes and also detecting mislabeled examples. Secondly, the editing scheme eliminates atypical samples. Finally, the usage of a condensing procedure allows the training set size to be reduced, in order to decrease complexity relative to handling a large number of examples in the training set.

Future plans include investigation of other families of instance-based classification models and also other metrics [Wilson, 1997][1] in order to achieve even better performance in terms of a well balanced trade-off between run-time and object recognition accuracy. In fact, we have been considering the possibility of designing a simplified Mahalanobis distance (*Global Covarianze Extended Mahalanobis distance*) that gives very good results at an acceptable cost. The idea is to take into account the dispersion of a feature among classes (as well as its stability) in order to define the corresponding weights. A second direction in our further work focuses on including new learning capabilities to the telerobotic system. Within this context, the employment of some unsupervised or clustering techniques could help to reliably detect new object classes among the candidate examples. Thirdly, we think the recognition accuracy could be greatly improved by using some kind of shape descriptor that considers the whole border information. In fact, we are trying to use some important information already extracted to find the best grasping points of an object (e.g., *k*-torsion vector) in order to find other features that will produce better results.

[1] [Wilson, 1997] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions", *Journal of Artificial Intelligence Research,* vol. 6, pp. 1-34, 1997.

Other improvements are focused on making the computer vision procedure even better. For example, as we previously saw, for those situations in which two objects in the scene are very different in color, the proper segmentation is not guaranteed, because of using a global threshold that is applied to the whole image. The idea is to find a computationaly cheap algorithm that uses a better approach, like for example region segmentation or similar. Secondly, it would be interesting to adapt the vision algorithms to be used with more general situations like scenes with irregular background, white objects over dark surfaces, different illumination intensity in the scene, shadows originated (for example) by a person that is looking at the robot, etc.

References

[Bell, 1965] E. T. Bell, *Men of Mathematics*, New York: Simon and Schuster, 1965.

[Chaudhuri, 1996] B. B. Chaudhuri, "A new definition of neighborhood of a point in multi-dimensional space", *Pattern Recognition Letters*, vol. 17, pp. 11-17, 1996.

[Cover, 1967] T. M. Cover, P. F. Hart, "Nearest neighbor pattern classification", IEEE Transactions on Information Theory, vol. 13, pp. 21-27, 1967.

[Dasarathy, 1990] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, Los Alamitos, CA: IEEE Computer Society Press, 1990.

[Devijver, 1982] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall: Englewood Cliffs, NJ, 1982.

[Devroye, 1996] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*; New York: Springer, 1996.

[Duda, 1973] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis,* New York: John Wiley & Sons, 1973.

[Goldberg et al., 2000] K. Goldberg, S. Gentner, C. Sutter, and J. Wiegley, "The Mercury project: a feasibility study for Internet robots: design and architecture of a public teleoperated system accessible by anyone on the Internet at any time", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Gonzalez et al, 1993] R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* EEUU: Addison-Wesley, 1993.

[Hart, 1968] P. E. Hart, "The condensed nearest neighbor rule", *IEEE Trans. on Information Theory*, vol. 14, pp. 515-516, 1968.

[Horn, 1993] B. K. P. Horn, "Robot Vision", Cambridge (E.E.U.U), McGraw-Hill, 1993.

[Hu, 1962] M. K. Hu, *Visual Pattern Recognition by Moment Invariants,* IRE Trans. Info. Theory, vol. IT-8, pp. 179-187.

[Marín et al., 2001] R. Marín, P.J. Sanz, *Telerobotic Training System through the web: Low Level Architecture,* Telematics Applications in Automation and Robotics, Oxford: Elsevier Science Ltd, 2001.

[Marín et al., 2001b] R. Marín, P.J. Sanz, J.S. Sanchez, " Design and Evaluation of a Telerobotic System with Object Recognition Capabilities ", In proceedings of the IASTED International Conference on Robotics and Applications (RA 2001), November 19-22, 2001 Tampa, Florida, USA.

[Marín et al., 2002] R. Marín, P.J. Sanz., J.S. Sanchez, A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality. In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Marín et al., 2002b] R. Marín, J.S. Sanchez, P.J. Sanz. Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System. In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Natarajan, 1991] B. Natarajan, *Machine Learning: A Theoretical Approach*, Morgan Kaufmann, 1991.

[Papoulis, 1965] A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* New York: McGraw-Hill, 1965.

[Parker, 1994] J. R. Parker, "Practical computer vision using C", E.E.U.U, John Wiley & Sons Inc, 1994.

[Proffit et al., 1979] D. Proffiet, D. Rossen, "Metrication errors and coding efficiency of chain-encoding schemes for the representation of lines and edges", Comp. Graph. Image Processing, 10: 318-32, 1979.

[Sánchez, 1998] J. S. Sánchez, "Aprendizaje y Clasificación basados en Criterios de Vecindad. Métodos Alternativos y Análisis Comparativo", Castellón (Spain), thesis dissertation, 1998.

[Sánchez, 2000] J. S. Sánchez, F. Pla, and F. J. Ferri, "Surrounding neighbourhood techniques for nearest-neighbour based classification and prototype selection", in *Recent Research Developments in Pattern Recognition*: Transworld Research Network, 2000, pp. 63-76.

[Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Thrun, 1995] S. Thrun and T. Mitchell, "Learning one more thing", in *Proc. of the 14th International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, 1995.

[Wilson, 1972] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data sets", *IEEE Trans. on Systems, Man and Cybernetics,* vol. 2, pp. 408-421, 1972.

[Wilson, 1997] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions", *Journal of Artificial Intelligence Research,* vol. 6, pp. 1-34, 1997.

Chapter V AUTONOMOUS GRASPING

A part from object recognition, in order to perform a proper response to high level commands (e.g., "Grasp pincers") it is necessary to calculate the correct grasping points that enable the robot to pick up an object and to manipulate it without dropping it. The idea is to know in advance, from the superior 2D projection of an object, the set of possible gripper positions that assure that a stable grasping operation will be accomplished.

First of all, we will present the geometrical interpretation of a planar grasping, which gives an idea of the mathematical constraints that assure a manipulation complies with the stability requirements.

Secondly, the Grasp Determination Algorithm is shown, selecting first the grasping point candidates, then applying a supervisory mechanism to check the manipulation conditions in terms of the object centroid, and finally assuring some safety conditions such as collisions.

Thirdly, the chapter shows the way an operator can select different grasping alternatives to manipulate a given object by means of the user interface. By default, the most stable grasping alternative is executed.

And finally, some experimental results are presented that summarize the computing time invested in calculating the n possible grasping points for a given object.

Please note the application of this Grasp Determination Algorithm is a novel contribution to web based robots research.

Topics:

Wisdom is the principal thing: therefore get wisdom and with all thy getting get understanging (Proverbs of Salomon).

1. Introduction

First of all, we must consider the existence of a piece of relevant previous work in "*Grasp Determination for Unknown Objects*" [Sanzp et al., 1998][1]. This research activity takes as input the contour representation of an object, and then generates the most stable grasp associated with that object. In an extension of this work we have used *Local Grasp Determination* instead of the previous *Global* one, which supposes generating to the output not only the most stable grasp possibility, but also some other alternatives [Morales et al., 2001][2].

The novel contribution of the present thesis has been the first ever integration the grasping determination capability into the online robot. Thus, for example, the different grasping alternatives are presented to the user in order to allow him/her to select the most convenient grasp for a given situation. By default, the most stable one is used. Note the Grasping Determination and Execution module is fundamental for enabling the human to control the robot by using such a high level of interaction.

The description of the grasping determination algorithm has been included in this chapter to complement the explanation.

2.Grasp Determination

2.1. Grasping Stability

Regarding the grasping and manipulation by a robot hand, many papers have been published. With the aim of taking a fairly specific approach to the problem, we will only comment on those that deal most closely with subject of the research.

The many existing theoretical approaches are only of limited interest for practical applications [Ponce J et al., 1993][3], [Faverjon et al., 1991][4] and others generalized [Nguyen, 1988][5] by using algebraic cell decomposition and global optimization methods for computing two-finger force-closure grasps of curved 2D objects. Unfortunately, these techniques are computationally very expensive, since they require solving square systems of polynomial equations, typically of total degree 10. Moreover, this analytic methods usually require objects to be modeled by piecewise-smooth curves defined by parametric polynomial equations.

[1] [Sanzp et al., 1998] Sanz PJ, del Pobil AP, Iñesta JM, Recatalá G. *Vision-Guided Grasping of Unknown Objects for Service Robots*. In IEEE *Proc.* on Robotics and Automation (ICRA'98), pp. 3018-3025. Leuven, Belgium. May 1998.

[2] [Morales et al., 2001] Morales A, Recatalá G., Sanz PJ, del Pobil AP. *Heuristic vision-based computation of planar antipodal grasps of unknown objects*. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 583-588, Seoul. 2001.

[3] [Ponce J et al., 1993] Ponce J, Stam D, Faverjon B. *On computing force-closure grasps of curved two dimensional objects*. Int. J. Robotics Res., Vol 12, No.3, pp. 263-273. June 1993.

[4] [Faverjon et al., 1991] Faverjon B and Ponce J. *On Computing Two-Finger Force-Closure Grasps of Curved 2D Objects. IEEE Proc. on Robotics and Automation, pp. 424-429.* April 1991.

[5] [Nguyen, 1988] Nguyen V-D. *Constructing Force-Closure Grasps*. The International Journal of Robotics Research. Vol.7, No 3, June 1988.

To understand the key of the problem presented here it is necessary to distinguish between two meanings of "grasping stability" [Nakamura, 1991][1]. One is the ability to return to the static equilibrium position when the object position is perturbed. This ability should be termed "object stability". The goal here is related to the determination of the optimal grasp in an object within a gravitatory field, and it is the grasping considered in [Sanz et al., 1996][2], [Sanzp et al., 1998][3], where a single optimal solution is proposed. The other meaning is the ability to maintain contact when the object is subjected to disturbing forces. This ability should be termed "contact stability". In the latter case several grips associated with a specific object can be determined. This second meaning represents the domain of the contribution presented in this work.

2.2. A Model of Grasp Based on Vision

An approach that uses a heuristic strategy to obtain grasping points guaranteeing contact stability is presented. Experimental evidence shows that this basic strategy is general enough to be satisfactory for a broad spectrum of everyday objects, *a priori* unknown, in an on-line efficient manner.

Bearing this in mind, our first stage in the planar grasping determination is a careful segmentation that preserves shape, and then to extract the shape features that will be used to obtain suitable grasping points for a given hand configuration. It is supposed that objects are nearly planar and homogeneous, and that they are not known in advance: no attempt at shape identification is made. The only *a priori* knowledge is the geometry of the fingers of the parallel-jaw gripper.

The technical details of the strategy for grasping point determination are provided elsewhere [Sanz et al., 1996], so relations between the criteria that characterize a stable grasp are shown. This stability criterion is evaluated *on-line* from geometric reasoning of images extracted in real time by the vision system. Including computer vision in the grasp determination permits the grasp characterization through the definition of the following three thresholds:

- *Curvature threshold* (α). For taking into account smooth conditions on the grasping zones of the object, in order to guarantee the finger adaptation with a maximum contact. This threshold is related to the condition of object-gripper finger adaptation, which implies that contour zones of radius *k* (the finger radius) must be found, centered in the grasping points, and showing very low curvature conditions.

[1] [Nakamura, 1991] Nakamura Y. "Advanced Robotic Redundancy and Optimization". Adisson-Wesley series in electrical and computing engeneering. Control engineering. Reading Mass, Adisson-Wesley, 1991.

[2] [Sanz et al., 1996] Sanz PJ, Domingo J, del Pobil AP, Pelechano J. *An Integrated Approach to Position a Robot Arm in a System for Planar Part Grasping.* Advanced Manufacturing Forum, vol. 1, pp.137-148. Special Issue on Applications of Artificial Intelligence, 1996.

[3] [Sanzp et al., 1998] Sanz PJ, del Pobil AP, Iñesta JM, Recatalá G. *Vision-Guided Grasping of Unknown Objects for Service Robots.* In IEEE *Proc.* on Robotics and Automation (ICRA'98), pp. 3018-3025. Leuven, Belgium. May 1998.

- *Angular threshold* (β). For the *non-sliding* condition, using the friction model, guaranteeing that the grasping line is inside the friction cone; i.e. nearly normal to the contour at both points according with the "force closure condition" [Nguyen, 1988][1].
- *Distance threshold* (γ). Permits a maximum distance from the *centroid* of the object to the *grasping line* $\overline{P_1P_2}$ (the line that joins the grasping points P_1 and P_2), so that the grasping line is close to the object centroid.

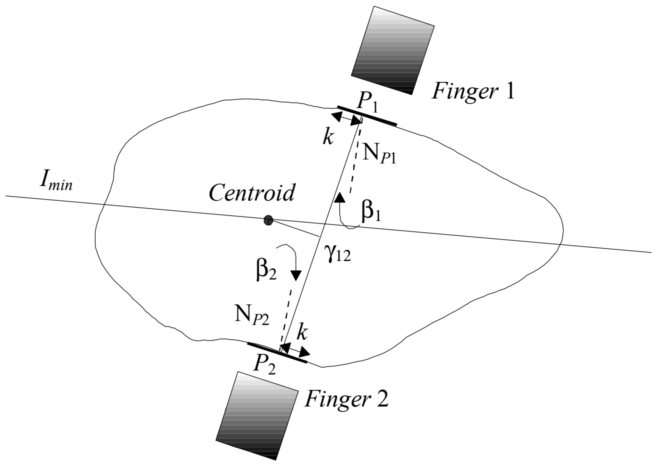


Figure V-1. Geometric interpretation of the stability conditions: smooth curvature in regions of radius *k* centered at P_1 and P_2; small angles β_1 and β_2 of the normals N_{P1} and N_{P2} with the grasping line; and a small distance, $d < \gamma$, from the centroid to that line (Adapted from [Morales et al., 1998][2]).

In a previous work, we exclusively included "object stability" (see 2.1) from the above conditions. But it should be pointed out the considerable advantage that would be gained by also computing "contact stability" [Nakamura, 1991][3], by means of relaxing the last condition (γ threshold). So description extends the previous research in that sense, showing their possible advantages in a more general manipulation domain.

It should be pointed out that only a few visual parameters were captured in execution time: the object centroid, the direction of the minimum inertia axis (I_{min} - main axis), and a description of the object contour, $\boldsymbol{C}$ [Sanz et al., 1996b][4] are necessary. The geometric interpretation of those conditions can be observed in Figure V-1, that shows a pair of grasping points, P_1 and P_2, under the assumption of a frictional point contact model, taking into account stability con-

[1] [Nguyen, 1988] Nguyen V-D. *Constructing Force-Closure Grasps*. The International Journal of Robotics Research. Vol.7, No 3, June 1988.

[2] [Morales et al., 1998] Morales A, Sanz PJ, del Pobil AP. *Computing Contact Stability Grasps of Unknown Objects by Means of Vision.* Proceedings of the IBERAMIA'98, pp. 241-252, Helder Coelho, Ediciones Colibrí. 1998.

[3] [Nakamura, 1991] Nakamura Y. "Advanced Robotic Redundancy and Optimization". Adisson-Wesley series in electrical and computing engeneering. Control engineering. Reading Mass, Adisson-Wesley, 1991.

[4] [Sanz et al., 1996] Sanz PJ, Domingo J, del Pobil AP, Pelechano J. *An Integrated Approach to Position a Robot Arm in a System for Planar Part Grasping.* Advanced Manufacturing Forum, vol. 1, pp.137-148. Special Issue on Applications of Artificial Intelligence, 1996.

ditions. Note that *d* is the distance between the grasping line and the centroid, and β_1 and β_2 are the angles between the grasping line and the normal to C at the grasping points (N_{P_1} and N_{P_2}).

2.3. The k-Angular Bending Vector

In grasping characterization it is necessary to provide a way to evaluate curvature of discrete curves. The measurement tools used in this paper are explained in more detail in [Sanz et al., 1996b]. One of the tools described is *k*-angular bending and *k*-angular bending vector. Before continuing with other matters, a description of these tools will be given here. These measurement tools are based on the *k*-vector notion [Rosenfeld et al., 1973][1]. A *k*-vector is a vector that joins the points p_i and p_{i+k}. This notion is basic in the definition of different curvature measurements. If we consider p_i a point of the contour we can associate two *k*-vectors $\vec{a}_{ik}$ and $\vec{b}_{ik}$. Using this k-vectors the *k*-angular bending is calculated as $\kappa_{ki} = -\vec{b}_{ki} \wedge \vec{a}_{ki}$.

Similar measurements are *k*-curvature and *k*-cosine. These tools are described extensively in [Rosenfeld et al., 1973][2].

It is important to notice the meaning of the sign of *k*-angular bending values. It is assumed that the list of points is ordered following clockwise contour sense. The consequences will be similar if the contrary assumption were taken. The main consequence is that the interior of the object will be always in the right hand of the sequence of points. It can be concluded from this that a positive value will be describing a convexity and a concavity in any other case.

The next step is the construction of the *k*-angular bending vector. This is obtained by grouping the *k*-angular bending of all points of a curve in a function. But, due to the fact that digital curves are obtained from a visual system and from real objects, a level of quantization noise is present. So a process of smoothing is needed in order to eliminate the irregularities derived from that noise.

Applying a convolution with a Gaussian kernel $G(\varpi)$ performs this smoothing of the *k*-angular bending function: $K_{ik}(\omega) = k_{ik} \otimes G_i(\omega)$. The resulting function will be known in this paper as the *k*-angular bending vector.

In [fig 2] the *k*-angular bending vector for the external contour of an Allen key is shown.

[1] [Rosenfeld et al., 1973] Rosenfeld A, Johnston E.*Angle detection on digital curves*. IEEE Transaction on Computers. Vol C-22, pp. 875-878. September , 1973.

[2] [Rosenfeld et al., 1973] Rosenfeld A, Johnston E.*Angle detection on digital curves*. IEEE Transaction on Computers. Vol C-22, pp. 875-878. September , 1973.

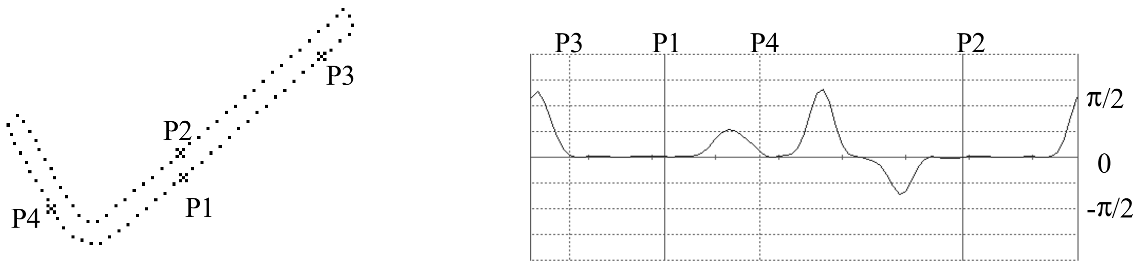


Figure V-2. Examples of the aspect of the k-angular bending vector. In this case the shape is an Allen key, and the values used for *k* and *w* are 3 and 1.0 respectively (Adapted from [Morales et al., 1998][1]).

Before ending this section it is important to notice that several parameters (such as *k*, and ω) have not been determined. These values will be adjusted according to geometric features or empirically.

Summarizing, a special tool has been developed, the *k*-angular bending vector, that is based on *k*-angular bending measure, which is itself a curvature measurement.

2.4. Grasping Regions

In the first section, it was stated that grasping points needed to have certain parameters within several threshold values. The first two were the α_i parameters (one by each point of a grasping pair) related with the named curvature threshold (α). In the previous section several methods for measuring the curvature of a contour point have been shown. It is important to remember that the contact of a finger with the object is not located at a single point, but it is located on a segment of the shape. So, it is interesting not only to determine which points have a satisfactory curvature, but also which segments of the shape have a constant satisfactory curvature. We will name those segments 'grasping regions'.

The next question is how those grasping regions will be detected. A simple procedure is to detect a sequence of neighboring points on the contour that reaches the curvature threshold. Any of the curvature measuring tools, even the angular bending vector can perform it. The last option is more useful than it seems. This is because a process of smoothing has been used for filtering that vector. Noise problems could appear if direct curvature measures were being used.

Once the *k*-angular bending vector has been established as the tool for measuring the curvature, the parameters that define the vector must be determined. These parameters are *k* and ω.

Different values for *k* and ω were used in previous works. In this paper other facts have been taken in consideration. The most important of them has been to relate the value of *k* to

[1] [Morales et al., 1998] Morales A, Sanz PJ, del Pobil AP. *Computing Contact Stability Grasps of Unknown Objects by Means of Vision.* Proceedings of the IBERAMIA'98, pp. 241-252, Helder Coelho, Ediciones Colibrí. 1998.

the geometry of the finger. In fact, it is similar to the meaning of *k* in [Sanz et al., 1996][1], which is the radius of a planar finger. A value of 3 has been taken for *k* during trials. In reference to ω, during trials it has been observed that the most adequate value is 1.0.

The next step is to use the *k*-angular bending vector for determining the grasping regions. It is important to remember that a grasping region is a segment of contiguous points that are within the curvature threshold requirement (α threshold). The procedure for determining grasping regions is simple. It will consist in filtering the points that are under the curvature threshold limit, and grouping them according to its neighborhood.

But we must consider whether these groups are grasping regions or not. Some problems may arise at this point. It may be that a group consists of only one point. This means that the contiguous points in the contour have not reached the level of curvature required. Probably this point will prove unsuitable as a grasping point.

Testing if a group is composed of a minimum number of points can solve this problem. With this requirement, it will be possible to assure that the curvature is constant under threshold limits in a sequence of points.

Once, they have been filtered, the groups can be considered grasping regions (GR). Grouping a great number of points in a smaller number of regions has some important advantages. Some considerations can be made about all the points that are included in a region. First, all of them reach the curvature requirement. And all of them have the same normal vector. A grasping region can be seen then as a straight segment of the shape (see Figure V-3).

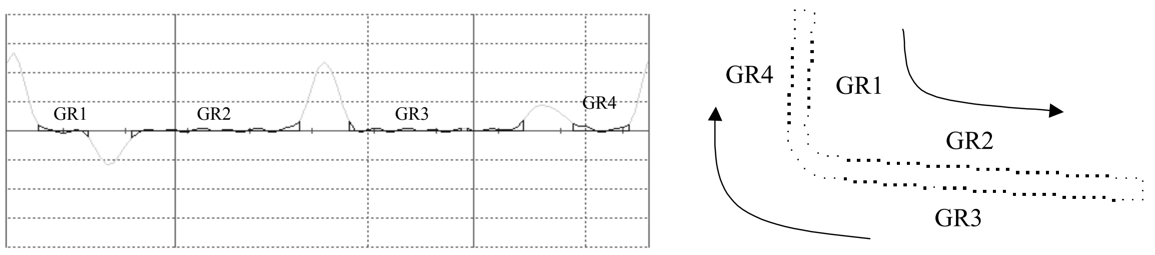


Figure V-3. In this figure it is possible to see how grasping regions are determined from the *k*-angular bending vector, and what the correspondence with object contour shape is.

The feature described in Figure V-4 is useful, but a problem appears in the case of circular shapes, or in long slight curves. In these cases, especially in the first the whole shape can present a small value of curvature. So the whole shape could be included in a large region. In the second case the region could get the whole curve. It is evident that in these cases the region would not be a straight segment.

The solution lies in considering the accumulated curvature. Each point included in a region presents a value of curvature that will be probably different from 0. When grouping points, a new point will be added to a group if the sum of curvatures does not reach a fixed limit. This

[1] [Sanz et al., 1996] Sanz PJ, Iñesta JM, del Pobil AP. Towards and Automatic Determination of Grasping points Through a Machine Vision Approach. In Proc. of the Ninth Intl. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'96), pp. 767-772. Fukuoka, Japan. 1996.

limit is defined in function of α threshold. If this limit is reached, the point will be the first point of a new region.

With this technique the greater regions will be split in some smaller regions. This makes it possible to assure that the regions can be approximated as straight segments.

2.5. Compatible Regions

Grasping regions are segments where a planar finger could be placed, although to get a complete grasping it is necessary to place the two fingers, each of them in a different region, with opposite normal vectors. Determining which pairs of regions allow a stable grasping will be also necessary. The next step of the strategy will consist in finding these pairs of regions.

It is important to remember that the stability of a grasping pair is affected by the β_i parameters, its related thresholds, and the grasping line that they define. If two regions are compatible there will be at least one pair of points, each one of them within the β threshold limits.

A definition of compatible regions is a couple of regions that contain at least two points (one per region) that meet the threshold requirements imposed by α and β.

The next question is how to determine whether two regions are compatible. Intuitively, two regions are compatible if they are fronted. Determining this compatibility more formally requires two conditions, the first of them is that the difference between angles of the normal vectors of each region must be π, and the second is that the projection of each region on the other region line must intersect with the other region.

Related to the first condition, several facts must be taken into consideration at this point. First of all, what the normal vector of a region is, and second how the β threshold can be inserted in this scheme.

In the previous section it was stated that a grasping region was approximately a straight segment, so it can be considered as a segment of a line, which will name the region line. The normal vector of a region will be the normal vector of this line. Moreover it must be assumed that the sense of the vector is from the shape to the inside of the object.

If the difference of two normal vectors is π it means that they are parallel lines. The β threshold allows some range of error in the parallelism of these lines. In fact, an error of at least 2β will be acceptable.

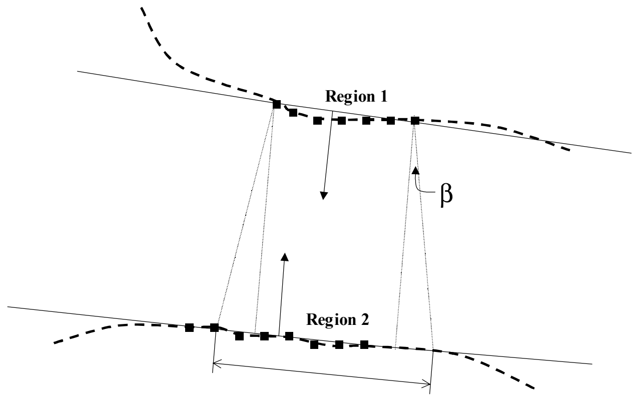


Figure V-4. Compatibility test between grasping regions (Adapted from [Morales et al., 2001][1])

To be sure that two regions are fronted, this requirement on its own is not enough. The segments could be perfectly parallel, but they might not be fronted. A second requirement is needed there. In this case, the β threshold must also be taken into account. It will affect the projections that will be expanded according to the value of β. When these two requirements have been reached the compatibility of two regions can be assured.

2.6. Looking for the Grasping Points

A method for recognizing compatible regions has been developed. Two regions are compatible whether two points exist which are within the α and β threshold requirements. But no other information about this pair is given.

The last stage in the strategy would be to find a pair of grasping points within the compatible regions specified. These could be seen as a refinement of the compatible regions.

Before starting, it is important to notice that the compatibility of regions has been set in terms of α ant β thresholds. The γ threshold has not been taken into account. So it is possible that a refined grasping does not met the three threshold requirements. The three thresholds assure object stability [Nakamura, 1991][2], but α ant β thresholds assure contact stability only. So the refinement performed over compatible regions will find a grasping pair that assures contact stability, but object stability will be not assured.

For performing the refinement several methods could be used. The first and most effective will be to make trials with all the possible pairs of points and finally choosing the best pair, which offer the best values of stability. The evident disadvantage of this method is the great number of pairs that it would have to test. But the great advantage is that it will find the best one.

[1] [Morales et al., 2001] Morales A, Recatalá G., Sanz PJ, del Pobil AP. *Heuristic vision-based computation of planar antipodal grasps of unknown objects*. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 583-588, Seoul. 2001.

[2] [Nakamura, 1991] Nakamura Y. "Advanced Robotic Redundancy and Optimization". Adisson-Wesley series in electrical and computing engeneering. Control engineering. Reading Mass, Adisson-Wesley, 1991.

Another method will be to propose an initial pair, and to enhance this one by making little modifications (moving the points to one of its neighbors). The convergence to the best pair is based on the assumption that a region can be seen as a straight segment.

But in these two methods it is necessary to include criteria for deciding which is the most stable pair. The best solution will be a function that gives a numeric measure of the stability. It is partially solved in other works [Fuentes et al., 1994][1], but an exhaustive computational effort is required and a different grip model is used.

In this chapter a simple and low computational cost method is proposed.

First of all, the method is focused on isolated parameters. Each one of them (α_i, β_i and γ_i) has an optimal value, which is π for α_i and 0 for β_i and γ_i. And they also have a limit that is defined by its thresholds (α, β and γ). An independent measure of stability could be defined for each visual parameter. For instance, this would be the measure for β_i parameter

$$S(\beta_i)_{\beta'} = \frac{\beta_i}{\beta'}$$

where α is the value of the characteristic, and α' is the threshold of this parameter. This function has some very interesting features. When the characteristic has its optimal value (α = 0) the result is 0, and when the characteristic gets the threshold value ($\alpha = \alpha$') the function gives 1. Moreover, it is an increasing linear function. The values from 0 to 1 will indicate a stable value of this parameter, and over 1 indicates instability. But the most important thing is that it gives a value of stability, which can be used for comparing the characteristics of different points.

But this has been defined for an isolated parameter. The next step will be to integrate the measure of stability for all the characteristics of a grasping. A function that performs this integration is this

$$S(\alpha_1, \alpha_2, \beta_1, \beta_2) = MAX\left\{S(\alpha_1)_{\alpha'}, S(\alpha_2)_{\alpha'}, S(\beta_1)_{\beta'}, S(\beta_2)_{\beta'}\right\}$$

This function presents the same features as the previous one. From 0 to 1, the stability is assured. It also gives a measure of stability or instability.

It should be pointed out that the γ characteristic has not been included. The above function gives a measure of contact stability. At present it is not clear how the γ_i parameters must be included.

2.7. Grasping Selection

At this point, a strategy for finding pairs of grasping points has been completely developed. It is different from the strategies proposed by [Sanz et al., 1997][2]. The difference between them

[1] [Fuentes et al., 1994] Fuentes O, Marengoni HF, Nelson RC. *Vision-based Planning and Execution of Precision Grasps.* Techincal Report 546, The University of Rochester, Computer Science Deparment, New York. 1994.

[2] [Sanz et al., 1997] Sanz PJ, del Pobil AP, and Iñesta JM. *Curvature-Symmetry Fusion in Planar Grasping Characterization from 2D Images.* In *Proc.* of the Tenth Intl. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'97), pp. 45-52. Atlanta, U.S.A. 1997.

lies in precisely what these methods are searching for. The method proposed by Sanz looks for the best grasping taking into account the object stability. The method, which is proposed in this chapter, takes a different approach. It finds all the possible pairs of grasping points and then selects the best one. Moreover, this strategy guarantees that all these grasping pairs present contact stability.

The criteria used in selection may be different. If the similarity with Sanz's results is the goal, a criterion that selects the pair of grasping points with the best γ value can be used. But other criteria, depending on the application, are also possible. The most important feature of this strategy is that it does not discard any possible grasping. It will determine the pairs of grasping points that are possible and a high level algorithm will decide which is the best depending on the application and the selected criteria. In this sense, it is a more flexible strategy than the one previously shown in [Sanz et al., 1997].

NOTE: *Security conditions are not taken into account yet. Once a grasping has been proposed it must be checked. This check determines if a grasping is possible according to the physical limitations of the grip, such as maximal separation of jaws, or the possibility of placing fingers at selected points without collisions. The first problem is when to do this checking. Two alternatives are possible: The first is to perform the checking during refinement, and the second one, is to do it after selecting the best grasping. The first option would complicate the algorithm being used to refine the grasping, due to this fact it includes a new parameter to be considered apart from thresholds. The second one is simpler than the first. The check is done after selecting the best grasping. Now the problem arises when a pair of grasping points does not meet the security conditions. In this case there are also two possible options. The first one is to discard the grasping and select another one among the other proposed grasping pairs. The second option is to try to enhance the refinement of grasping in order to met security conditions. If it is not possible, the first option is always available.*

In conclusion, a complete algorithm for performing grasping selection would be like this

```
Algorithm Grasping Searching

Determine regions
For all compatible pairs of regions
      Refine grasping
Choose best grasping
Repeat
If grasping does not met security conditions then
      Refine grasping with security conditions
      If it does not met security conditions yet then
            Choose another grasping pair
Until finding a secure grasping or there is no possible grasping
```

The strategy offers other important advantages. The most important is the reduction of searching space. In other works [Faverjon et al., 1991][1] all possible combinations between two points of the contour could be tested. The new strategy will filter out unstable points and will test only points belonging to different regions; even combinations of incompatible regions will be discarded. In fact, this reduction of searching space will permit all possible grasping pairs to be found within real time constraints.

[1] [Faverjon et al., 1991] Faverjon B and Ponce J. *On Computing Two-Finger Force-Closure Grasps of Curved 2D Objects. IEEE Proc. on Robotics and Automation, pp. 424-429.* April 1991.

3.Method Validation

In the first place we show a qualitative set of results (see table 1), that will permit comparisons to be made with other works. The Allen key, object 1 in table 1, has been chosen for comparing the results with a previous one, shown by [Sanz et al., 1997][1], where the same object was used. In [Sanz et al., 1997] one grip is proposed only, (the strategy used in that paper only finds the optimal solution, according with "object stability" criteria). Although, in our case, after several trials two solutions have been found, and one of them is the specific solution proposed in Sanz's work.

The pincers, object 2 in the table 1, are an example of a complex contour. Many types of grips are proposed in this case. Some of them are little modifications of others, so they can be grouped into families. But the most important result that can be observed is the two different kinds of types of grips proposed. The first one refers to the grip over the external shape of the pincers. Note that these grips could be discarded by security conditions. The second type includes the solutions that get only one branch of the pincers, which is a sensible option for grasping pincers. A third group of solutions is proposed around the tips of pincers, but they will be probably discarded due to security conditions (collision of fingers with other parts of the object).

The third example, object 3 in table 1, is probably the most interesting. It has been taken from [Faverjon et al., 1991]. In this paper a method for obtaining all grips that meet force closure is proposed. The similarity with our method is evident, but there are important differences. In [Faverjon et al., 1991] the figures are modeled by parametric curves, while we use a real world vision model, which is affected by many undetermined factors, inherent to the digitalization process itself (e.g. quantization noise). Moreover Faverjons's paper proposes an exhaustive computational algorithm, which can not be acceptable under the real-time constraints in a robotic domain. In [Faverjon et al., 1991] several solutions are proposed, so it would be interesting to compare our results with them. In summary, almost all of the solutions proposed by [Faverjon et al., 1991] are also found here, but some comments must be made. In some cases, a family of grips has been proposed related to only one solution of Faverjons's. It is caused by the fragmentation of grasping regions. Some other grasps do not appear in our solutions, because the main cause of this is the irregularities of contour (due mainly, to the quantization noise). They make impossible the existence of certain stable regions that appear in Faverjons's proposals. In this paper we have worked with real objects that have no perfect contour. Finally there are some regions that are discarded when the geometry of the fingers (the radius) is taking into account.

[1] [Sanz et al., 1997] Sanz PJ, del Pobil AP, and Iñesta JM. *Curvature-Symmetry Fusion in Planar Grasping Characterization from 2D Images*. In *Proc.* of the Tenth Intl. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'97), pp. 45-52. Atlanta, U.S.A. 1997.

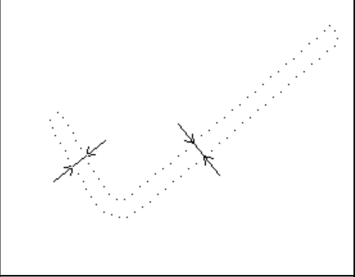
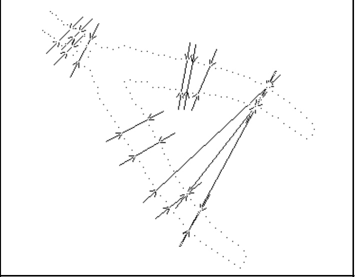
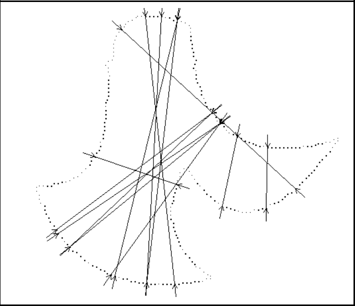
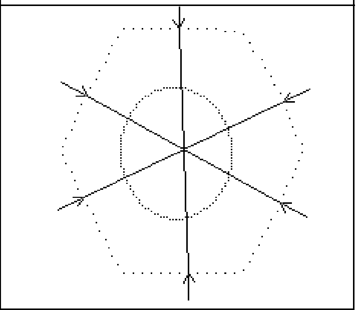
Object 1: Allen Key	Object 2: Pincers
	
Object 3: From [Faverjon et al., 1991]	Object 4: Hexagonal Nut
	

Table V-1: These are four examples of shapes used during tests. The first object is an Allen key; the second is a pair of pincers; the third object is a special figure that was proposed in [Faverjon & Ponce, 91]; and the last object is a nut. In each of the objects the grasps proposed by the strategy are represented.

Figures	Image Analysing	Grasping refinement
Allen Key	0.207 sec.	0.002 sec.
Pincers	0.209 sec.	0.018 sec.
Figure from [Faverjon et al., 1991]	0.210 sec.	0.018 sec.
Hexagonal Nut	0.208 sec.	0.003 sec.

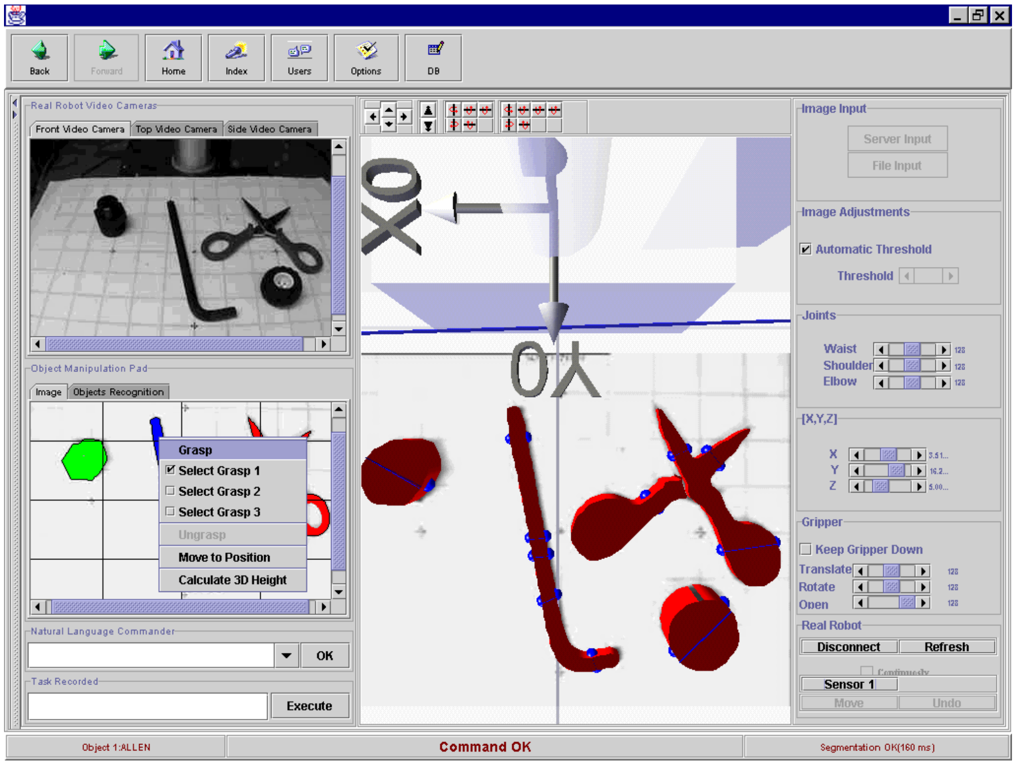
Table V-2. This table represents the times required for the determination of all grips. Region determination is the time needed for determining all the grasping regions of a contour. And Grasping refinement refers to the time needed for determining compatible regions, and for finding a grasping pair between these regions.

The last object, object 4 in table 1, is a hexagonal nut. An interesting result showing how all the possible grasps of the figure can be found is observed. A high-level task planner must decide which of them is the best, depending on the situation and the context. This can be said about all the objects. In the end the goal of this strategy is to determine all stable grasps of an object according to contact stability criteria, and high-level applications will decide which the most suitable grasp is.

Another important result is that the required times for running the operations of the strategy are close to real time. From table 2 we can observe that the times are small enough to meet real-time constraints.

4.The User Selects the Grasping Alternatives

As can be appreciated in Figure V-5, once the original state of the virtual environment is established the Grasp Determination algorithm is applied to every object in the scene. Hence, for example, the allen key provides several grasping alternatives, which are presented into the 3D virtual environment as a blue line and two spheres.



Object 1:ALLEN

Command OK

Segmentation OK(160 ms)

Figure V-5. Selecting different grasp alternatives for the allen key.

By clicking over an object at the Manipulation Pad (e.g. Clicking the allen key in Figure V-5) a menu comes up that allows executing the Grasp action. By default, the most stable grasp alternative is selected (i.e. Checkbox with “Select Grasp 1”). Moreover, if the user prefers using other alternatives (up to three) he/she is able to do it.

5. Conclusions and Future Work

A new method to compute all the grips taking into account “contact stability” on unknown planar parts with exclusively visual parameters as input has been shown. This approach extends a previous one that obtains grips under “object stability” conditions.

The algorithm has been implemented in such a way that enables the user to select the different grasping alternatives in the predictive display before sending the action to the real robot. By default, the most stable grasping possibility is applied.

It is important to take into account that some information extracted from the Grasp Determination procedure could be very useful in order to extract object features that would allow us to implement a better object recognition module. In fact, we are already working in extending the research described in Chapter IV, in order to make it more accurate and less time consuming.

References

[Faverjon et al., 1991] Faverjon B and Ponce J. *On Computing Two-Finger Force-Closure Grasps of Curved 2D Objects. IEEE Proc. on Robotics and Automation, pp. 424-429.* April 1991.

[Fuentes et al., 1994] Fuentes O, Marengoni HF, Nelson RC. *Vision-based Planning and Execution of Precision Grasps.* Techincal Report 546, The University of Rochester, Computer Science Deparment, New York. 1994.

[Groan et al., 1978] Groan FCA, Verbeek PW. *Freeman-Code Probabilities of Object Boundary Quantized Contours.* Computer Vision, Graphics, Image Processing, vol. 7, pp391-402. 1978.

[Iñesta et al., 1997] Iñesta Jm, Sanz PJ, del Pobil AP.*An Automatic Transformation from Bimodal to Pseudobinary Images.* Image Analysis and processing. Lectures Notes in Computer Science, vol 1310, pp. 231-238, A del Bimbo, Springer-Verlag. 1997.

[Marín et al., 1998] Marin R, Recatalá G, Sanz P, del Pobil AP, Iñesta JM. *Telerobotic System Based on Natural Language and Computer Vision.* Lectures Notes in Artificial Intelligence, vol II, pp. 353-364, A. del Pobil, J. Mira & M. Ali (Eds.). Springer-Verlag. 1998.

[Morales et al., 1998] Morales A, Sanz PJ, del Pobil AP. *Computing Contact Stability Grasps of Unknown Objects by Means of Vision.* Proceedings of the IBERAMIA'98, pp. 241-252, Helder Coelho, Ediciones Colibrí. 1998.

[Morales et al., 2001] Morales A, Recatalá G., Sanz PJ, del Pobil AP. *Heuristic vision-based computation of planar antipodal grasps of unknown objects.* Proceedings of the IEEE International Conference on Robotics and Automation, pp. 583-588, Seoul. 2001.

[Nakamura, 1991] Nakamura Y. "Advanced Robotic Redundancy and Optimization". Adisson-Wesley series in electrical and computing engeneering. Control engineering. Reading Mass, Adisson-Wesley, 1991.

[Nguyen, 1988] Nguyen V-D. *Constructing Force-Closure Grasps.* The International Journal of Robotics Research. Vol.7, No 3, June 1988.

[Ponce J et al., 1993] Ponce J, Stam D, Faverjon B. *On computing force-closure grasps of curved two dimensional objects.* Int. J. Robotics Res., Vol 12, No.3, pp. 263-273. June 1993.

[Rosenfeld et al., 1973] Rosenfeld A, Johnston E.*Angle detection on digital curves.* IEEE Transaction on Computers. Vol C-22, pp. 875-878. September , 1973.

[Sanz et al., 1996] Sanz PJ, Domingo J, del Pobil AP, Pelechano J. *An Integrated Approach to Position a Robot Arm in a System for Planar Part Grasping.* Advanced Manufacturing Forum, vol. 1, pp.137-148. Special Issue on Applications of Artificial Intelligence, 1996.

[Sanz et al., 1996b] Sanz PJ, Iñesta JM, del Pobil AP. Towards and Automatic Determination of Grasping points Through a Machine Vision Approach. In Proc. of the Ninth Intl. Conf. on

Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'96), pp. 767-772. Fukuoka, Japan. 1996.

[Sanz et al., 1997] Sanz PJ, del Pobil AP, and Iñesta JM. *Curvature-Symmetry Fusion in Planar Grasping Characterization from 2D Images*. In *Proc.* of the Tenth Intl. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'97), pp. 45-52. Atlanta, U.S.A. 1997.

[Sanzp et al., 1998] Sanz PJ, del Pobil AP, Iñesta JM, Recatalá G. *Vision-Guided Grasping of Unknown Objects for Service Robots*. In IEEE *Proc.* on Robotics and Automation (ICRA'98), pp. 3018-3025. Leuven, Belgium. May 1998.

Chapter VI **SPEECH RECOGNITION**

Up to now, the system is able to receive camera inputs, recognize the whole set of objects, and calculate the grasping point possibilities for every one of them. Thus, if for example we type into the command field the operation "Grasp cube", the robot will execute the most stable grasping associated with that object. The new challenge consists of providing this "Grasp cube" command in a more natural way, which is the voice.

This Chapter shows the way we have implemented a speech module that provides the transfer between voice input and the convenient text based command to be executed by the robot. The novel contribution is the way the procedure is defined to be run over the Internet, and the interface implemented to connect any kind of external speech recognition program to the robot controller.

First section gives an overview of the whole service, from the user point of view. It introduces some of the technologies (e.g. Microsoft Speech SDK) used to implement this capability.

Secondly, the software architecture is presented, giving low level details on the connectivity procedure and the interface defined between the remote controller and the speech recognition and synthesizer program.

And finally, results show the performance of the speech recognition module and its computing time for both, when the user has passed the speech training procedure, and when the operator is commanding the robot directly without training the speech recognition engine.

Topics:

All that is human must retrograde if it does not advance (Edward Gibbon).

1. Introduction

After several decades of research on automatic speech recognition and the advent of affordable desktop computers that can deploy speech recognition in real time, voice recognition software components have become popular and applications have been provided with the possibility of exploiting voice interfaces with low development costs. It opens the doors to applying such a capability with aim of improving the way humans can communicate with a robot remotely.

Speech recognition technology is becoming mature enough to be successfully used in human-machine interaction. Current speech recognition systems are based on Hidden Markov Models (HMM), which are probabilistic finite-state automata that can be efficiently trained from examples [Rabiner et al., 1993][1] [Jelinek, 1998][2]. An interesting property of HMMs is that they can be combined to form new HMMs. A word, for example, can be defined as a network of phonetic HMMs which is a new HMM, as far as the network is a probabilistic finite-state automaton. The set of sentences in a limited domain task, such as controlling a robot, can be properly modeled in terms of a finite-state automaton of words (regular grammar). Thus, the resulting language model is a large HMM. In order to recognize a sentence, the Viterbi algorithm (or its faster, suboptimal beam-search version) is used to find the most likely visited sequence of states. The pronounced words can be recovered by detecting HMM boundaries in the optimal sequence of states. Sometimes, it is not necessary to know the exact sequence of words, but only the final state, since it can represent a semantic value such as "Grasp object [number]", where [number] is an attribute. The speech recognizer is then performing some kind of "understanding" of commands, which is appropriate for controlling the robot.

In recent years, speech recognition systems are being distributed as applications for low-end systems. IBM [Viavoice][3] and Microsoft [SAPI][4] are distributing free versions of their software development kits (SDK) for automatic speech recognition. Both systems allow the programmer to define the grammar that models the specific task language and to use the microphone as an input device for the application. Microsoft is currently trying to define a standard API for speech recognition libraries. In this work, we have used the Microsoft Speech API (SAPI) SDK to build the speech recognizer to command the robot.

In this Chapter we present the way in which the UJI Telerobotic Training System has incorporated a speech recognition module into its architecture. It enables the user to send commands to the robot not only by mouse interactions, or using written commands on a graphical panel, but also *speaking directly via the microphone*. The description focuses on the way the voice-based interface has been implemented to allow access through the Internet and in particular, how this has been applied to the telerobotics domain.

[1] [Rabiner et al., 1993]	Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Englewood Cliffs NJ: PTR Prentice Hall (Signal Processing Series), 1993.
[2] [Jelinek , 1998]	Jelinek, F. Statistical Methods for Speech Recognition. MIT Press:Cambridge (1998).
[3] [ViaVoice]	The IBM Viavoice Automatic Speech Recognition SDK (http://www-4.ibm.com/software/speech/dev/)
[4] [SAPI]	Microsoft Speech SDK (http://www.microsoft.com/speech/speechSDK/sdkinfo.asp)

Figure VI-1 shows the operator using a microphone and headphones to interact with the robot. The photo on the left shows the Java applet that interprets and predicts the user commands. The photo on the right shows how the robot is executing the commanded actions on the real scenario.

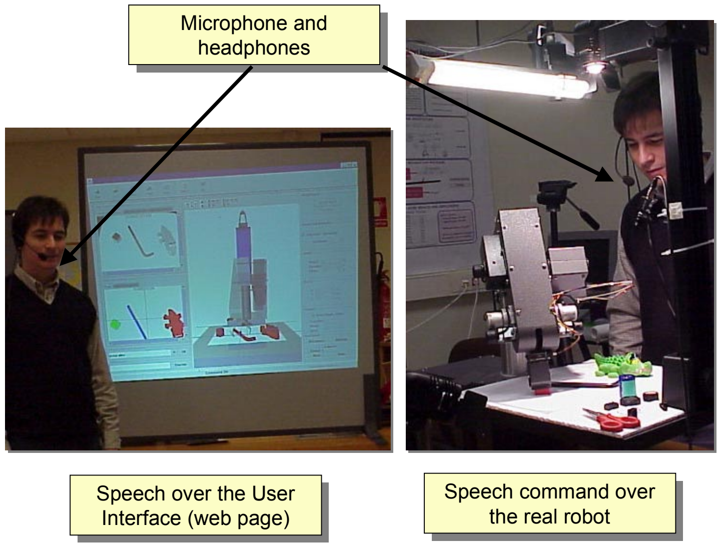

Microphone and headphones
Speech over the User Interface (web page)
Speech command over the real robot

Figure VI-1. Real robot being commanded with the voice

Firstly, the software architecture is presented, which shows the way the speech recognition module, running on the client machine can, communicate with the 3D virtual environment, in order to specify a manipulation task by simply using voice input. Subsequently, the set of available commands to the user is summarized, which gives an idea of the potential of the whole system. Thirdly, some design considerations related to the speech recognition module are discussed. Finally, the performance of the speech recognition module (recognition rate and its computing time) is presented before and after a training phase. The recognition rate is compared with the performance of users when dealing directly with the written text interface.

2. Architecture

As shown at Figure VII-2, the client side has five main modules: the "*HTML Robotics Tutorial*", the "*Online Teaching Assistance*", the "*Lab Exercises*", the "*Telerobotic Controller*", and the "*Speech Recognizer & Synthesizer*" module. The Telerobotic Controller implements the whole set of functionalities (image processing, object recognition, visually guided grasping, etc.) necessary to command the robot by using high level text-based commands like, for example, "*Grasp the allen*". In fact, the voice input is optional, and every robot feature can be programmed by using alternative types of user inputs.

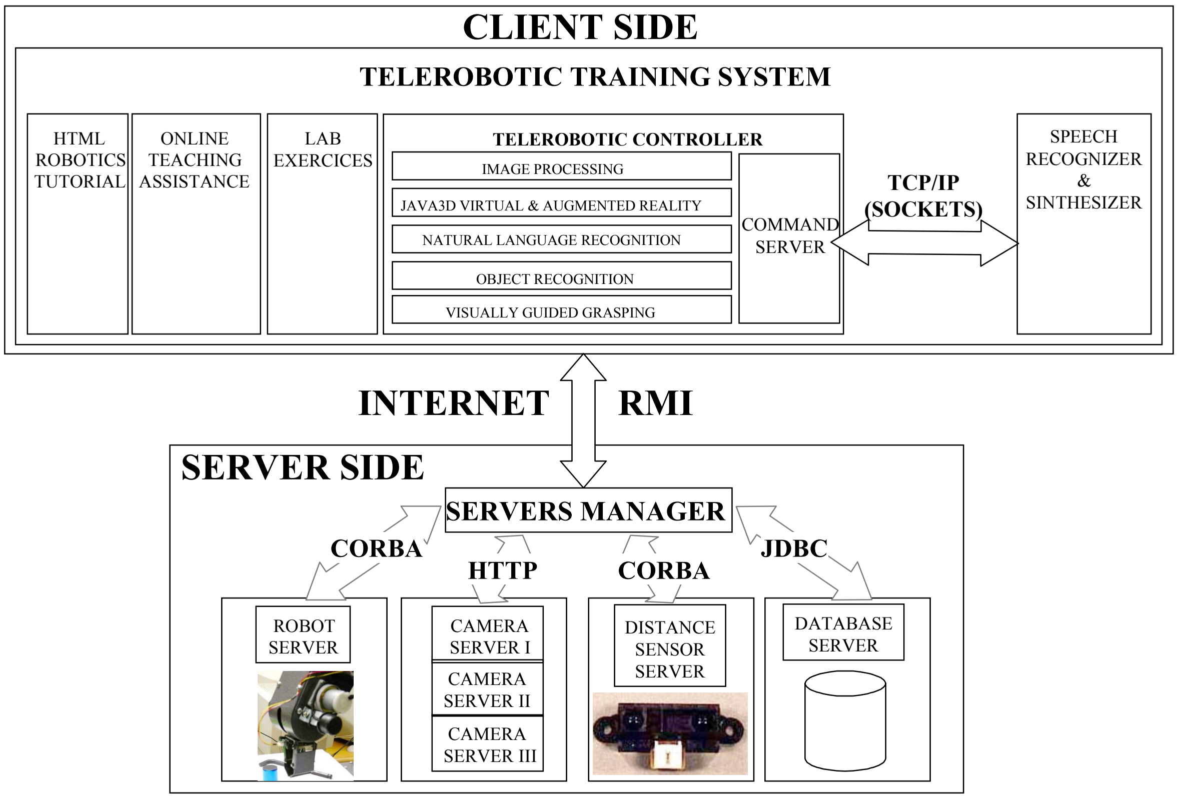
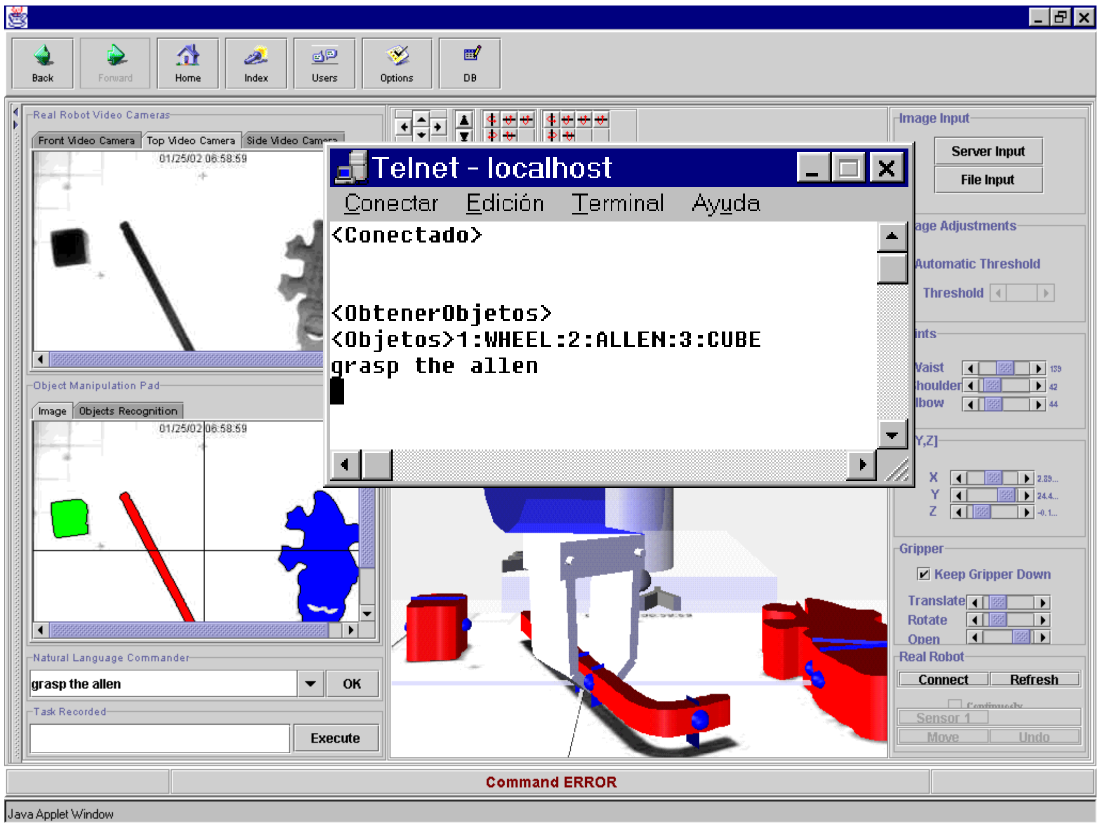


Figure VI-2. Software architecture

The situation now is, how can we open a door to external modules in order to accept these text-based high level commands from other programs that are optionally present in the system? The solution, as explained in Chapter III, consists of implementing a northbound interface by means of the *command server*. The *command server* submodule is listening to the TCP/IP 7745 port on the client machine for incoming connections, and once the external control has been got, the commands read from this port are interpreted into the *telerobotic controller,* in the same way as if they would have been typed directly into the Java applet window. In fact, any external program (e.g. telnet application) could use this *northbound* interface in order to command the robot.

Please note that Java applets have a security restriction that prevents them from opening a port on the client side machine when the applet has not been certified by the operator. It means, up to now, that in order to perform the voice interaction it is necessary to accept the Java applet as a trusted source.

As seen at Figure VI-3, we could simply execute the telnet program and then specify a connection to the *local host* (client machine) on port 7745. Then, every command written into the telnet application would be received and interpreted by the telecontroller module, in order to be executed on the robot. Thus, we could, for example, type "*Grasp the allen*", so the robot would perform accordingly. Besides this, some other features, like obtaining the objects' list in the scene ("<ObtenerObjetos>"), are enabled through the same technique.



```
<Conectado>


<ObtenerObjetos>
<Objetos>1:WHEEL:2:ALLEN:3:CUBE
grasp the allen
```

Natural Language Commander: grasp the allen

Command ERROR

Figure VI-3. Controlling the robot by using the northbound interface through the telnet program

The same procedure shown at Figure VI-3 to control the robot interface from the outside, can be utilized by other applications like, in our case, the *speech recognizer & synthesizer*.

As presented in Figure VII-4, once the speech recognizer application has been launched on the client side, it asks the controller for the list of objects on the scene. After that, the user can speak directly to the microphone and, if the sentence is properly recognized, the command is transmitted to the remote controller in order to activate the robot. Auditive feedback is returned from the robot by means of speech synthesis to notify whether the operation has been successful or not. Once the operator stops the voice interaction, the speech recognition module closes the connection.

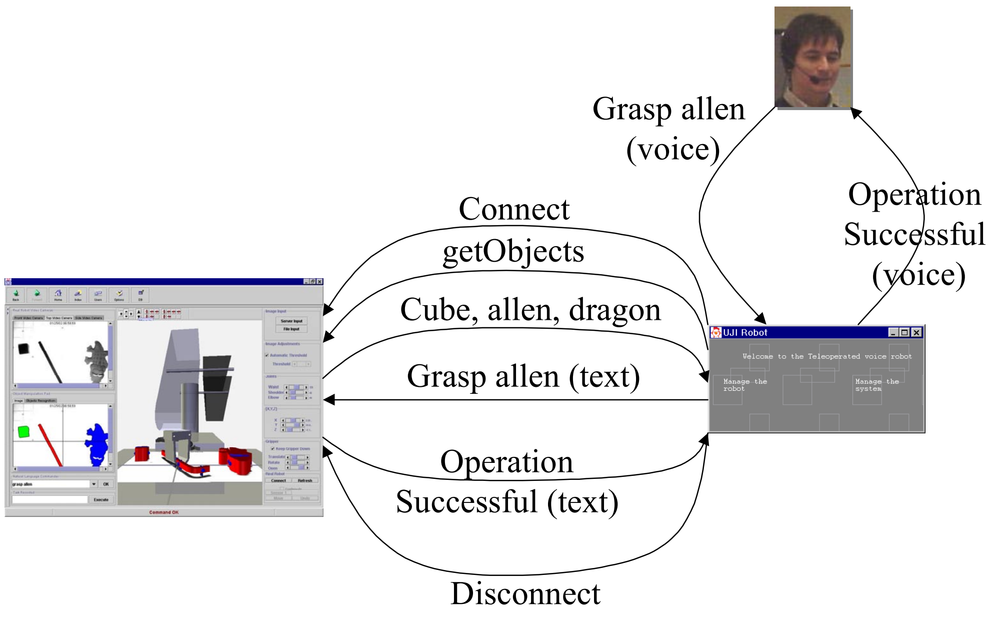


Figure VI-4. Use case of connecting the speech recognition module to the robot controller

In summary, any application can connect to the remote controller by using the TCP/IP port 7745 on the client machine. This means that programs like speech recognizers, motion planners, etc., can use this interface to improve the robot capabilities. This kind of functionality is very convenient in the education and training domain, to let students to create their own robot programs by using any programming language (e.g. Python, C, Java, etc).

3. Available Commands

In order to control the robot by using a voice input, we can use the whole set of commands accepted by the command server module (see Chapter II for more details). The speech recognition module extends this simple language by adding synonyms and more natural ways of giving spoken commands. Basically, the set of available voice commands is the one explained in Figure VI-5.

Command	Description

Grasp the {object name}
Take the {object name}
	Given a scene with objects, it executes the most stable grasping alternative on the object {object name}

Grasp the object {number}
Take the object {number}
	Given a scene with several objects, it executes the most stable grasping alternative on object {object number} numbered from top to down and from left to right.

Move ahead	It moves the TCP ahead 1 cm
Move down	It moves the TCP down 1 cm
Move left	It moves the TCP to the left 1 cm, taking the manipulation pad orientation as the reference
Move right	It moves the TCP to the right 1 cm, taking the manipulation pad orientation as the reference
Move to position {x} {y} {z}	It moves the TCP to the world coordinates position [x, y, z]
Move up	It moves the TCP up 1 cm
Move to the sector {number} Move to the quadrant {number}	The scenario is divided into 16 sectors, which are numbered from 1 to 16. The command moves the robot to the {number} sector.
Open gripper to {number}	It opens the gripper a {number}% of its capacity
Pick the {object name} up	It executes the grasping operation over the object labelled as the {object name}, and then elevates it 5 cm over the board
Pick the object {number} up	It executes the most stable grasping on object {number} and then elevates it over the board 5 cm (by default)
Place it at position {x} {y}	Having an object grasped in the gripper, it places it at position {x, y} in image coordinates
Place it over the {object name}	Having an object grasped in the gripper, it places it on top of the {object name}
Place it over the object {number}	Having an object grasped in the gripper, it places it on top of object {number}
Refresh Synchronize	It brings the robot to the initial position, captures the scene image, recognizes the objects, and constructs the 3D model.
Rotate gripper to {± number}	It rotates the gripper {± number}%
Ungrasp Release the object Drop the object Place it over the table	Having an object grasped into the gripper, it places it at the current robot position.
Ungrasp at position {x} {y} Release the object at position {x} {y} Drop the object at position {x} {y} Place it over the table at position {x} {y}	Having an object grasped in the gripper, it places it at position {x, y} in image coordinates
Ungrasp over the {object name} Release the object over the {object name} Drop the object over the {object name} Place it over the {object name}	Having an object grasped in the gripper, it places it on top of the {object name}
Ungrasp over the object {number} Release the object over the object {number} Drop the object over the object {number} Place it over the object {number}	Having an object grasped in the gripper, it places it on top of object {number}

Figure VI-5. Available robot commands at the northbound interface

The recognition module uses some additional commands to handle the user input (see Figure II-6).

Command	Description
Manage the robot Control the robot	It initiates the voice based human-robot interaction
Manage the system Control the system	It allows the user to access the voice recognition module's properties window
Stop listening Don't hear Disable Standby	It tells the voice recognition module to stop generating commands
Begin listening	It tells the voice recognition module to restart the recognition procedure

Figure VI-6. Voice recognition module's configuration commands

4. Design Details

4.1. Platform alternatives

In order to implement a speech recognition module, there are several alternatives. Possibly, one of the most famous ones is the "*Via voice naturally speaking*" of IBM [ViaVoice][1]. Other possibilities are the "*Dragon Naturally Speaking of Lernout & Hauspie*" [Dragon][2], or even the "*Hidden Markov Model Toolkit*" [HTK][3], that have the special feature of enabling programmers to use the speech recognition engine as a library to integrate their own voice-based applications. In our case, we are using the software components available with the Microsoft Speech SDK (Software Development Kit)[SAPI][4], which has the following characteristics:

- The SDK is free.
- The whole installation occupies 60 Mb.
- Once installed, it is integrated in the Windows operating system, and it is very easy to train the engine for a particular user.
- It is platform dependent, and works for Windows-based systems only.
- It enables recompiling the grammar at run-time, which is very convenient for our purpose.

[1] [ViaVoice] The IBM Viavoice Automatic Speech Recognition SDK (http://www-4.ibm.com/software/speech/dev/)

[2] [Dragon] Dragon Naturally Speaking of Lernout & Hauspie (http://www.lhsl.com/default2.htm)

[3] [HTK] Hidden Markov Model Toolkit HTK (http://htk.eng.cam.ac.uk/)

[4] [SAPI] Microsoft Speech SDK (http://www.microsoft.com/speech/speechSDK/sdkinfo.asp)

4.2. Changing the grammar in runtime

One of the most interesting features of the speech recognition module is its ability to change the input grammar once the application is running. The speech recognition error rate is greatly reduced when the grammar is modified to consider only objects that are present in a given scene.

Thus, once the operator takes control over the robot, the remote controller obtains the list of objects on the scene. Then, the voice recognition module gets this list and recompiles the grammar accordingly.

5.Results

We have run an experiment in order to assess the speech recognizer performance. Three different people uttered 80 sentences twice, once with the default phoneme models and once after training the system with their voices.

First of all, we present the computing time results, which show the speech recognition procedure takes an average of 0,2 seconds to accept a given command as successful. Some other times are presented as well, in order to have an idea of the response time of the system as a whole.

TABLE I
COMPUTING TIME

Operation	Time (sec)
Program Launching	5.06
Robot Initialization	3.32
Images acquisition	0.20
Image Segmentation and HU extraction	0.32
3D objects reconstruction	0.07
Objects Recognition	3.45
Speech Recognition	*0.20*
Robot movement execution	2.29

Table I shows the most time consuming operations are "Program Launching" and "Object Recognition Table II shows the performance of the voice recognition procedure in terms of the percentage of correctly recognized sentences.

TABLE II
PERFORMANCE

User	Untrained rate/deviation	Trained rate/deviation
User 1	76,25 / 1,50	92,5 / 0,78
User 2	87,5 / 0,59	97,5 / 0,30
User 3	67,5 / 0,88	72,5 / 1,09
Total Mean	*77,08*	*87,50*
Total Deviation	*6,55*	*8,64*

As can be appreciated in Table II, when using the speech recognition without any training, almost 80% of the sentences are properly accepted by the system. When training is completed, sucessful results are nearly 90%. Note the User 3 was not very fluent in English, so some of the sentences he said were not properly pronounced.

6.Conclusions

A speech recognition module based on the Microsoft Speech SDK has been implemented. A special feature of this system is that it is integrated into an already existing robot remote controller through the web. The result is a web page that enables programming a real robot by using not only text-based commands, but also voice interactions. Note the speech recognition module performs voice synthesis too, which means human-robot interaction can be accomplished by using simply a microphone and headphones.

The speech recognition module interacts with the system by means of a generic TCP/IP port that defines an external interface to control the robot (northbound interface). It allows students and researchers to use this as a way of creating their own specific robot programs, by using any existing programming language (e.g. Perl, Java, etc.).

Results have shown the potential of the system by means of the computing time and the voice recognition performance. Finally, current efforts are focused on extending the existing learning capabilities of the automatic object recognition procedure, to the speech module as well. This would not only enable the system to learn the representation of objects, but also several ways to identify them by using the voice.

References

[Dragon] Dragon Naturally Speaking of Lernout & Hauspie (http://www.lhsl.com/default2.htm)

[HTK] Hidden Markov Model Toolkit HTK (http://htk.eng.cam.ac.uk/)

[Jelinek , 1998] Jelinek, F. Statistical Methods for Speech Recognition. MIT Press:Cambridge (1998).

[Rabiner et al., 1993] Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Englewood Cliffs NJ: PTR Prentice Hall (Signal Processing Series), 1993.

[SAPI] Microsoft Speech SDK (http://www.microsoft.com/speech/speechSDK/sdkinfo.asp)

[ViaVoice] The IBM Viavoice Automatic Speech Recognition SDK (http://www-4.ibm.com/software/speech/dev/)

Part II

Part II: Results & Conclusions

Chapter VII THE SYSTEM IN ACTION

At this chapter, we will focus on a real application of the UJI Online Robot System that has proved highly successful: *the Education and Training domain*. That is the reason why the system is called *The UJI Telerobotic Training System* in some situations. When teaching Robotics to undergraduate students, one difficult problem is give every person the opportunity to program a real robot and develop the lab exercises in a scenario that goes beyond simulation. Robots are expensive devices, and even though our teaching laboratory is provided with 3 educational arms and 1 industrial manipulator, this number is not sufficient to almost 100 students with proper training. In this situation, the UJI Telerobotic Training System has provided 24 hour accessibility to an educational robot, as well as an unlimited number of connected students in a 3D virtual environment.

The chapter is organized in two main parts: (1) *System Validation: Education & Training*, and (2) *The UJI Online Robot Open To The World*. The first one focuses on the activities that students on the Robotics course have accomplished through the Telerobotic Training System. That is, students have been programming "Pick & Place" robotics tasks in a virtual environment that could be executed later on with the real robot. Moreover, as the students were interacting with other Robots on the Web, they have expressed their opinions that are presented as a comparative analysis. The results are really encouraging!

The second part shows the connectivity rates of people around the world that have been using the robot till the moment of writing. It is interesting seeing how the Telerobot motivates students from our University and even many people around the world to manipulate objects over a board using the Web as the medium of access.

Topics:

I cannot teach anybody anything, I can only make them think (Socrates).

1.System Validation: Education & Training

We have tested the worth of the system by means of an application in the Education and Training domain. Undergraduate students at the Jaume I University have been using the web-based interface in order to program “Pick and Place” operations in a controlled manner. The Education and Training application has been divided in two parts, which correspond to the two laboratory experiences that have been developed at our university during the Robotics course. The first one introduced the UJI Telerobotic Training System to the students. Then, they had to accomplish six "Pick & Place" operations using the educational robot. The task specification procedure was performed using the 3D virtual interface. After that, they could transmit their instructions on the real robot when accessible, from the laboratory or even better, from home.

The second education and training activity outlined the characteristics of three other web robots: "Telegarden", "Australia's Telerobot", and "Ariti". Finally, students give their opinion about them, by means of presenting a comparative analysis of their main capabilities.

1.1. The Students Program The UJI Online Robot

The first time students came to the laboratory they were introduced to capabilities of the UJI Telerobotic Training System: (1) Robotics Tutorial, (2) Online Teaching Assistance, (3) Complementary Exercices, (4) 3D virtual scenarios for industrial manipulators, and (5) The Educational Telerobot functionality (Pick & Place exercises).

1.1.1. Robotics Tutorial

First of all, we presented the existence of a web based Robotics Tutorial that they could use in order to better understand the complicated theoretical subjects that had already been explained at the classroom. As introduced in Chapter II, one of the most difficult concepts that students have to acquire in Robotics, is “Kinematics”. They must be able to understand and even resolve the direct and inverse kinematics associated with any robotic arm. This aspect of the Robotics Tutorial has worked very well, motivating the students and illustrating multimedia presentation of these advanced techniques [Sanz et al., 1998][1]. Apart from static and illustrated information, this multimedia Tutorial offers the user the possibility to interact with some screens and learn the “Kinematics” concepts in a more natural manner. Moreover, some of the explanations are prerecorded and can be heard by means of the speakers.

[1] [Sanz et al., 1998] Sanz P.J., Adell S. An undergraduate Robotics Course via Web. Teleteaching’98 Conference, a part of the 15th IFIP World Computer Congress. Distance Learning, Training, and Education. Austrian Computer Society (book series of). Edit. by Gordon Davies, pp. 859-867, Viena, Austria. 1998.

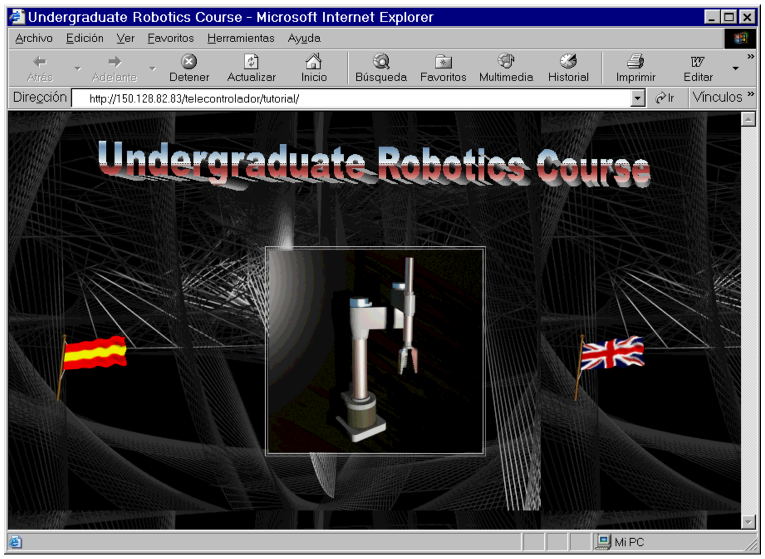
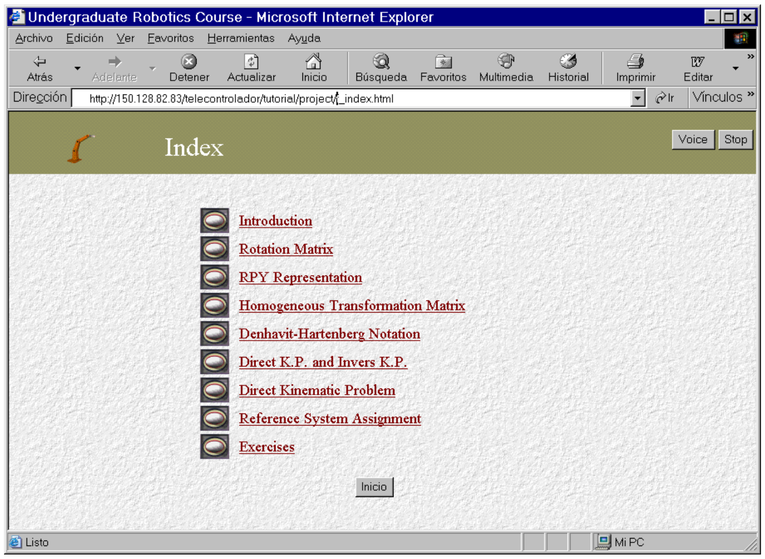

Undergraduate Robotics Course - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Adelante Detener Actualizar Inicio Búsqueda Favoritos Multimedia Historial Imprimir Editar
Dirección http://150.128.82.83/telecontrolador/tutorial/
Ir Vínculos
Undergraduate Robotics Course
Mi PC

Figure VII-1. Robotics Tutorial home page


Undergraduate Robotics Course - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Adelante Detener Actualizar Inicio Búsqueda Favoritos Multimedia Historial Imprimir Editar
Dirección http://150.128.82.83/telecontrolador/tutorial/project/_index.html
Ir Vínculos
Voice
Stop
Inicio
Listo
Mi PC

Index

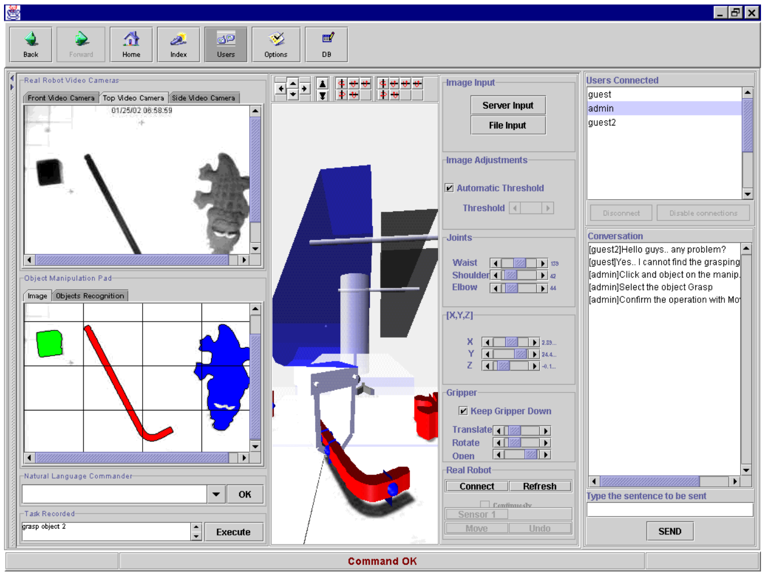
- Introduction
- Rotation Matrix
- RPY Representation
- Homogeneous Transformation Matrix
- Denhavit-Hartenberg Notation
- Direct K.P. and Invers K.P.
- Direct Kinematic Problem
- Reference System Assignment
- Exercises

Figure VII-2. Robotics Tutorial Index

As can be appreciated in Figure VII-1 and Figure VII-2, the students had online information about Kinematics, Reference System Assignment, and Robotics concepts in general. For more information, refer to the section "Education and Training" at Chapter II.

1.1.2. Online Teaching Assistance

While the students were accomplishing the exercises proposed in the laboratory and studying technical concepts of Robotics with the tutorial, they were able to interact with each other by means of a Chat capability. This has proved to be a very positive feature because it made students feel they belong to a group, even when they were interacting with the robot from home. In some situations, people asked the teacher questions using this capability. This kind of feature is commonly called "Collaborative Learning", which means students are learning not only from the teacher's lectures, but also from the collaboration with their partners in order to accomplish a given task.



grasp object 2

guest
admin
guest2

[guest2]Hello guys.. any problem?
[guest]Yes.. I cannot find the grasping
[admin]Click and object on the manip.
[admin]Select the object Grasp
[admin]Confirm the operation with Mo

Figure VII-3. Using the Online Teaching Assistance to learn how to grasp an object

As seen at Figure VII-3, the student called "guest" has been asking for help in order to grasp the allen key. The system administrator (teacher) responds saying that a simple way would be to click on the allen key from the manipulation pad and then to select the "Grasp" option.

1.1.3. Complementary Exercices

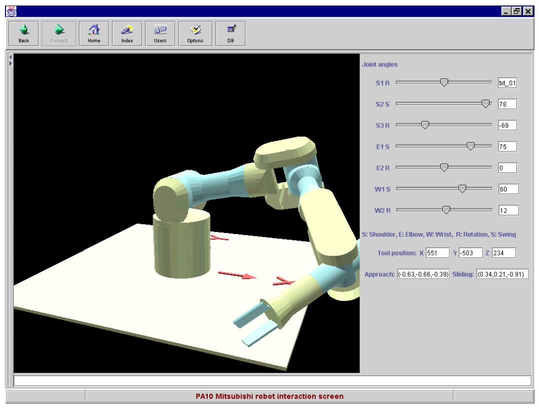
As introduced in Chapter II, in order to make students understand some basic concepts that are necessary for implementing a "Vision Guided Robotic System", some experiments about "Kinematics" and "Image Segmentation" have also been provided.

By using them, students have been able to understand which are the environment constraints (i.e. "Illumination", "bimodal images", etc.) that enable the proper segmentation of a camera image, and subsequently the automatic object recognition. For more information about this procedure, please refer to Chapter II.

1.1.4. 3D virtual scenarios for industrial manipulators

Before starting programming (at task-level) the educational Mentor robot, students were able to interact virtually with two industrial manipulators, the Mitsubishi PA10, and the AdeptOne Scara robot arm. They were able to move every joint separately, as well as using the Java3D virtual reality capabilities to navigate through the models by using the mouse input.

It was really interesting for them, due to the fact that these two manipulators really exist in the Intelligent Robotics Laboratory, and most of them have had the opportunity of watching these two robots in action.



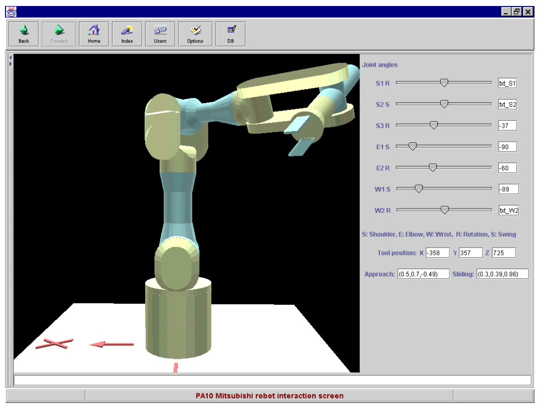
Joint angles

S1 R txt_S1
S2 S 78
S3 R -69
E1 S 75
E2 R 0
W1 S 60
W2 R 12

S: Shoulder, E: Elbow, W: Wrist, R: Rotation, S: Swing

Tool position: X 551 Y -503 Z 234

Approach: (-0.63,-0.66,-0.39) Sliding: (0.34,0.21,-0.91)



Joint angles

S1 R txt_S1
S2 S txt_S2
S3 R -37
E1 S -90
E2 R -60
W1 S -89
W2 R txt_W2

S: Shoulder, E: Elbow, W: Wrist, R: Rotation, S: Swing

Tool position: X -358 Y 357 Z 725

Approach: (0.5,0.7,-0.49) Sliding: (0.3,0.39,0.86)

Figure II-4. 3D virtual environment for the Mitsubishi industrial robot. Example of movements that can not be accomplished with the Mentor robot

As seen in Figure II-4, the Mitsubishi PA10 robot is able to accomplish movements that can not be performed by an educational robot like the Mentor, since the PA10 is a redundant robot with seven degrees of freedom.

1.1.5. The Educational Telerobot functionality (Pick & Place exercises).

Finally, once the students have been introduced to the system and Robotics in general by means of the above mentioned exercices, they had the chance to use the Telerobot functionality, with the aim of implementing some "Pick and Place" Robotics tasks on a virtual scenario, and then execute them on the real Mentor robot once it was available.

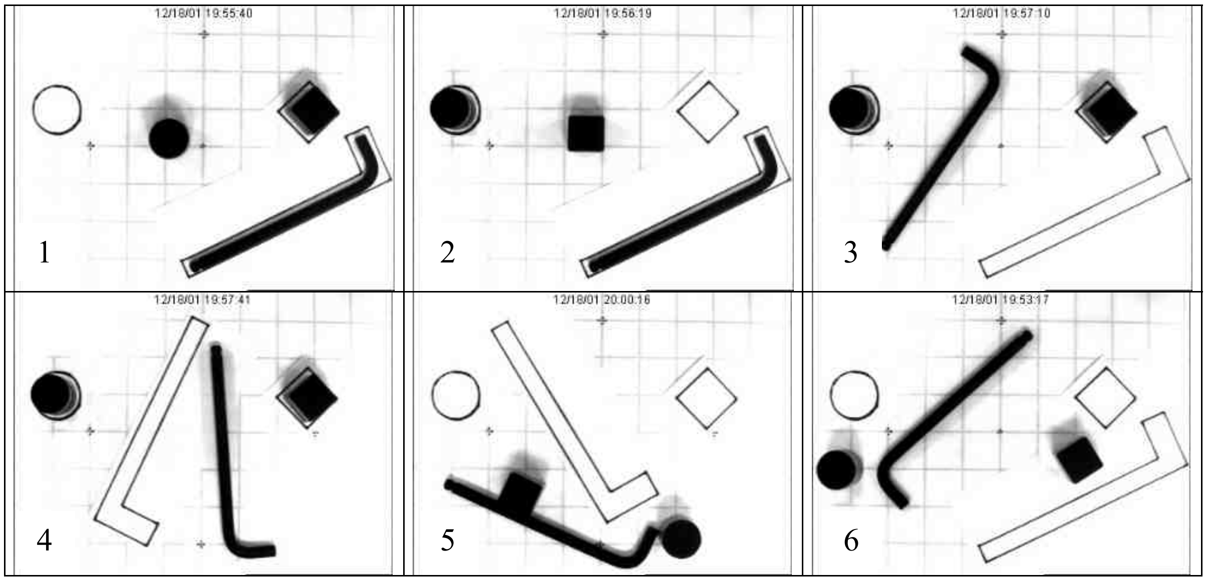


Figure VII-5. Example of Pick & Place exercises

Six "Pick & Place" exercises were proposed (see Figure VII-5), which consisted of placing the indicated objects onto their corresponding mouldings (outline of the objects). Exercises five and six were the most difficult to accomplish. For example, the fifth one presents an isolation problem that makes the objects appear to the system as just one. However, as the system proposes several grasping alternatives, whether or not the object is known, the students have been able to execute some of these graspings in order to separate the objects before the manipulation. Note this manipulation problem requires the user to provide most of the intelligence. The seventh example is very interesting because the allen key must be manipulated in two steps before it can be placed in the correct position.

Now we are going to explain in detail the steps the majority of the students followed in order to accomplish exercise four:

1. *Initialization*: First of all, the student selects the button “File Input” in order to incorporate the initial scene state in the virtual environment. As seen in Figure VII-6, the system automatically segments the scene, recognizes the objects, and calculates the set of grasping alternatives for every one of the objects. Note the grasping points are represented by two blue spheres joined by a line.

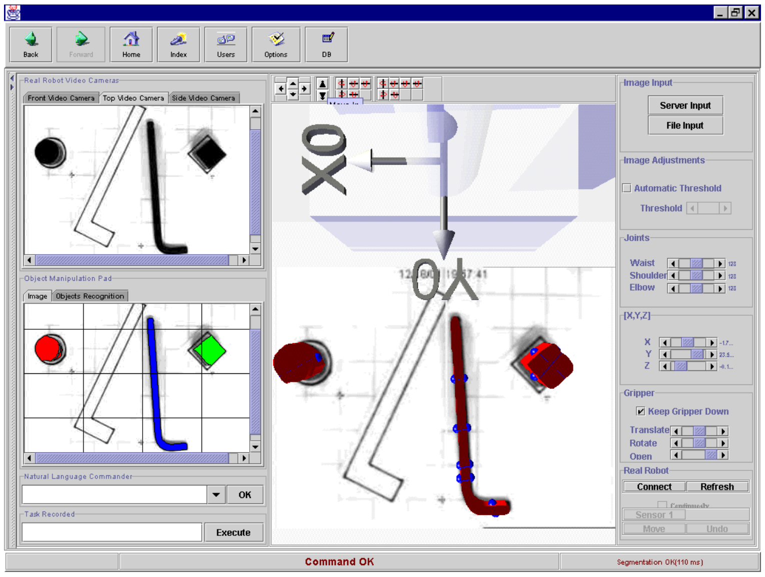


Figure VII-6. Initializing the virtual scenario by using the exercise four as initial state

2. *Commanding “Grasp the allen”*: Second step consists of commanding the “Grasp allen” task to the robot. It can be accomplished (as shown at Chapter II) in several ways. One could be clicking directly on the allen key from the Manipulation Pad and then selecting the “Grasp” option. Another possibility would be entering the command “Grasp allen” directly into the “Natural Language Commander”. The third alternative consists of using the “Speech Recognition” function and saying directly via the microphone the sentence “Grasp allen” (or “Take object one”). Then, the telerobotic system makes a prediction of the command before executing it with the real robot. If the user realizes this movement is not the one required, he can press the “Undo” button and the predictive display comes back to the original position (see Figure VII-7).

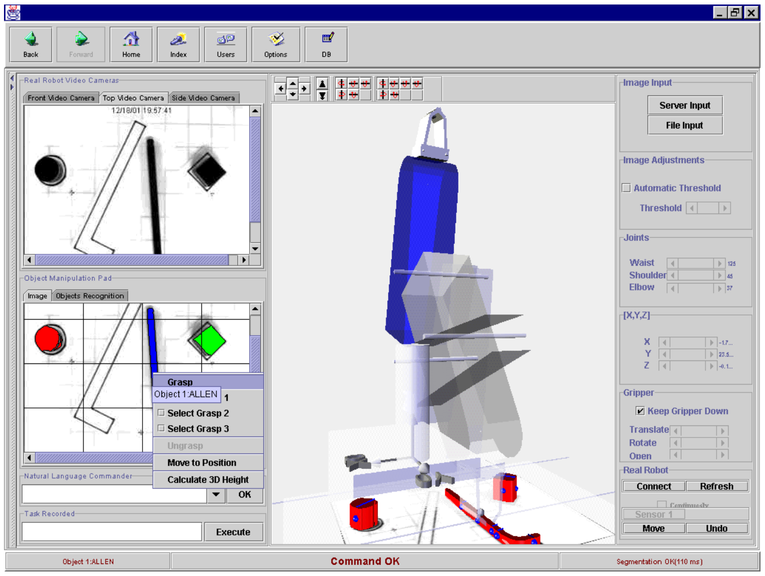


Figure VII-7. Selecting the Grasp Menu on the Manipulation Pad. The predictive display shows the operation on the transparent robot before sending the task to the real robot (or to the task specification file when working off-line)

3. *Confirming the operation*: If the user agrees with the manipulation action performed by the predictive display, the operation can be confirmed by pressing the "Move" button, or by saying "Confirm the operation" on the microphone. The result is shown at Figure VII-8.

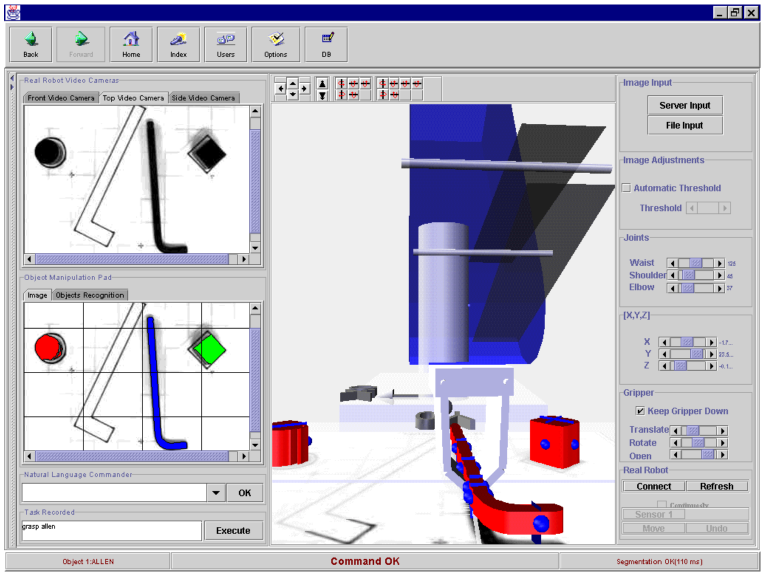


Figure VII-8. Confirming the "Grasp Allen" operation by pressing the "Move" button. The command has been recorded into the Task file and the opaque robot (real one) performs the operation.

4. *Selecting ungrasping point*: The following step consists of clicking the manipulation pad in order to specify the position where the object is going to be placed. Note we could elevate first the gripper a few centimeters in order to prevent the object colliding with the robot scenario. The result can be viewed at Figure VII-9. See how interesting is the top view in order to specify the exact point on the scene where the gripper must be placed.

See as well the importance of representing the projections of the gripper on the scenario.

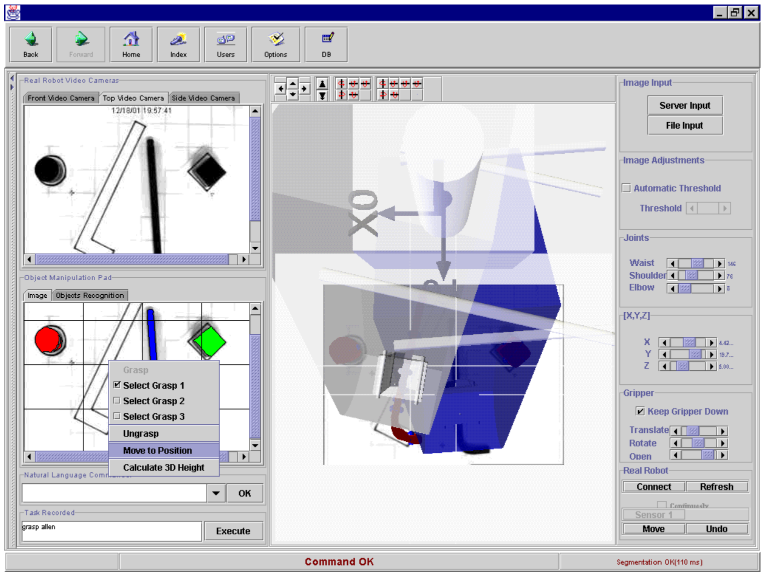


Figure VII-9. Selecting the point where the allen key is going to be placed.

5. *Confirming the operation*: Once the operation has been confirmed, the result can be viewed in Figure VII-10.

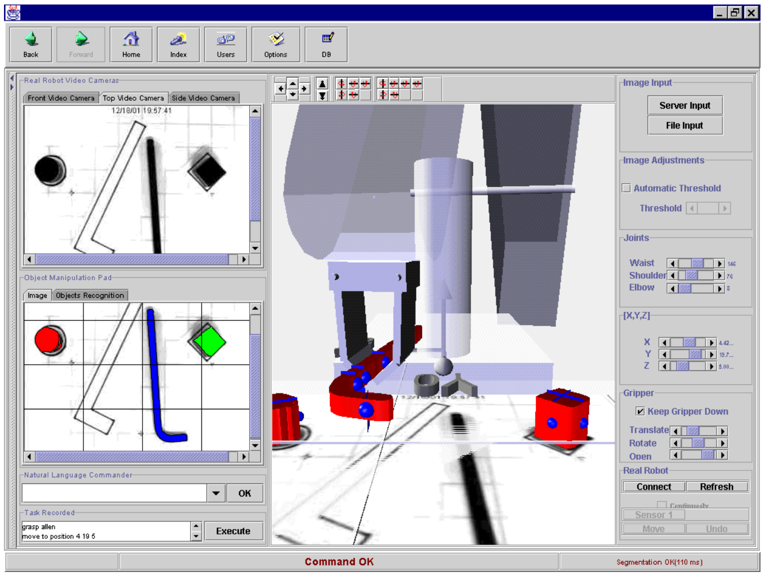
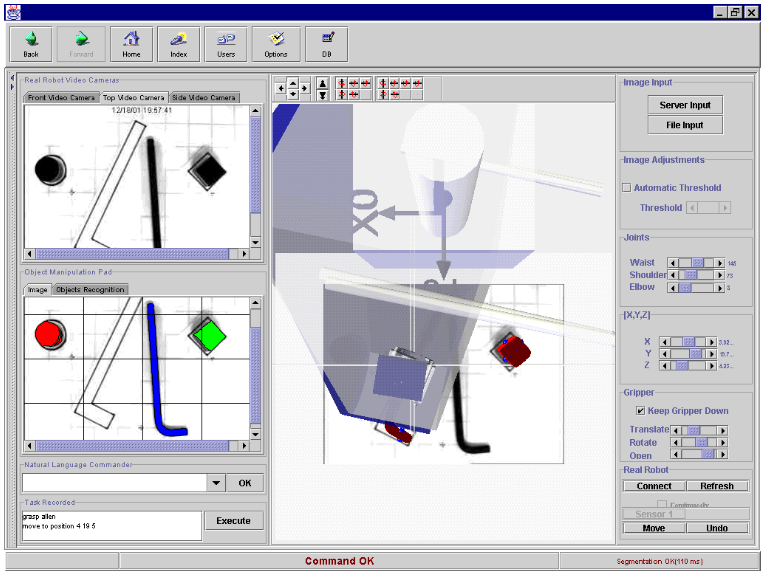


Figure VII-10. Confirming the operation of approximation before ungrasping.

6. *Rotating the Gripper:* Before ungrasping the allen we should rotate the gripper a little bit so that it can place the object on its mold with precision. To accomplish this, the 3D model top view is the most convenient for placing the allen precisely on the marked position (see Figure VII-11).



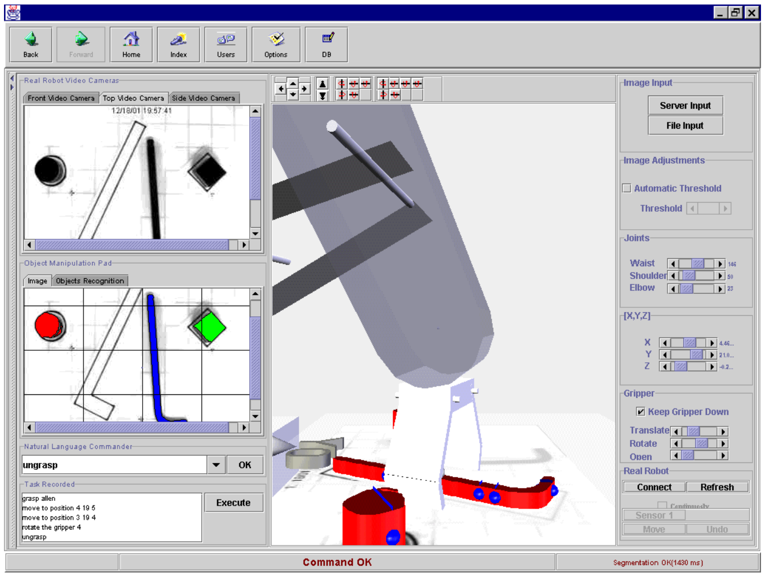
```
grasp allen
move to position 4 19 5
```

Command OK

Segmentation OK(110 ms)

Figure VII-11. Adjusting the gripper rotation before ungrasping the object

7. *Ungrasping the allen*: Then, by selecting the option "Ungrasp" on the manipulation pad or by simply typing or saying into the microphone the command "ungrasp", the manipulator lowwers the gripper and places the object on the correct position. Of course, confirmation is needed in order to allow the real robot or the task specification procedure to perform the action accordingly. See the result in Figure VII-12.



Natural Language Commander: ungrasp

```
grasp allen
move to position 4 19 5
move to position 3 19 4
rotate the gripper 4
ungrasp
```

Command OK

Segmentation OK(1430 ms)

Figure VII-12. Ungrasping the allen at the correct position

Finally the "Pick & Place" task for exercise four is shown in Figure VII-13, and has the characteristic of defining completely a robotic task in just five lines of code. See Chapter II for more details about the available robot commands, as well as the position independent commands.

```
grasp allen
move to position 4 19 5
move to position 3 19 4
rotate the gripper 4
ungrasp
```

Figure VII-13. Task specification file that solves the "Pick & Place" exercise four

Finally, the students were able to accomplish the 6 different classification exercises in less than 1 hour, which is a very good indication of how user-friendly and powerful is the robot interface. In Figure VII-14 the students can be seen interacting with the UJI Telerobotic Training System via Web.

Figure VII-14. The laboratory in action

1.2. Online Robots: Comparative Analysis

The second experience in the education and training domain consisted of showing to the students the existence of other three interesting web-based robots: "*The Telegarden*" [Telegarden][1], "*Ariti*" [Ariti][2], and "*The Australian Telerobot*" [Australian][3].

[1] [Telegarden] Telegarden webpage (http://www.usc.edu/dept/garden/).

[2] [Ariti] Augmented Reality Interface for Telerobot Application via Internet (http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/index.html).

[3] [Australian] Australian Telerobot webpage (http://telerobot.mech.uwa.edu.au/).

Eighty-seven students spent over 2 and a half hours interacting with these web robots. In the end, it was suggested they fill an anonymous questionnaire (see Figure VII-15).

Item description	Telegarden	Ariti	Australian	UJI
1. Is the user interface easy to understand?				
2. Is the object recognition capability useful?	-	-	-	
3. Are the robot commands of a high enough level?				
4. Is the graphical quality appropriate?				
5. Did you learn to move the robot rapidly?				
6. Does it allow the off-line interaction when robot is busy?				
7. Can you do interesting things when the robot is occupied?				
8. Can you program the robot and keep your actions for future execution?				
9. Would you use it again?				
10. Global assesment of the Web robot?				

Figure VII-15. Anonymous questionnaire to compare 4 different robots on web

Basically, as appreciated at Figure VII-15, the most significant questions are the last two: (1) Would you use it again?, and (2) Global assessment of the Web robot?. Finally, 87 students filled the questionnaire, and the results are presented at Figure VII-16.

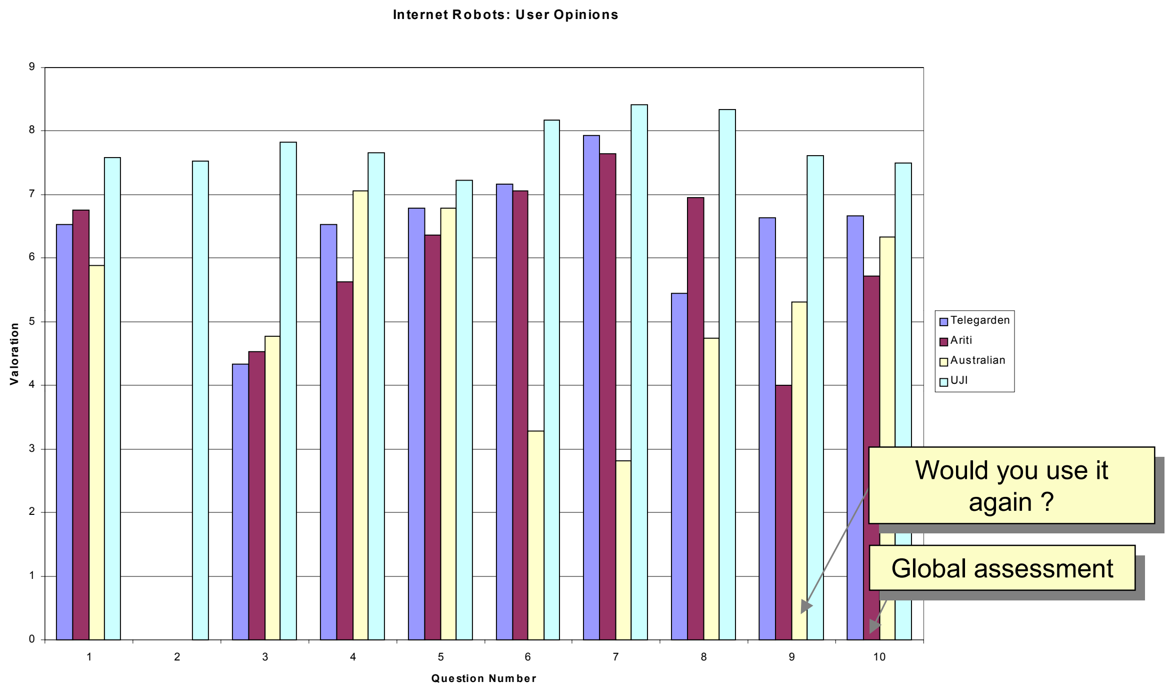


Figure VII-16. Students opinions of 4 different telerobotic system via web: Telegarden, Ariti, Australian, UJI. Eighty-seven students filled the questionaire.

As can be seen from results, students liked the UJI Telerobotic System very much. Some of them continued to use the robot from home in order to improve their abilities. They considered the Telegarden a very interesting robot too. They were highly motivated when planting and watering their own plants. However, some of them asked whether they were actually seeing a real robot action or not. It proves that users are interested in having real-time feedback of the whole robot scenario, which means monitoring not only the scene but the robot too. The Australian telerobot did not have this problem. Users realized the robot was actually moving by monitoring the camera inputs. The only aspect that they did not like was having to wait such a long time to control the robot. Including an off-line virtual scenario would be interesting for those situations.

2.The UJI Online Robot Open To The World

The UJI Telerobotic Training System has been online since August 2000. At the beginning it was used only for research purposes, and it was necessary to avoid the external connection to the system in order to elaborate experiments about "Object Recognition", "3D Virtual Environments", and so on. Then, on the 30th November 2001 it begun to be used at the Jaume I University as a tool for the students to learn Robotics, not only at the teaching laboratory, but also from home. Since then, the system has received almost six hundred connections from all over the world, which is in our opinion, an excellent rate, due to the fact that at the moment of writing this thesis, the robot has been available on the Web less than five months.

In this section we are going to present several statistics about the connection rates obtained by the UJI Telerobotic Training System since its origin.

2.1. How many connections?

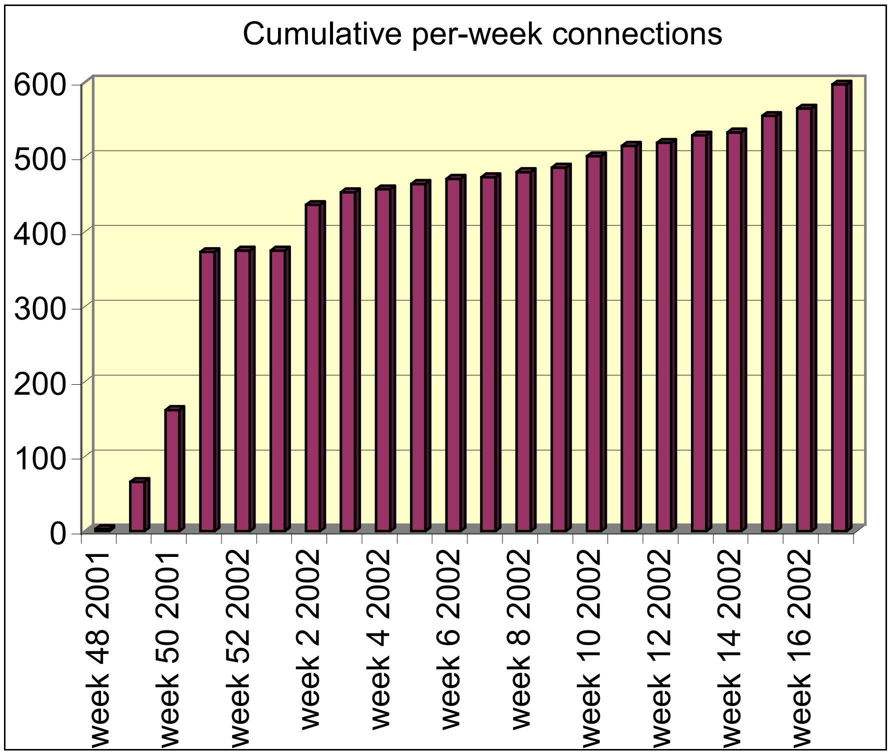


Figure VII-17. Cumulative per-week connections

As can be seen in Figure VII-17, at the moment of writing, the number of connections to the UJI Telerobotic Training System webpage were well over 500 hundred in less than five months. Moreover, once connected to the webpage some people did not log on the UJI Telerobotic Training System Java Applet. In fact, 85,25% of the users that visited the webpage decided to make use of the Java remote controller too. The rest (i.e. 14,75%) accessed the homepage and then exited (Something similar has occurred to the Australian Telerobot). In our opinion, there may be several reasons for this:

1. *Java 2 plugin installation*: As the UJI Telerobotic Training System requires the Java 2 runtime (and preferably the Java 1.3.x version), some people prefer not to install this package and they exit from the webpage.

2. *Java 3D plugin installation*: In order to use the 3D virtual environment functionality, the Java 3D interpreter must be installed on the client machine too. At this point, some people would prefer not to install it and, as in the case above, they go to another webpage without entering the Java applet. Moreover, the Java 3D runtime is only available (at the moment) for Windows based systems and Solaris machines. Of course this is an inconvenience for Linux and Mac users, but this drawback is expected to be resolved soon.

3. *Low bandwidth connections*: For those situations were people is accessing the Java applet using a very limited bandwidth device (e.g. 9,600 modem), they would have to spend quite a long time for the Java applet to be launched. Some of them would probably cancel the execution.

4. *Firewalled people*: Another possibility is having connections of people that are getting access to the Internet behind a firewall. It means they cannot access a TCP/IP port other than the 80 (webserver port). There is a work-around for this that at the moment has been discarded for performance reasons. It consists of tunneling every connection from the client to the servers by using the HTTP port. We have been trying this alternative for some time, and the overall system performance was reduced notably, so we decided not to use it.

However, for our situation, every student belonging to the UJI Robotics course has been able to use the system without problems, by means of the teaching laboratory computers, which means our former objective has been accomplished.

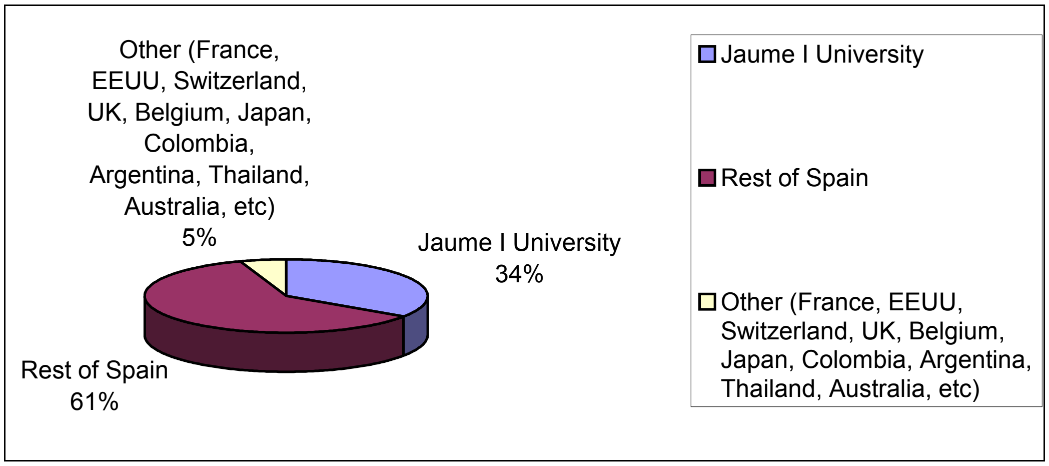


Figure VII-18. Connection rates by countries

As shown at Figure **VII-18** most of the people connected to the webpage are from Spain, and specially from the Jaume I University (34%). This is so because of our students, who at one time (i.e. 20th December 2001) made a total of 138 web connections. Moreover, people from other Universities around the world have been interested about the project, so the UJI Telerobotic Training System has received connections from other countries too (i.e. France, Switzerland, Japan, Colombia, Argentina, Thailand, EE.UU., Belgium and others).

2.2. With What?

Another interesting point is knowing which kind of browser and operating systems pleople are using from the world wide web when accessing the UJI Telerobotic Training System web-page. In fact, this information can be utilized in order to better test the system response depending on the current client configuration.

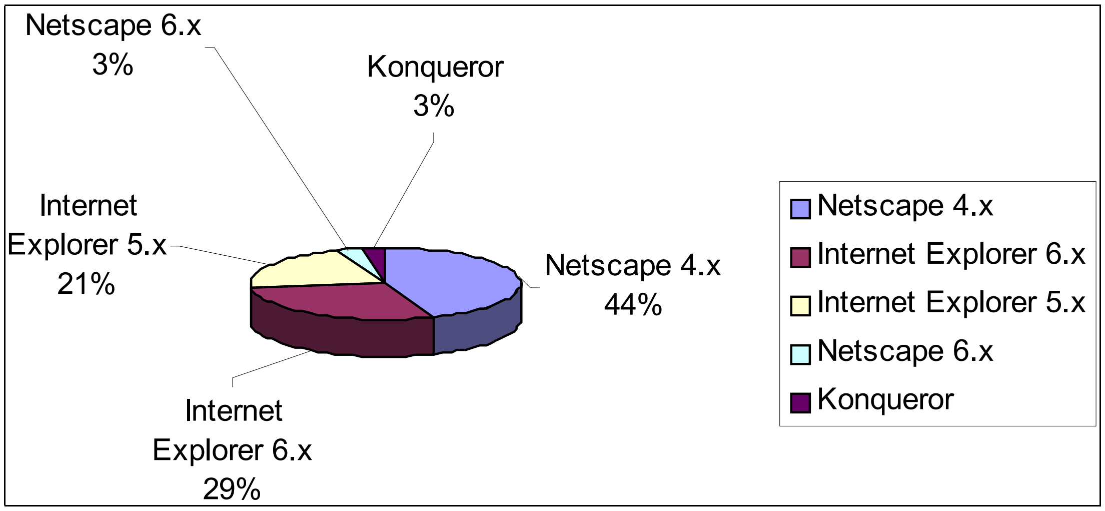


Figure VII-19. Browser statistics

As shown in Figure VII-19, most of the people are still using the Netscape 4.x browser, which means they need to install the Java 2 runtime in order to execute the Java applet properly. Moreover, as many users are migrating to the new versions (IE 6.x and Netscape 6.x), most of them will not require installing such a plugin. The only library necessary for them would be the Java 3D runtime.

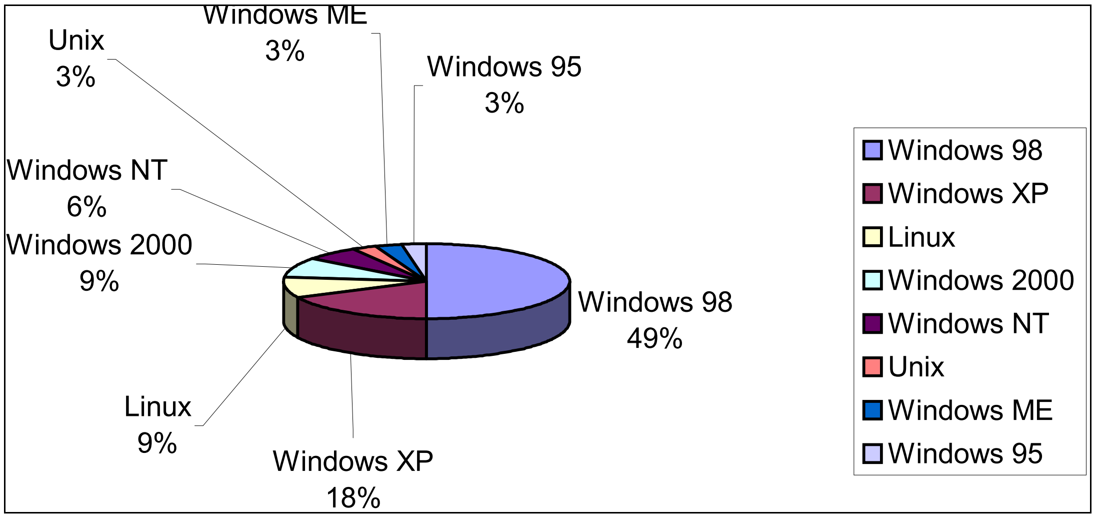


Figure VII-20. Operating System statistics

The operating systems used by the majority of people connected to the website are Windows based. However, 11% are working on Unix and Linux platforms. For those who work with Linux, the Java3D library has not been launched yet, which means they can access the Java applet but they are not able to interact with the 3D virtual environment. On the other hand, those who work under Solaris (Sun Microsystems) have the whole set of libraries necessary and can, of course, interact with the Telerobotic System through the web.

More information about statistics can be found by pressing the "Statistics" button at the UJI Telerobotic Training System website.

3.Summary

The present chapter has described how people have been using the UJI Telerobotic Training System in order to accomplishing manipulation of objects via web. In particular, the students at the Robotics course in our university have been implementing "Pick & Place" operations in a controlled environment in order to facilitate the testing and validation of the system. In fact, they have interacted not only with the UJI Telerobotic Training System, but also with other Online Robots like "Australian Telerobot", "Telegarden", "Ariti", etc. Finally, they have given their opinion about these systems and the comparative results have been shown in this chapter, which we have found very encouraging especially with a view to improving the systems capabilities.

It has been demonstrated that the system works very well in a structured environment applied to the Education and Training field. Moreover, with this experience it has been possible to better test the system performance and robustness, which is very important if we intend to extend the current technology to others fields like mobile manipulation in partially structured environments.

We have been able to extract some interesting conclusions:

1. The use of this kind of tool has shown that it represents a very useful complement to the conventional teaching (lectures) employed in the classroom. Moreover, the normal use of this kind of systems by the student improves their autodidactic capabilities.

2. At the same time, the lecturer response has been good , and all of them have accepted the idea of extending the content of the tutorial to other important parts of the robotics subject.

3. Finally, the extensive use of the web, permits easy access to the system without space-temporal constraints, and opens the possibility of distance teaching-learning in many other fields.

The future activities will be oriented towards using a professional robot instead of an educational one. Besides this, several experiments will be carried out in order to improve the training capabilities of the system and the Human-Computer interaction.

The last part of the chapter focuses on the Connectivity rates, which shows the way people have been using the Web in order to access the Telerobotic System. A total of almost six hundred connections to the system have been reported in less than five months. Just note we have not spent any time to register the webpage into any Web searcher and so on, due to the fact that we preferred having smaller amount of people connected, in order to better test and validate the system. It means we expect many people to access the system in a near future, as soon as the scientific community and the society in general gets the opportunity to know the existence of the UJI Online Robot Project.

References

[Ariti] Augmented Reality Interface for Telerobot Application via Internet (http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/index.html).

[Australian] Australian Telerobot webpage (http://telerobot.mech.uwa.edu.au/).

[Backes et al., 2001] P. G. Backes, K. S. Tso, J. S. Norris, and G. K. Tharp. *Internet-based Ground Operations for Mars Lander and Rover Missions,* K. Goldberg, Roland Siegward, Beyond Web Cams: An introduction to Online Robots, MIT Press, Massachusetts, 2001.

[Marín et al., 2001a] R. Marín, P.J. Sanz. *Telerobotic Training System through the web: Low level Architecture.* In 1st IFAC Conference on Telematics Applications in Automation and Robotics, K. Schilling & H. Roth, IFAC Publications, Elsevier Science Ltd., Weingarthen, Germany, 2001.

[Marín et al., 2001b] R. Marín, P.J. Sanz. *The UJI Telerobotic Training System. 1st EURON Workshop on Robotics Education and Training (RET2001),* Alicia Casals, Antoni Grau, Grup Artyplan-Artymprès, S.A., Weingarthen, Germany, 2001.

[Marín et al., 2001c] R. Marín, J.S. Sanchez, P.J. Sanz. *Design and Evaluation of a Telerobotic System with Object Recognition Capabilities,* Applied Informatics-Artificial Intelligence and Applications; Advances in Computer Applications, M.H. Hamza, IASTED International Conference on Robotics and Applications (RA 2001), Florida (USA), 2001.

[Marín et al., 2002] R. Marín, J.S. Sanchez, P.J. Sanz., *A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality.* In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Marín et al., 2002b] R. Marín, J.S. Sanchez, P.J. Sanz. *Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System.* In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Marín et al., 2002c] R. Marín, J.S. Sanchez, P.J. Sanz. *Teleoperated Robot System Via Web: The UJI Telerobotic Training System.* accepted for publication in the Special Issue on Webbased Robotics and Automation of the International Journal of Robotics and Automation, Khaled Elleithy & Tarek Sobh, Actapress, 2002.

[Marín et al., 2002d] R. Marín, P.J. Sanz. *Augmented Reality to Teleoperate or Robot through the Web.* In 15th IFAC World Congress b'02, Barcelona (Spain), 2002.

[Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

[Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Telegarden] Telegarden webpage (http://www.usc.edu/dept/garden/).

[Sanz et al., 1998] Sanz P.J., Adell S. An undergraduate Robotics Course via Web. Teleteaching'98 Conference, a part of the 15th IFIP World Computer Congress. Distance Learning, Training, and Education. Austrian Computer Society (book series of). Edit. by Gordon Davies, pp. 859-867, Viena, Austria. 1998.

Chapter VIII **CONCLUSIONS**

The present work has produced many objectives, and some of them are considered novel research contributions to the online robotics field. For example, as confirmed at ICRA'2002 [Marín et al., 2002a][Marín et al., 2002b], the object recognition and grasp determination modules are new ways to accomplish the design of a web-based telemanipulation system. They allow a higher level of interaction between the human and the robot, which means the user can concentrate on the task supervision instead of controlling every robot movement. Besides this, the augmented and virtual reality possibilities offered by the 3D modeling have not yet been presented in any other web based system. In fact, as presented in chapter III, the 3D representation allows the prediction of a task before sending it to the real robot, which overcomes the problems of delay and bandwidth associated to the Internet.

This chapter is organized in three main sections: The first one summarizes the overall capabilities of the project and its novel contributions to Telerobotics and online robots in particular. The second one focuses on different online robots that are currently working in the Web, and then presents a comparative analysis of their capabilities focusing on the user interface function. After that, the third section enumerates the research areas which the results of this Ph.D dissertation have suggested, and which will be investigated in future in our further work.

Topics:

1. Novel contributions
2. Online Robots: Comparative Analysis
3. Further work
4. Possible Applications

Anyone who has never made a mistake has never tried anything new (Albert Einstein).

1.Novel contributions

In summary, a whole vision-based robotic system to control an educational robot via web has been designed and implemented. The user is provided with multiple ways to operate the robot: (1) by moving the joints directly, (2) by moving the robot in world coordinates, (3) by using the mouse over the camera scene and selecting directly the required operation (i.e. “Grasp”, “Ungrasp”, “Move To”, etc.), (4) commanding text-based sentences (e.g. “Grasp the allen”), and (5) by using voice commands like "Grasp object one" or "Grasp the cube".

As the user is able to decide the way he (or she) can program the robot (“adjustable interaction level”), it means the system can be controlled from a low-level teleoperated manner to a more supervised way. Hence, for those situations where the robot scenario is difficult to model automatically due to the fact that the environment is not structured enough, the user can take the absolute control of the robot and perform the operations by moving directly the joints (teleoperated manner). On the other hand, when the user detects that the robot is intelligent enough to carry on a “Pick & Place” operation (scene properly segmented, proper illumination, etc.), the interaction way can be performed at the highest level (i.e. saying “Grasp [object]” into the microphone). In this situation the user programs the robot in a rather supervised and natural manner.

One of the major difficulties associated with the design of online robots is the Internet latency and time delay. When using the Internet as connectivity medium, it cannot be assured that a certain bandwidth is going to be available at a given moment. It means that once the information is sent from the server to the client (and vice versa), there are no guarantees that this data will reach its destiny in a minimum period of time. The delay associated with the Internet communications is unpredictable, so that the system design must take care of this situation, by means of giving more intelligence to the server side and using “predictive display” techniques.

Hence, the user interface has been designed to allow the operator to "predict" the robot movements, before sending the programmed commands to the real robot ("Predictive system"). This kind of interface has the special feature of saving network bandwidth and even being used as a whole task specification, off-line, programming, interface. By using a predictive virtual environment and giving more intelligence to the robot means a higher level of interaction, which avoids the “cognitive fatigue” associated with many teleoperated systems, and helps enormously to diminish the Internet latency and time delay effects.

Such a complex and friendly user interface would not be possible without the help of virtual and augmented reality techniques. With these two technologies the user is able to control almost every viewpoint of the real robot scenario, and then simulate the programmed tasks before sending it to the real manipulator. Moreover, automatic object recognition and autonomous grasping execution are crucial in order to provide such a level of user interaction.

At this point we are going to summarize the novel contributions that the current thesis offers to the scientific community in the web Telerobotics domain. As discussed in next section, some of these capabilities have never been used before, to the best of our knowledge, in the online robotis domain (web-based teleoperated systems):

1. *Automatic Object Recognition*: The first novel contribution that has not been reported yet in any other web-based telerobotic system is the ability to recognize different objects from a camera input. This capability is fundamental in order to allow the user to specify high level commands like "Grasp the allen". In addition, this object recognition module can be used for other robots as long as the camera view is from the top of the scene (multirobot configuration).

2. *Incremental Learning*: The object recognition procedure requires some kind of training before performing properly. The UJI Telerobotic Training System introduces the Incremental Learning capability that means that the robot is always learning from user interaction. It means the robot is becoming more “intelligent” as time goes by.

3. *Autonomous Grasping*: Once an object has been recognized in a scene, the following question is, how can we grasp it? The autonomous grasping module calculates the set possible grasping points that can be used in order to manipulate an object conforming to the stability requirements. By default, it is the robot which has the intelligence to know how to manipulate an object. This ability is a novel contribution that enables the system to obey commands like "Grasp the object 1". It means that, in some situations, the Autonomous Grasping process is intelligent enough to perform a proper "Pick & Place" interaction. On the other hand, if the user thinks that the grasping alternative offered by the robot is not the best one, he (or she) can select among three different options (if available).

4. *Non-Immersive Virtual Reality*: In order to avoid the Internet latency and time-delay effects, the system offers to the user an interface based on non-immersive virtual reality. Hence, taking the camera data as input, a 3D virtual reality scenario is constructed, which allows the specification of tasks that can be confirmed to the real robot in one step. Besides this, by using such a technology the user is able to monitor the scenario from any viewpoint, which is enormously helpful to the user making remote operated manipulation.

5. *Augmented Reality*: The 3D virtual scenario is complemented with computer generated information that helps very much to improve the human performance (e.g. projections of the gripper over the scene is shown, superposition of data in order to avoid robot occlusions, etc.). In some situations the user has more information by controlling the robot from the user interface (web based) than seeing the robot scenario directly.

6. *Task specification*: The system permits specifying complete “Pick & Place” actions using both, the on-line and the off-line modes. Then, the whole task specification can be saved in a text file for later execution.

7. *Voice recognition*: This novel contribution means that the robot can be controlled by simply using a microphone and headphones. In chapter VI many details are given on the architecture and the design that has been followed in order to accomplish such functionality.

As explained at Chapter I, more than fifteen scientific publications have been derived from this work. In fact, preliminary results of this thesis have been already presented at several conferences and in journals. The most important is for example the presentation of two papers at the *IEEE International Congress on Robotics and Automation 2002,* and the publication of an extended article at the *Special Issue on web telerobotics of the International Journal on Robotics and Automation* (November 2002).

2.Online Robots: Comparative Analysis

A Telerobotic System consists of 3 main parts: (1) the remote robot, (2) the communication network, and (3) the User Interface. In our situation, in order to compare the UJI Telerobotic Training System with other online robots, we are going to focus on the user interface, which is our main interest area. Note the remote robot depends on the current application, and the communication network is always similar (the Internet).

First of all we are going to enumerate the parameters that, in our opinion, will make clear the elaboration of the comparative analysis:

1. *Internet based versus Web based*: Although every Telerobotic system included in the comparative analysis uses the Internet as the connection medium, the ones that are really interesting for us are the online robots, which means their remote control can be accomplished using the World Wide Web. For a list of the advantages of using the Web refer to Chapter I.

2. *Predictive display*: The use of predictive displays helps to diminish the Internet latency and delay effects. Besides this, it can help to the implementation of virtual task specification environments.

3. *Augmented reality*: By using Augmented reality techniques the user information on the client side is enhanced enormously. It obviously helps the robot programming and monitoring.

4. *Virtual reality (3D model)*: By using 3D-model construction from camera input the user is able to interact with the system from any point of view. Besides this, it is a very useful technique for implementing the predictive display functionality and the task specification environment.

5. *Automatic Object Recognition*: In order to specify tasks in a more natural manner, the automatic object recognition module is very helpful in those situations were the environment is structured enough to perform the operation in a reasonable time. Moreover, by defining robotic tasks by using object names instead of object numbers implies the operations are going to be invariant to the initial position, which enhances a lot the system performance and robustness.

6. *Incremental Learning:* In our opinion a system should have the ability to *learn* from experience. It means the system is becoming more intelligent as time goes by. This learning feature should at least enhance the Automatic Object Recognition system performance.

7. *Autonomous Grasping*: One of the fundamental key modules in order to enable high level interaction is the Autonomous Grasping. By default, the Telerobotic system should calculate the best grasping alternatives for every object in the scene.

8. *Very high level commands*: In order to avoid the "cognitive fatigue" of the user when manipulating a scenario remotely, the idea is to allow the specification of very high level commands. It means the user is able to operate the robot in a more supervised manner.

9. *Voice Recognition*: In our opinion it would be very interesting if the system allows the user to make these high level commands using a microphone. This means that the user is able to interact with the remote robot at a distance using the voice as the only input.

10. *Off-line Programming*: Another important situation to take into account when implementing an online robot is what to do when someone else is using the real manipulator. The Off-line programming technique allows the user to program the robot in a simulated environment, without needing to have physical control of the manipulator. It is very important in order to let people (e.g. students) work with the robot as long as possible.

11. *Tasks Specification*: In our opinion it would be interesting to have a way of specifying tasks that have previously been recorded from another interaction. This capability could be used for increasing the robot possibilities (task learning).

12. *Multiple Users Management*: Of course, an online robot should implement some criteria in order to allow multiple users to share the robot control. For example, the FIFO (First In First Out) alternative could be a way of doing it.

At Figure VIII-1 a comparative study between several web telerobotics systems is presented. Some of this information has been acquired by asking the authors directly (i.e. K. Goldberg, etc.). Others have been obtained by revising proceedings, books, research magazines and by using them directly via Web.

	Mercury Project	Telegarden	Australian Telerobot	Ariti	Columbia University	Tenerife University	VISIT	UJI
Internet based	●	●	●	●	●	●	●	●
Web based	●	●	●	●		●		●
Predictive Display					●		●	●
Augmented Reality			●	●				●
Virtual reality (3D model)				●	●	●		●
Automatic Object Recognition					●			●
Object Recognition under occlusions					●			
Incremental Learning								●
Autonomous Grasping								●
Very high level commands								●
Voice Recognition								●
Off-line Programming				●				●
Tasks Specification				●			●	●
Multiple users management	●	●	●	●		●		●

Figure VIII-1. Comparative study of different online robots

3.Further work

Internet-based Telerobotics, and particularly online robots, have the advantage of allowing the robot control from almost anywhere. It means expert people around the world can collaborate to accomplish a difficult teleoperated task without having to move from home. Of course, using a public network introduces some drawbacks that must be taken into account, like for example, security strategies, and network latency. For example, the Web Interface for Telescience (WITS)[1] provides the ability to many researches around the world to program the ground operations for planetary lander and rover missions. Collaboration by geographically distributed scientists at their home institutions enables participation in missions by a greater number of scientists and reduces operation costs [Backes et al., 2001][2].

Future research is going to be focused on Telerobotics applied to Service Robotics, which is basically the area of interest of the Intelligent Robotics Lab at the Jaume I University of Castellón (Spain). In fact, at this moment two important research projects in this area are being performed, which are:

[1] (http://mars.graham.com/mpfwits/)

[2] [Backes et al., 2001] P. G. Backes, K. S. Tso, J. S. Norris, and G. K. Tharp. *Internet-based Ground Operations for Mars Lander and Rover Missions,* K. Goldberg, Roland Siegward, Beyond Web Cams: An introduction to Online Robots, MIT Press, Massachusetts, 2001.

- Project funded by the *Generalitat Valenciana*, under the grant No. (GV01-244: "*Design and implementation of a Robotics system for the manipulation of objects in dinamic 3D scenarios*". Leader: Pedro José Sanz Valero.
- Project funded by the Spanish CICYT (Department of Science and Technology): "*Design of Service Tasks for a Mobile Robot Manipulator*". Leader: Ángel Pasqual del Pobil.

In particular, the next step will consist of studying and implementing some work related to the *Remote Visual Servoing* area. In fact, the author has got funds from Bancaixa Foundation in order to do a placement in the LASMEA Laboratory, at the Blaise Pascal University of Clermont-Ferrant (France), where collaboration on this subject is going to be accomplished. This functionality is very important in order to apply the current results to more challenging applications like Telemanipulation for Mobile Robotics, and so on.

The other improvements that we are already developing are the integration of two industrial robots (Mitsubishi PA10 and AdeptOne) into The UJI Telerobotic Training System. By doing this, researchers and students (under security restrictions) will be able to program high level tasks for these robots, not only in off-line mode, but also on-line.

4.Possible Applications

The major advantage of the telerobotic systems via Web is that the expert user has the possibility of controlling the robot from almost anywhere. On the other hand, by using the Web as communication medium we must take care of the Internet latency and delays. Due to the fact that the Internet is continuously evolving in robustness, velocity and bandwidth, the Robotic applications that we will be able to accomplish via Web will be more important as time goes by. The user will have more information from the real scenario (force sensor, stereoscopic vision, etc.), so it means the remote control will be more accurate, faster, and more secure. In fact, when in 1997 we had the idea of designing a system to manipulate objects via Web, we already knew that 9600 bps were not sufficient for our purpose, but we expected the Internet bandwidth would increase enormously before the project would be completed. Nowadays, the broadband connections (i.e. ADSL, etc.) are being used normally for many people in order to get connection to the Internet from their homes. It makes us think that in a near future the Robotic applications that we will be able to design via Web are going to be very interesting.

It should be noted that The UJI Online Robot System can be executed in two modes, as a Java Applet launched from a Web page (by default), and as an application that is already installed into the client computer and can be run directly (without recompiling) over a private network that offers more performance (i.e. delays, bandwith, etc.). This characteristic would be very interesting for those situations where the use of a public network (e.g. Web) is not the best option (i.e. Medical applications, etc.).

Hence, we have made a classification of the different applications that this project could have in different areas. Some of them has been already been proved (i.e. Education and Training), and others could be performed in the mid-term. It depends basically on the bandwidth related to the communication network, and the human-robot interaction devices can be used.

1. **Areas where the project has already been tested:**

    1.1. *Education and Training:* First of all, as we have seen in Chapter VII, The UJI Online Robot has been tested with success in the Education and Training domain. In particular, students from the Jaume I University were using this system in their Robotics Lab sessions and were able to manipulate objects on a board by programming high level tasks. Students have had access to the Robotics Lab (via Web) twenty-four hours a day during his training period.

    1.2. *Entertainment:* Once the UJI Online Robot webpage was launched we realized many people were getting access to the robot for entertainment. For them, the possibility of controlling a robot via Web is motivating. In fact, some people have been accomplishing manipulation tasks with the robot during several hours.

2. **Areas where the project could be applied in the mid-term:**

    2.1. *Security and Surveillance:* By enhancing the system in such a way that allows the manipulator to be located on a mobile platform, the project could be used to give support to the security and surveillance services inside a building. We could for example attach a camera to the robot arm and program the system to detect anomalous situations (e.g. movement of people out of schedule, etc.) and to raise an alarm that would be received by the surveillance staff in a terminal. One of the advantages of such a system would be the possibility to manipulate, which means the robot would be able to open doors, etc.

    2.2. *Service Robotics:* Considering the mobile configuration explained above, we could make an effort and provide the robot with the abilities to locate targets inside the building and allow people via Web to specify high level tasks (e.g. "Go to Pedro's office and bring the scissors back"). In particular, a similar application is being designed in order to command the robot to bring a book from the University Library. The robot must be able to move to the Library Building avoiding collisions, find the book inside the building, manipulate it, and bring it back. Of course, one of the most interesting problems is allowing the robot to recognize the book in the shelves, by means of its label.

3. **Areas where the project could be applied in the long-term:**

    3.1. *Demining:* A very interesting problem would be using a remote teleoperated system (perhaps via Web) in order to manipulate Landmines and disable them. In fact, some conversations have been initiated in order to analyze the feasibility of adapting and enhancing the UJI Online Robot system in order participate into the EUREKA program, by means of the project called "Advanced Global System To Eliminate Anti-Personnel Landmines (Apl)".

    3.2. *Space Robotics:* Taking into account that The UJI Online Robot system is able to control a robot through a communication link that has delays (i.e. the Web), an extension of this project could be perhaps been performed in order to use the system in environments where the communication delay is even bigger, the Space Robotics. Moreover, as explained before, there are some situations where the use of the Internet to specify robotic tasks in a collaborative manner could be very convenient (e.g. NASA's WITS Project).

3.3. Medical Robotics: Virtual Reality and Augmented Reality techniques are becoming very useful in order to implement robust systems that enable a non-invasive surgery. Although this kind of applications need a communication link very secure, robust and with a high level bandwidth, we must take into account that the Internet is evolving very rapidly, and possibly in some years fully immersive teleoperated systems could be implemented in order to give support for surgery operations. The advantage of the Internet as communication medium is that the expert can get access to the system from almost anywhere in the world. Although this can appear to us like fiction nowadays, in some decades such a system could become a reality.

References

[Ariti] Augmented Reality Interface for Telerobot Application via Internet (http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/index.html).

[Australian] Australian Telerobot webpage (http://telerobot.mech.uwa.edu.au/).

[Backes et al., 2001] P. G. Backes, K. S. Tso, J. S. Norris, and G. K. Tharp. *Internet-based Ground Operations for Mars Lander and Rover Missions,* K. Goldberg, Roland Siegward, Beyond Web Cams: An introduction to Online Robots, MIT Press, Massachusetts, 2001.

[Marín et al., 2001a] R. Marín, P.J. Sanz. *Telerobotic Training System through the web: Low level Architecture.* In 1st IFAC Conference on Telematics Applications in Automation and Robotics, K. Schilling & H. Roth, IFAC Publications, Elsevier Science Ltd., Weingarthen, Germany, 2001.

[Marín et al., 2001b] R. Marín, P.J. Sanz. *The UJI Telerobotic Training System. 1st EURON Workshop on Robotics Education and Training (RET2001),* Alicia Casals, Antoni Grau, Grup Artyplan-Artymprès, S.A., Weingarthen, Germany, 2001.

[Marín et al., 2001c] R. Marín, J.S. Sanchez, P.J. Sanz. *Design and Evaluation of a Telerobotic System with Object Recognition Capabilities,* Applied Informatics-Artificial Intelligence and Applications; Advances in Computer Applications, M.H. Hamza, IASTED International Conference on Robotics and Applications (RA 2001), Florida (USA), 2001.

[Marín et al., 2002] R. Marín, J.S. Sanchez, P.J. Sanz., *A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality.* In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Marín et al., 2002b] R. Marín, J.S. Sanchez, P.J. Sanz. *Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System.* In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Marín et al., 2002c] R. Marín, J.S. Sanchez, P.J. Sanz. *Teleoperated Robot System Via Web: The UJI Telerobotic Training System.* accepted for publication in the Special Issue on Webbased Robotics and Automation of the International Journal of Robotics and Automation, Khaled Elleithy & Tarek Sobh, Actapress, 2002.

[Marín et al., 2002d] R. Marín, P.J. Sanz. *Augmented Reality to Teleoperate or Robot through the Web.* In 15th IFAC World Congress b'02, Barcelona (Spain), 2002.

[Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

[Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Telegarden] Telegarden webpage (http://www.usc.edu/dept/garden/).

Part III

Part III: Appendix

Appendix I TECHNICAL SPECIFICATIONS

1.MENTOR ROBOT

The Mentor robot is an inexpensive educational manipulator distributed by ALECOP[1] that has the special characteristic of being very robust to collisions, which is an advantage when using it in an educational environment, and specifically remotely controlled through the web. The principal drawbacks are two: (1) the robot joints must be calibrated, since they are programmed in steps, and (2) the robot work area is very limited, so that the robot cannot access some specific points in the work area. The robot work area is shown at Figure 1.

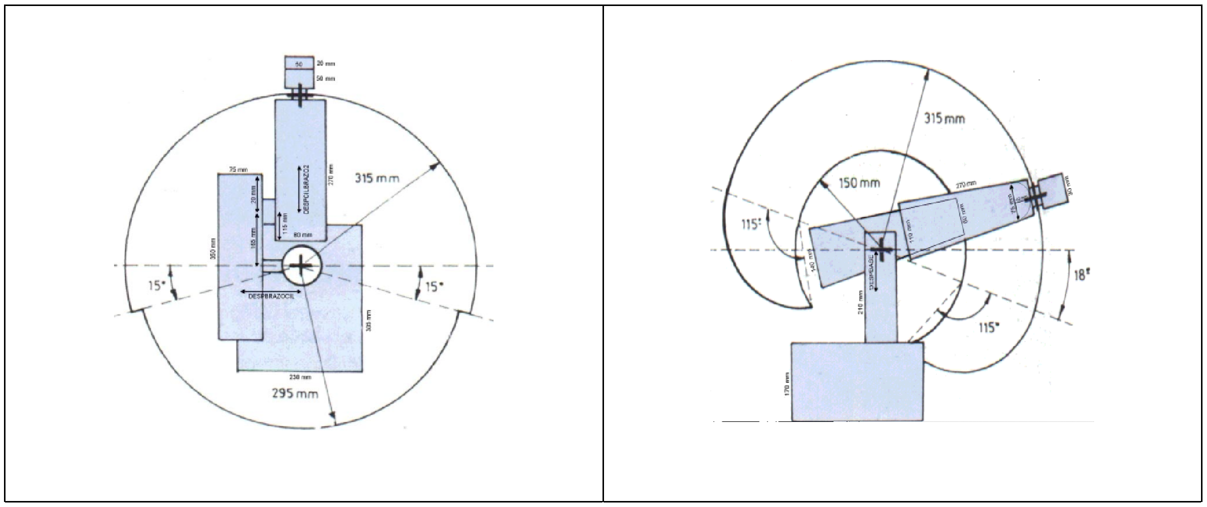


Figure 1. Mentor Work Area

The Mentor technical information is shown in Figure 2, where can be seen for example all the characteristics of the joints, robot velocity, and sensor connectivity. The major advantage of using the Mentor robot is its excellent robustness, which makes it ideal in an Education and Training domain, and specially for using it in a teleoperated system. Its drawbacks are fundamentally its repeatability, which, as explained in the technical documentation, is of 2mm.

[1] (http://www.alecop.com)

Joint 0 mobility (waist)	210º
Joint 1 mobility (shoulder)	180º
Joint 2 mobility (elbow)	230º
Joint 4 mobility (wrist elevation)	140º
Joint 5 mobility (wrist rotation)	320º
Joint 6 mobility (gripper opening)	45 mm
Repeatability	2 mm
Elevation capacity	1 Kg
Maximum velocity	46º / seg
Reach (from axes 1 center)	420 mm
Analog input	0-5 Vcc
Digital input	TTL (5 Vcc)
Digital output	TTL (5 Vcc / 20 mA)
Power supply	220 Vca / 50-60 Hz

Figure 2. Robot Characteristics

2.ADEPTONE ROBOT

As defined at (ISO 10218:1992(E)), a Manipulating Industrial Robot is an automatically controlled, reprogrammable, multi-purpose, manipulative machine with several degrees of freedom, which may be either fixed in a place or mobile for use in industrial automation applications.

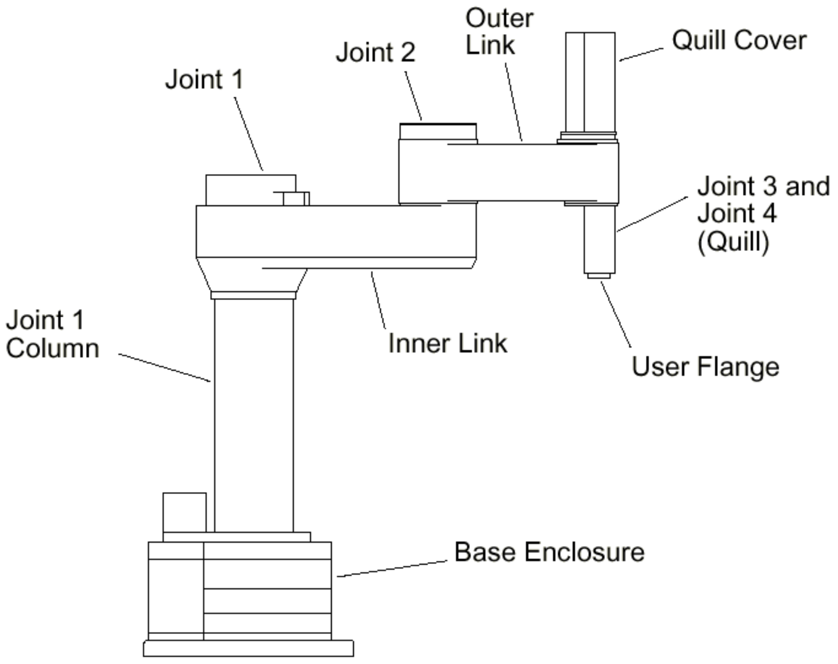


Figure 3. Adept Robot Joint Locations

The AdeptOne robot is a four-axis SCARA robot. Joints 1, 2, and 4 are rotational and Joint 3 is translational. See Figure 1-3 for a description of the robot joint locations. Control and operation of the robot is programmed and performed through the Adept MV controller. Additional safety features are controlled by the Security Panel.

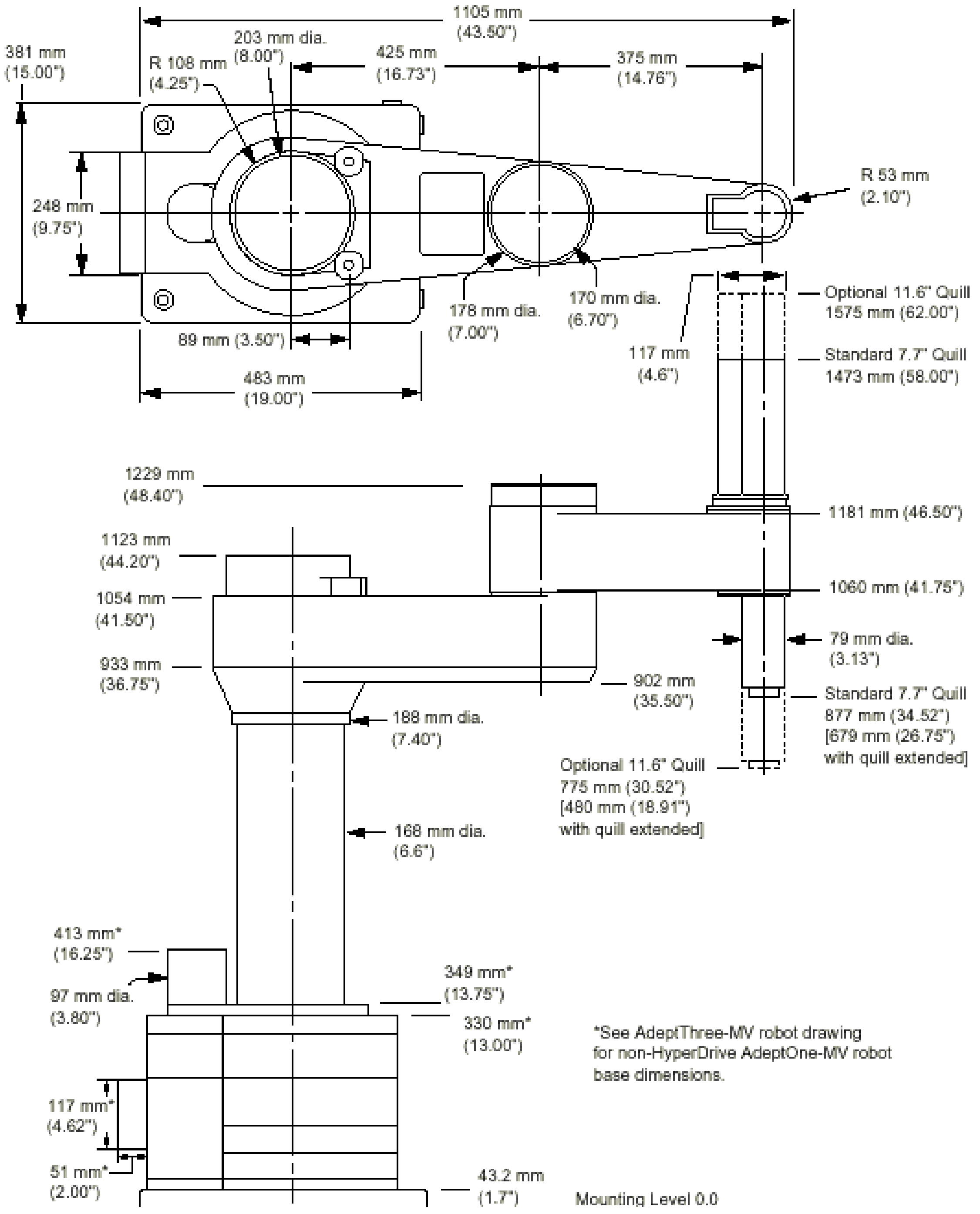


Figure 4. Adept Robot Dimensions

Reach	
Maximum radial	800 mm (31.5 in.)
Minimum radial	231 mm (9.1 in.)
Vertical Stroke - Z direction	
Standard Joint 3	195 mm (7.7 in.)
Optional Joint 3	295 mm (11.6 in.)
Joint Rotation	
Joint 1	300°
Joint 2	294°
Joint 4	554°
Payload (Including End Effector)	9.09 kg (20 lb)
Inertia Load (Maximum)	
About Joint-4 axis – standard	281 kg-cm^2 (96 lb-in^2)
Maximum	2926 kg-cm 2 (1000 lb-in 2)
Resolution	
Joint 1	0.00078°
Joint 2	0.00312°
Joint 3 (vertical Z)	0.0033 mm (0.00013 in.)
Joint 4 (tool rotation)	0.047°
Repeatability	
X,Y plane	±0.025 mm ±0.001 in.)
Joint 3 (vertical Z)	±0.050 mm ±0.002 in.)
Joint 4 (rotational)	±0.05°
Joint Speed (maximum)	
Joint 1	540°/sec
Joint 2	540°/sec
Joint 3	500 mm/sec (19.7 in./sec)
Joint 4	3600°/sec
Weight	
Standard robot without options	180 kg (400 lb)
Hyperdrive robot without options	190 kg (418 lb)
Power chassis, with 3 amplifier modules	Approximately 16.4 kg (36 lb)
MV-8 controller, with 030, SIO, VGB	Approximately 14.5 kg (32 lb)
Design Life	42,000 hours

Figure 5. Adept Robot Performance Specifications

3.MITSUBISHI PA10 ROBOT

PA-10[1] is a powerful super-lightweight industrial robotic arm with 7-axis redundancy control that can be controlled by a PC or a built in control unit, and it offers a payload of 10 Kg.

[1] (http://www.mitsubishitoday.com/1/PA10.htm)

Arm Weight	343N (35 kg)
Max Payload	98N (10 kg)
Number Of Axes	7
Actuation Method	Ac servo motors
Max. Speed (Att/Tip)	1550mm/s
Positional Repeatability	+/- 0.1mm
Arm Length	950mm (SPAN BETWEEN JOINTS)
Protection	Dust proof, drip proof
Controller Weight	245N (25kg)
Programming (Teaching)	Each axis numerical input, position orientation input, 6-axis key-in, direct teaching
Trajectory Control	Point-to-point (ptp) control (line interpolation, circular interpolation, each axis), continuous path (cp) control
System Configuration	Open architecture (isa bus built-in dos/v type)
Power Supply	AC 100 V +/-10%, 50/60 Hz, LESS 1.5 kva
Variation Of Controller	Isa bus built-in windows nt

Figure 6. Mitsubishi PA10t Robot Performance Specifications

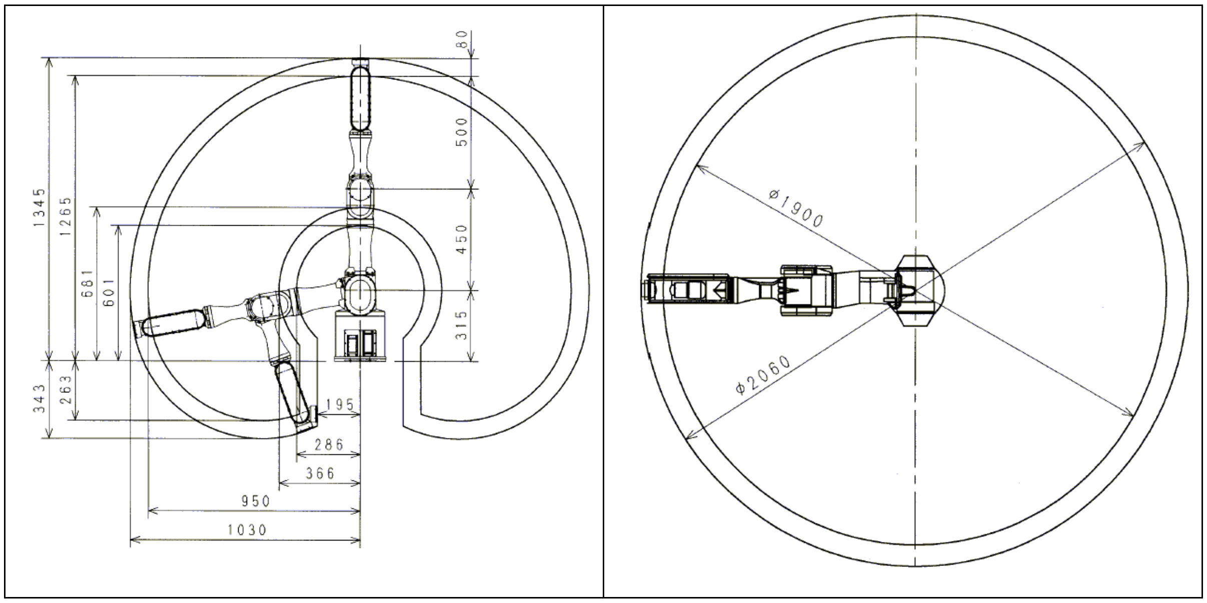


Figure 7. Mitsubishi PA10 Robot Work Area

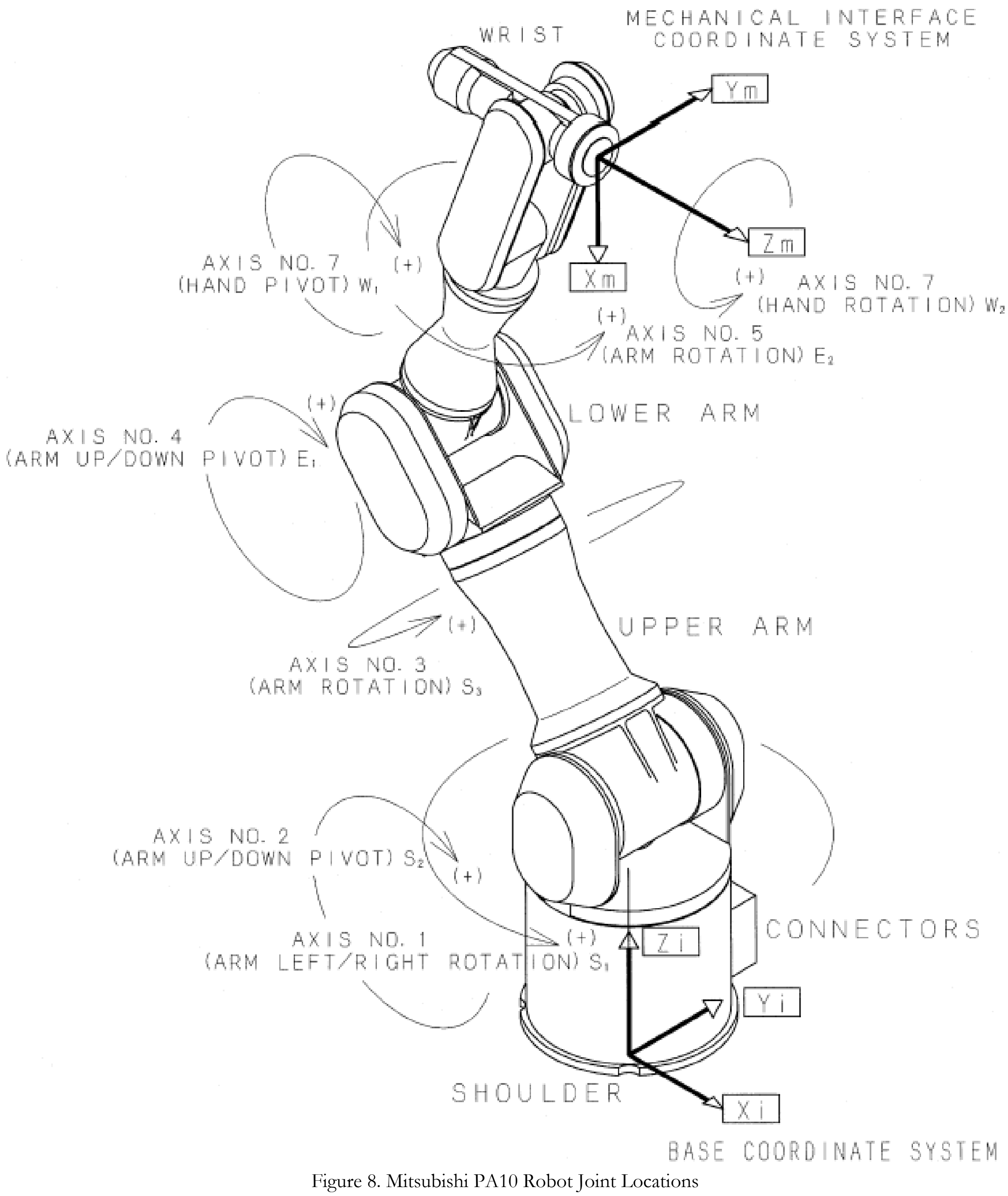

WRIST
MECHANICAL INTERFACE COORDINATE SYSTEM
Ym
Zm
Xm
AXIS NO. 7 (HAND PIVOT) W1
(+)
(+) AXIS NO. 7 (HAND ROTATION) W2
(+) AXIS NO. 5 (ARM ROTATION) E2
LOWER ARM
(+)
AXIS NO. 4 (ARM UP/DOWN PIVOT) E1
UPPER ARM
(+)
AXIS NO. 3 (ARM ROTATION) S3
AXIS NO. 2 (ARM UP/DOWN PIVOT) S2
(+)
(+)
AXIS NO. 1 (ARM LEFT/RIGHT ROTATION) S1
Zi
CONNECTORS
Yi
Xi
SHOULDER
BASE COORDINATE SYSTEM

Figure 8. Mitsubishi PA10 Robot Joint Locations

Appendix II IMPLEMENTATION DETAILS

1.CORBA INTERFACE

1.1. What is CORBA?

CORBA is the acronym for Common Object Request Broker Architecture, OMG's[1] open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard protocol IIOP, a CORBA-based program from any vendor (e.g. "Visibroker", "OrbixWeb", "OmniORB", etc.), on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network.

CORBA applications are composed of objects, individual units of running software that combine functionality and data, and that frequently (but not always) represent something in the real world. Typically, there are many instances of an object of a single type - for example, the UJI Telerobotic Training System website is able to have many robot object instances, all identical in functionality but differing in that each is assigned to a different robot, and contains data representing the actual robot state. For other types (i.e. if we have a unique robot online), there may be only one instance.

1.2. How does it work?

For each object type, such as the robots that we just mentioned, an interface in OMG IDL must be defined. The interface is the syntax part of the contract that the server object offers to the clients that invoke it. Any client that wants to invoke an operation on the object must use this IDL interface to specify the operation it wants to perform, and to marshal the arguments that it sends. In the following section an example of IDL interface for the Mentor Robot is presented. When the invocation reaches the target object, the same interface definition is used there to unmarshal the arguments so that the object can perform the requested operation with them. The interface definition is then used to marshal the results for their trip back, and to unmarshal them when they reach their destination.

The IDL interface definition is independent of programming language, but maps to all of the popular programming languages via OMG standards: OMG has standardized mappings from IDL to C, C++, Java, COBOL, Smalltalk, Ada, Lisp, Python, and IDLscript.

This separation of interface from implementation, enabled by OMG IDL, is the essence of CORBA - how it enables interoperability, with all of the transparencies needed. The interface

[1] (http://www.omg.org)

to each object is defined very strictly. In contrast, the implementation of an object - its running code, and its data - is hidden from the rest of the system (that is, encapsulated) behind a boundary that the client may not cross. Clients access objects only through their advertised interface, invoking only those operations that the object exposes through its IDL interface, with only those parameters (input and output) that are included in the invocation.

Figure 1 shows how everything fits together, at least within a single process: You compile your IDL into client stubs and object skeletons, and write your object (shown on the right) and a client for it (on the left). Stubs and skeletons serve as proxies for clients and servers, respectively. Because IDL defines interfaces so strictly, the stub on the client side has no trouble meshing perfectly with the skeleton on the server side, even if the two are compiled in different programming languages, or even running on different ORBs from different vendors.

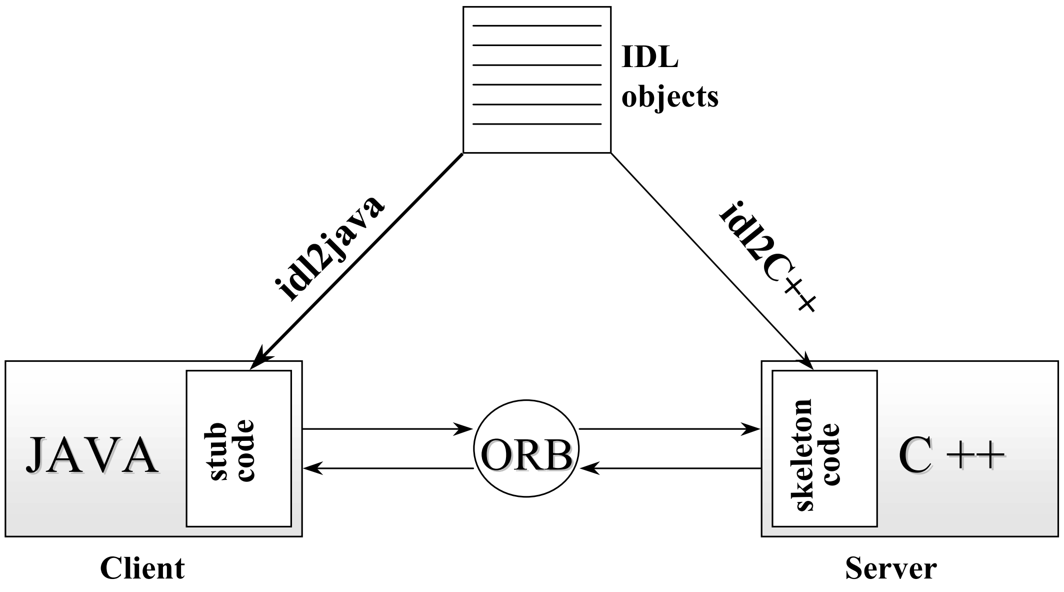


Figure 1. CORBA communication framework

In CORBA, every object instance has its own unique object reference, an identifying electronic token. Clients use the object references to direct their invocations, identifying to the ORB the exact instance they want to invoke (Ensuring, for example, that the operations go into the correct robot). The client acts as if it's invoking an operation on the object instance, but it is actually invoking on the IDL stub which acts as a proxy. Passing through the stub on the client side, the invocation continues through the ORB (Object Request Broker), and the skeleton on the implementation side, to get to the object where it is executed.

1.3. MENTOR Robot IDL Interface

One of the most interesting implementation details of the UJI Telerobotic Training System is the way a CORBA interface has been designed in order to allow any agent located in any platform to control the robot movements. Note the final objective is having a single CORBA interface that allows the low-level control of multiple robot agents (Mitsubishi, Mentor, etc.). The CORBA IDL interface for the Mentor robots is shown at Figure 2.

```
// File: Mentor.idl
//
// Description: Interface to the Mentor Server that gives access to the real Mentor robot  capabilities.
//
// History:
//        10/7/2000, R.Marin,  Creation
//        19/7/2000, A.J. Yepes & R.Marin, Adding return values
//        20/7/2000, R.Marin, Adding comments and return values decisions

module TeleMentor
{
            //Mentor Robot functionality
  interface Mentor
  {
            //Allows a client to get access to the robot.
    short connect();
            //Disconnects the current client.
    void disconnect();
            //It returns the current Mentor position in World coordinates. Returns 1 if succesful and
            //0 if the operation could not be completed. This function makes the client wait until it
            //finishes.
    short getPosition(out double x, out double y, out double z);
            //It forces the robot to move to a world coordinate position. It returns 1 if successful and
            //0 otherwise.
    short moveToPosition(in double x, in double y, in double z);

            //It returns the current Mentor's articulation steps. It returns 1 if successful and 0
            //otherwise.
    short getSteps(out short r1, out short r2, out short r3, out short r4, out short r5, out short r6);
             //It forces the robot to move to the specified articulation steps. It returns 1 if successful and
             //0 otherwise.
    short moveToSteps(in short r1, in short r2, in short r3, in short r4, in short r5, in short r6);

            //It returns the current Mentor's articulation angles (RADIANS). It returns 1 if successful and 0
            //otherwise.
    short getQrad(out double r1, out double r2, out double r3, out double r4, out double r5, out double r6);
             //It forces the robot to move to the specified articulation angles (RADIANS). It returns 1 if successful and
             //0 otherwise.
    short moveToQrad(in double r1, in double r2, in double r3, in double r4, in double r5, in double r6);

            //It returns the current Mentor's articulation angles (DEGREES). It returns 1 if successful and 0
            //otherwise.
    short getQdeg(out double d1, out double d2, out double d3, out double d4, out double d5, out double d6);
             //It forces the robot to move to the specified articulation angles (DEGREES). It returns 1 if successful and
             //0 otherwise.
    short moveToQdeg(in double d1, in double d2, in double d3, in double d4, in double d5, in double d6);
  };
};
```

Figure 2. CORBA IDL Interface to the Mentor Robot.

2.JAVA 3D

In order to implement the 3D virtual scenario including virtual and augmented reality capabilities, the Java 3D[1] technology has been used. It has the particularity of allowing the gen-

[1] (http://java.sun.com/products/java-media/3D/index.html)

eration of 3D models in run-time, which is necessary in order to construct a model of the robot scenario every time the robot moves an object over the scene.

2.1. What is Java3D?

The Java 3D™ API is an application programming interface used for writing stand-alone three-dimensional graphics applications or Web-based 3D applets. It gives developers high level constructs for creating and manipulating 3D geometry and tools for constructing the structures used in rendering that geometry. With Java 3D API constructs, application developers can describe very large virtual worlds, which, in turn, are efficiently rendered by the Java 3D API.

The Java 3D API specification is the result of a joint collaboration between Silicon Graphics, Inc., Intel Corporation, Apple Computer, Inc., and Sun Microsystems, Inc. All had advanced, retained mode APIs under active internal development, and were looking at developing a single, compatible, cross-platform API using Java technology.

The Java 3D API draws its ideas from the considerable expertise of the participating companies, from existing graphics APIs, and from new technologies. The Java 3D API's low-level graphics constructs synthesize the best ideas found in low-level APIs such as Direct3D API's, OpenGL, QuickDraw3D, and XGL. Similarly, Java 3D API's higher-level constructs leverage the best ideas found in several modern scene graph-based systems. It also introduces some concepts not commonly considered part of the graphics environment, such as 3D spatial sound to provide a more immersive experience for the user.

2.2. How does Java3D work?

A programmer constructs a scene graph containing graphic objects, lights, sounds, environmental effects objects, and behavior objects that handle interactions or modify other objects in the scene graph. The programmer then hands that scene graph to Java 3D for execution. Java 3D starts rendering objects and executing behaviors in the scene graph.

Note virtual reality scenarios go through an identical writing process. However, before a user can use such an application, Java 3D must additionally know about the user's physical characteristics (height, eye separation, and so forth) and physical environment (number of displays, their location, trackers, and so on). Not surprisingly, such information varies from installation to installation and from user to user. So Java 3D lets application developers separate their application's operation from the vagaries of the user's final display environment. An API that enables Virtual Reality (VR) applications must support generating 3D graphics, handling input trackers, and continuously integrating that tracker information into the rendering loop. The Java 3D API includes specific features for automatically incorporating head tracker inputs into the image generation process and for accessing other tracker information to control other features. It does this through a new view model that separates the user's environment from the application code. The API also explicitly defines the values returned by six-degrees-of-freedom detectors as primitive Java 3D objects, called Sensors. Together, the new view and input models make it quite easy to turn interactive 3D graphics applications into VR-enabled applications.

2.3. Example of Java3D code

```
    //Mentor robot shoulder
private void createShoulder(Appearance appear, Appearance appear2){
        Transform3D translateHombroT3D = new Transform3D();
        Transform3D rotateHombroT3D = new Transform3D();
        Transform3D translate_ejeHombroCodo = new Transform3D();
        Transform3D rotate_ejeHombroCodo = new Transform3D();
        Transform3D translate_ejeTroncoHombro = new Transform3D();

                //The shoulder 3D representation is kept at Brazo1 class
        Shape3D hombro = Brazo1.BoxBrazo1(appear2);
        hombro.setCollidable(false);  //We disable the collision detection feature because of the performance
        Cylinder ejeHombroCodo =    //We create the Shoulder-Elbow joint
                new Cylinder(EJE_HOMBRO_CODO_WIDTH, 0.085f+0.0105f+0.005f+0.005f+0.08f, appear2);
        ejeHombroCodo.setCollidable(false);   //We disable the collision detection feature because of the performance
        Cylinder ejeTroncoHombro = //We create the Base-Shoulder joint
                 new Cylinder(EJE_TRONCO_HOMBRO_WIDTH, 0.005f+0.085f+0.08f+0.005f, appear2);
        ejeTroncoHombro.setCollidable(false); //We disable the collision detection feature because of the performance

                //We activate the possibility to modify the shoulder transforms (moving the robot shoulder)
        hombroTGT.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        hombroTGR.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

                //We activate the possibility to modify the Elbow transforms (moving the robot Elbow)
        ejeTroncoHombroTGT.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        ejeTroncoHombroTGR.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

                //We adjust the different transforms in order to make them work together
        rotate_ejeTroncoHombro.rotX(Math.PI/2.0);

        translate_ejeHombroCodo.set(
          new Vector3f(0.163f, (0.085f+0.0105f+0.005f+0.005f+0.08f)/2.0f-(0.005f+0.085f+0.08f+0.005f)/2.0f, 0.0f));
        translate_ejeTroncoHombro.set(
          new Vector3f(0.0f, -0.21f/2.0f+0.18f, -(0.005f+0.085f+0.08f+0.005f)/2.0f+0.064f/2.0f+0.005f) );
        translateHombroT3D.set(
          new Vector3f(-0.355f/2.0f+0.183f, 0.08f/2.0f+0.005f-(0.005f+0.085f+0.08f+0.005f)/2.0f, 0.0f));
        rotateHombroT3D.rotX(Math.PI /2.0F);

        hombroTGR.setTransform(rotateHombroT3D);
        hombroTGT.setTransform(translateHombroT3D);

        ejeHombroCodoTGT.addChild(ejeHombroCodoTGR);
        ejeHombroCodoTGR.addChild(ejeHombroCodo);
        ejeHombroCodoTGT.setTransform(translate_ejeHombroCodo);
        ejeHombroCodoTGR.setTransform(rotate_ejeHombroCodo);

        ejeTroncoHombroTGT.addChild(ejeTroncoHombroTGR);
        ejeTroncoHombroTGR.addChild(ejeTroncoHombro);
        ejeTroncoHombroTGT.setTransform(translate_ejeTroncoHombro);
        ejeTroncoHombroTGR.setTransform(rotate_ejeTroncoHombro);
        ejeTroncoHombroTGR.addChild(hombroTGT); //NOTA: La base es el eje inferior
        hombroTGT.addChild(hombroTGR);
        hombroTGR.addChild(hombro);
        ejeTroncoHombroTGR.addChild(ejeHombroCodoTGT);
}
```

Figure 3. Using Java3D to construct the Mentor robot shoulder's 3D representation

At Figure 3 is shown the Java 3D code required to define the 3D representation of the Mentor robot shoulder. Note that the shoulder 3D mathematical description is kept in an already existing Java class called "*Brazo1.class*". Basically, this description consists of a vector of 3D points which determines a generic 3D shape.

It is important to remark that Java3D allows loading 3D worlds in runtime, which have been specified using other definition languages and packages, like for example, "3D Studio Max", "VRML 2.0", etc. In fact, although the Mentor 3D representation have the entirely specified in Java3D, the Mitsubishi and AdeptOne 3D models have been implemented by using VRML and then loading the model into the Java3D library. At Figure 3 a piece of VRML code is presented that represents one of the box where the Mitsubishi base is placed. Of course, the whole Mitsubishi robot representation is much wider.

```
WorldInfo {
  title "Mitsubishi PA10"
  info "Raúl Marín - UJI - 12/08/2001"
}
DEF Focus_libre04 SpotLight {
  intensity 1
  color 0.7059 0.7059 0.7059
  location 0.09861 -107.5 0.5888
  direction 0 1 0
  cutOffAngle 0.7854
  beamWidth 0.7505
  on TRUE
  radius 799.4
}
DEF Focus_libre03 SpotLight {
  intensity 1
  color 0.7059 0.7059 0.7059
  location -101.1 101.9 -96.7
  direction 0.683 -0.2588 0.683
  cutOffAngle 0.7854
  beamWidth 0.7505
  on TRUE
  radius 799.4
}
DEF Focus_libre01 SpotLight {
  intensity 1
  color 0.7059 0.7059 0.7059
  location -99.17 99.9 103.1
  direction 0.6124 -0.5 -0.6124
  cutOffAngle 0.7854
  beamWidth 0.7505
  on TRUE
  radius 799.4
}
DEF Focus_libre02 SpotLight {
  intensity 1
  color 0.7059 0.7059 0.7059
  location 99.89 99.66 101.8
  direction -0.6124 -0.5 -0.6124
  cutOffAngle 0.7854
  beamWidth 0.7505
  on TRUE
  radius 799.4
}
```

```
DEF Focus_libre05 SpotLight {
 intensity 1
 color 0.7059 0.7059 0.7059
 location 101 99.66 -98.98
 direction -0.6124 -0.5 0.6124
 cutOffAngle 0.7854
 beamWidth 0.7505
 on TRUE
 radius 799.4
}
NavigationInfo { headlight FALSE }
DEF ChafBox02-ROOT Transform {
 translation 22.06 639 46.33
 rotation -0.5774 0.5774 -0.5774 -2.094
 scale 0.01 0.01 0.01
 children [
  Shape {
   appearance Appearance {
    material Material {
     diffuseColor 0.898 0.651 0.8431
    }
   }
   geometry DEF ChafBox02-FACES IndexedFaceSet {
    ccw TRUE
    solid TRUE
    coord DEF ChafBox02-COORD Coordinate { point [
     -0.01 0.02 -0.01, 0.01 0.02 -0.01, -0.01 0.02 0.01, 0.01 0.02 0.01,
     -0.01 0.02 -0.01, -0.01 0.02 -0.01, -0.01 0.02 0.01,
     -0.01 0.02 0.01, 0.01 0.02 0.01, 0.01 0.02 0.01, 0.01 0.02 -0.01,
     0.01 0.02 -0.01, -0.01 0 -0.01, -0.01 0 -0.01, -0.01 0 0.01,
     -0.01 0 0.01, 0.01 0 0.01, 0.01 0 0.01, 0.01 0 -0.01,
     0.01 0 -0.01, 0.01 0 0.01, -0.01 0 0.01, 0.01 0 -0.01,
     -0.01 0 -0.01]
    }
    coordIndex [
     0, 2, 3, -1, 0, 3, 1, -1, 0, 4, 5, -1,
     0, 5, 6, -1, 0, 6, 2, -1, 2, 6, 7, -1, 2, 7, 8, -1,
     2, 8, 3, -1, 3, 8, 9, -1, 3, 9, 10, -1, 3, 10, 1, -1,
     1, 10, 11, -1, 1, 11, 4, -1, 1, 4, 0, -1, 4, 12, 13, -1,
     4, 13, 5, -1, 5, 13, 14, -1, 5, 14, 6, -1, 6, 14, 15, -1,
     6, 15, 7, -1, 7, 15, 16, -1, 7, 16, 8, -1, 8, 16, 17, -1,
     8, 17, 9, -1, 9, 17, 18, -1, 9, 18, 10, -1, 10, 18, 19, -1,
     10, 19, 11, -1, 11, 19, 12, -1, 11, 12, 4, -1, 12, 23, 13, -1,
     13, 23, 21, -1, 13, 21, 14, -1, 14, 21, 15, -1, 15, 21, 20, -1,
     15, 20, 16, -1, 16, 20, 17, -1, 17, 20, 22, -1, 17, 22, 18, -1,
     18, 22, 19, -1, 19, 22, 23, -1, 19, 23, 12, -1, 22, 20, 21, -1,
     22, 21, 23, -1]
    }
  }
 ]
}
```

Figure 3. Using VRML to construct the 3D box where the Mitsubishi robot is held

Java3D enables the implementation of Augmented Reality scenarios, due to the fact that the 3D models can be modified in run-time. Hence, virtually generated information like transparencies, gripper projections over the scene, or even prediction of robotic tasks, can be added to the 3D model in order to enhance the information the user gets from the real scenario. This kind of functionality is very convenient in order to facilitate the way users interact with robots.

Appendix III FREQUENTLY ASKED QUESTIONS

This appendix is a collection of interesting questions that have been proposed by experts in the field during meetings, conferences, and other events. I think by reading these questions and their answers people can understand much better the UJI Online Robot System.

1. ¿ Why is the object recognition module included on the server side? ¿Would it not be better to have it as close as possible to the objects database, in order to save network bandwidth? (A. P. del Pobil)

The are two reasons that justify having the configuration distributed in this way:

- As many people can be connected to the Telerobotic System at the same time, ones working with the real robot and others with the simulated one, by including expensive modules on the server (i.e. the object recognition one) means we are taking the risk of overloading the server. The distributed architecture used in the project has the advantage of permitting the re-location of a module depending on the actual requirements. At the moment, the strategy is executing as many codes as possible on the client side, in order to avoid situations where the overall performance goes down. In the future, we will experiment new distributed configurations that make the system work better in every situation.
- The telerobotic system allows interaction using both, on-line and off-line approaches. A single user can be connected in an on-line mode at the same time, and many users can use the off-line approach simultaneously. When the off-line version is used, the input image can be provided by both, the real robot camera (server side) and any sample scene saved on the client side. For the on-line situation, having the object recognition on the server side could be a good alternative, however, as the off-line approach has the image on the client's computer (provided by the user), it would mean having to send it to the server in order to perform the recognition. Obviously this would be very time consuming and would surely overload the server machine.

2. Predictive 3D virtual environments seem to be a good solution to avoid the Internet latency effect. However, what happens when the prediction is different from the real situation (e.g. we predicted the allen key has been grasped, and then, when executed on the real robot an unexpected situation came up that make the operation fail)? (P. J. Sanz)

There are two mechanisms to avoid this situation. First of all, the system follows an augmented reality approach that means the real camera input information is complemented with virtually generated information in order to improve the human-robot communication. If an operation fails and the 3D virtual information does not correspond to the simulated one, the operator will notice it by means of the cameras input. Then, the user is able to refresh the 3D virtual environment by moving the robot to the initial position and then reconstructing the 3D virtual environment state accordingly. This first approach supposes that the user is teleoperating the robot in real time, and is always monitoring what is happening on the robot scenario.

The second alternative is giving the robot some intelligence that makes it detect these unexpected situations and respond accordingly. For example, in our case, collisions are detected by comparing the real robot state with the required values sent by the operator. When they are significantly different it means something is preventing the robot movement and then the robot should stop moving, and an alarm should be sent to the operator.

3. Is it possible controlling the robot by using a simple 56K-modem connection? How long does it takes to be launched? What about the video feedback from the robot? (J. Adell)

One special feature of distributed Java based architectures is their capacity to save network bandwidth because the whole user interface logic is implemented on the client-side, so the information that is going through the network is the minimum required to update the user interface accordingly, depending on the current server-side state (e.g. robot position). In fact, other technologies like, for example, CGI (Common Gateway Interface) require a TCP/IP connection every time the user has pressed a button, due to the situation where the whole user interface logic is held on the server side.

However, some drawbacks have to be assumed, like for example the launching time needed to download a Java applet from the server once user connects to the webpage. When working on campus the launching time is less than 5 seconds. However, when using a low bandwidth connection it increases to more than five minutes. Taking into account that the whole Java applet code occupies 954Kb, which supposes a total of almost 2Mb including some graphical libraries.

By the other hand, once the applet is downloaded and stored on cache the bandwidth effect is less significant. In fact, images from the robot scenario are refreshed every 4 seconds (by default), and they are compressed to a size of 4Kb before sending. Modem based connections give enough bandwidth to support such an amount of data.

4. How can the system construct a 3D model representation of the scene objects by using a top camera input image only? What about the object heights? (Henrik Christensen)

There are two ways to obtain the object's height: using a sensor on top of the gripper, and including this information in advance (when an object is learnt) into the automatic object recognition module.

As explained in chapter II, the optical sensors are appropriate when the illumination conditions are constant, and when the objects to be manipulated have the same material configuration. For those cases where a metallic allen key and a wood piece coexist on the same board, the sensor calibration is very difficult to accomplish. A good alternative would be to use other sensor possibilities, like for example the sonar.

The second alternative consists of storing the objects' heights on the object recognition database. Thus, when a screw is recognized in the scene, as we know the screw has a medium height of 13 mm, we can use this value to construct the 3D model accordingly. The only inconvenience is having to provide the height once a new object has been learnt.

At the moment, as the kind of objects that are being manipulated can have different surfaces, the object recognition alternative is being used. In the near future some experiments with more capable sensors will be carried out.

5. Once the user has completed his task program in the off-line mode, how can he/she be sure that the initial state of objects on the real scene corresponds to the one he/she has used in his predictive task program? (P. J. Sanz)

Refer to independent position tasks in Chapter II.

6. Does the system include collision detection functionality? (G. Recatalà)

The 3D predictive display has the possibility of activating the collision detection functionality thanks to the already existing Java3D libraries. However, when working with the collision detection system activated, the system performances goes down considerably. In fact, the alternative of offering this capability on the virtual scenario has been discarded because of the performance.

On the other hand, the real robot does include reactive collision detection on the robot server. Once the real robot has been commanded to take up a certain position, the real robot server checks if the real robot position corresponds to the required one. If not, we consider a collision has taken place, so the robot stops automatically.

7. Does the system provide the possibility to be executed on a private network, other than the public World Wide Web? Can we avoid the Java Applet downloading time? (A. P. del Pobil)

As explained in Chapter VIII, The UJI Online Robot System can be launched in two modes, as a Java Applet downloaded from a Web page, and as an application that is already installed into the client computer and can be run directly (without recompiling) over a private network. This second possibility increases a lot the system performance, due to the fact that the whole system program is already in the client’s machine. It is not necessary to download the whole Java Applet before executing the program. Moreover, this characteristic would be very interesting for those situations where the use of a public network (e.g. Web) is not the best option due to security and network restrictions (i.e. Medical applications, etc.).

BIBLIOGRAPHY

For the reader convenience, this section includes all the references listed at the end of every individual chapter:

[Adept] (http://www.adept.com/main/products/robots/index.html)

[Alecop] (http://www.alecop.es)

[Ariti] Augmented Reality Interface for Telerobot Application via Internet (http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/index.html).

[Australian] Australian Telerobot webpage (http://telerobot.mech.uwa.edu.au/).

[Backes et al., 1998] P. G. Backes, K. S. Tao, and G. K Tharp. Mars pathfinder mission Internet-based operations using WITS. In proceedings of IEEE International Conference on Robotics and Automation (ICRA), p. 284-ed.291, May 1998.

[Backes et al., 2001] P. G. Backes, K. S. Tso, J. S. Norris, and G. K. Tharp. *Internet-based Ground Operations for Mars Lander and Rover Missions,* K. Goldberg, Roland Siegward, Beyond Web Cams: An introduction to Online Robots, MIT Press, Massachusetts, 2001.

[Bejczy et al., 1990] A. K. Bejczy, Steven Venema, and W. S. Kim., Role of computer graphics in space telerobotics: Preview and predictive displays. In Cooperative Intelligent Robotics in Space, pp. 365-377. Proceedings of SPIE, vol. 1387, 1990.

[Bell, 1965] E. T. Bell, *Men of Mathematics¸* New York: Simon and Schuster, 1965.

[BYG] BYG System Ltd webpage (http://www.bygsystems.com), Nottingham (UK).

[Cave] (http://www.evl.uic.edu/EVL/VR/systems.shtml)

[Chaudhuri, 1996] B. B. Chaudhuri, "A new definition of neighborhood of a point in multi-dimensional space", *Pattern Recognition Letters*, vol. 17, pp. 11-17, 1996.

[Convay et al., 1990] L. Conway, R. Volz and M. Walker. Tele-autonomous systems: Projecting and Coordinating Intelligent Action at a Distance. IEEE Robotics and Automation Journal, vol. 6. No. 2, 1990.

[Cover, 1967] T. M. Cover, P. F. Hart, "Nearest neighbor pattern classification", IEEE Transactions on Information Theory, vol. 13, pp. 21-27, 1967.

[Dalton, 1998] B. Dalton, H. Friz,, and K. Taylor. Augmented reality based object modelling in internet robotics. In Matthew R. Stein, ed., SPIE Telemanipulator and Telepresence Technologies V, vol. 3524, p. 210-217, Boston, 1998.

[Dalton, 2001] B. Dalton. A Distributed Framework for Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001.

[Dasarathy, 1990] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, Los Alamitos, CA: IEEE Computer Society Press, 1990.

[Devijver, 1982] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall: Englewood Cliffs, NJ, 1982.

[Devroye, 1996] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*; New York: Springer, 1996.

[Dragon] Dragon Naturally Speaking of Lernout & Hauspie (http://www.lhsl.com/default2.htm)

[Duda, 1973] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis,* New York: John Wiley & Sons, 1973.

[Dvantech Sonar] (http://www.acroname.com/robotics/parts/R93-SRF04.html)

[Fakespace] (http://www.fakespace.com)

[Faverjon et al., 1991] Faverjon B and Ponce J. *On Computing Two-Finger Force-Closure Grasps of Curved 2D Objects. IEEE Proc. on Robotics and Automation, pp. 424-429.* April 1991.

[Ferrell et al., 1967] W. R. Ferrell and T.B. Sherindan. Supervisory control of remote manipulation. IEEE Spectrum 4 (10): 81-88, October 1967.

[Fuentes et al., 1994] Fuentes O, Marengoni HF, Nelson RC. *Vision-based Planning and Execution of Precision Grasps.* Techincal Report 546, The University of Rochester, Computer Science Deparment, New York. 1994.

[Goertz, 1954] Goertz, R. and Thompson, R. (1954): Electronically Controlled Manipulator: Nucleonics, 1954.

[Goldberg et al., 1995] K. Goldberg, M. Mascha, S. Gentner, J. Rossman, N. Rothenberg, C. Sutter, and J. Widgley, "Beyond the Web: Manipulating the Real World", Computer Networks and ISDN Systems Journal 28, no 1 (December 1995).

[Goldberg et al., 1996] K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, C. Sutter, and J. Wiegley. A Telerobotic Garden on the World Wide Web. SPIE Robotics and Machine Perception Newsletter 5(1). Reprinted in SPIE OE Reports 150, June 1996.

[Goldberg et al., 2000] K. Goldberg, S. Bui, B. Chen, B. Farzin, J. Heitler, D. Poon, R. Solomon and G. Smith. Collaborative Teleoperation on the Internet. In IEEE International Conference on Robotics and Automation (ICRA). San Francisco, CA. April, 2000.

[Goldberg et al., 2000] K. Goldberg, S. Gentner, C. Sutter, and J. Wiegley, "The Mercury project: a feasibility study for Internet robots: design and architecture of a public teleoperated system accessible by anyone on the Internet at any time", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Goldberg et al., 2001] K. Goldberg, Roland Siegward (ed.), Beyond Web Cams: An introduction to Online Robots, MIT Press, Massachusetts, 2001.

[Gonzalez et al, 1993] R. C. Gonzalez and R. E. Woods, *Digital Image Processing,* EEUU: Addison-Wesley, 1993.

[GP2D15] (http://www.acroname.com/robotics/parts/R49-IR15.html)

[Groan et al., 1978] Groan FCA, Verbeek PW. *Freeman-Code Probabilities of Object Boundary Quantized Contours*. Computer Vision, Graphics, Image Processing, vol. 7, pp391-402. 1978.

[Gupta et al., 1998] K. Gupta, A. P. del Pobil (ed.), "Practical Motion Planning in Robotics: Current Approaches and Future Directions", John Wiley & Sons, New York, 1998.

[Hannaford, 2000] B. Hannaford. Feeling is Believing: History of Telerobotics Technology: The robot in the garden: telerobotics and telepistemology in the age of the Internet, K. Goldberg (ed.), MIT Press, Massachussetts, 2000

[Hart, 1968] P. E. Hart, "The condensed nearest neighbor rule", *IEEE Trans. on Information Theory*, vol. 14, pp. 515-516, 1968.

[Hirzinger, 1993] G. Hirzinger. ROTEX the first robot in space. In International Conference on Advanced Robotics, pp. 9-33, 1993.

[Horn, 1993] B. K. P. Horn, "Robot Vision", Cambridge (E.E.U.U), McGraw-Hill, 1993.

[HTK] Hidden Markov Model Toolkit HTK (http://htk.eng.cam.ac.uk/)

[Hu, 1962] M. K. Hu, *Visual Pattern Recognition by Moment Invariants,* IRE Trans. Info. Theory, vol. IT-8, pp. 179-187.

[Iñesta et al., 1997] Iñesta Jm, Sanz PJ, del Pobil AP.*An Automatic Transformation from Bimodal to Pseudobinary Images*. Image Analysis and processing. Lectures Notes in Computer Science, vol 1310, pp. 231-238, A del Bimbo, Springer-Verlag. 1997.

[Jelinek , 1998] Jelinek, F. Statistical Methods for Speech Recognition. MIT Press:Cambridge (1998).

[Kim et al., 1993] W. S. Kim and A.K. Bejczy, "Demostration of a high-fidelity predictive/preview display technique for a telerobotic servicing in space, IEEE Trans. Robotics and Automation 9, no. 5 (October 1993): 698-702.

[Kosuge et al., 2001] K. Kosuge, J. Kikuchi, and K. Takeo, "VISIT: A Teleoperation System via the Computer Network" Beyond webcams, and introduction to online robots, MIT Press, Massachussetts, 2001.

[Lindsay, 1992] T. S. Lindsay, Teleprogramming-remote site task execution. Ph.D. dissertation. The University of Pennsylvania, Philadelphia, 1992.

[Lloyd et al., 1997] J. E. Lloyd, J. S. Beis, D. K. Pai, D. G. Lowe. Model-based Telerobotics with Vision.K. In proceedings of IEEE International Conference on Robotics and Automation (ICRA), p. 1297-ed.1304, April 1997.

[Lucent] Switching and Access R&D Group of Lucent Technologies (Bell Labs Innovations) (http://www.bell-labs.com/).

[Marín et al, 1997] Marín R, Sanz P.J., Coltell O., et al. *Student-teacher communication directed to computer-based learning environments*. Displays, Elsevier Science. Special Issue on Displays for Multimedia (17) pp. 167-178. 1997. Abstract available on http://www.system-concepts.com/displays/

[Marín et at, 1998a] "Distributed Arquitecture for a Learning Telerobotic System" Marín R, Recatalá G, Sanz PJ, Iñesta JM and Pobil AP, Proceedings of the 7th European Workshop on Learning Robotics (EWLR-7), Edinburgh (UK), 1998.

[Marín et al., 1998b] Marin R, Recatalá G, Sanz P, del Pobil AP, Iñesta JM. *Telerobotic System Based on Natural Language and Computer Vision.* Lectures Notes in Artificial Intelligence, vol II, pp. 353-364, A. del Pobil, J. Mira & M. Ali (Eds.). Springer-Verlag. 1998.

[Marín et al., 1998c] R. Marín, "Aplicaciones Distribuidas Orientadas a Objetos como Medio para conseguir la Calidad Total: Standard CORBA", Proceedings of the "VII Jornadas Técnicas de Calidad en Tecnologías de la Información (CIECAT´98)", T*elefónica I+D, Madrid, 1998.* (http://www.tid.es/calidad/ciecat/contra.html*)*

[Marín et al., 1999] R. Marín, E. Jimenez, "Gestores de Red basados en CORBA: Un caso real", Proceedings of the IX Telecom I+D Congress, Madrid (Spain), 1999. (http://www.telecomid.com)

[Marín et al., 2001a] R. Marín, P.J. Sanz. *Telerobotic Training System through the web: Low level Architecture.* In 1st IFAC Conference on Telematics Applications in Automation and Robotics, K. Schilling & H. Roth, IFAC Publications, Elsevier Science Ltd., Weingarthen, Germany, 2001.

[Marín et al., 2001b] R. Marín, P.J. Sanz, *Telerobotic Training System through the web: Low Level Architecture,* Telematics Applications in Automation and Robotics, Oxford: Elsevier Science Ltd, 2001.

[Marín et al., 2001c] R. Marín, J.S. Sanchez, P.J. Sanz. *Design and Evaluation of a Telerobotic System with Object Recognition Capabilities,* Applied Informatics-Artificial Intelligence and Applications; Advances in Computer Applications, M.H. Hamza, IASTED International Conference on Robotics and Applications (RA 2001), Florida (USA), 2001.

[Marín et al., 2001d] R. Marín, P.J. Sanz. *The UJI Telerobotic Training System. 1st EURON Workshop on Robotics Education and Training (RET2001),* Alicia Casals, Antoni Grau, Grup Artyplan-Artymprès, S.A., Weingarthen, Germany, 2001.

[Marín et al., 2001e] R. Marín, P.J. Sanz, *Telerobotic Training System through the web: Low Level Architecture,* Proceedings of the IFAC Conference: Telematics Applications in Automation and Robotics, K. Schilling and H. Roth (ed.), Oxford: Elsevier Science Ltd, 2001.

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

[Marín et al., 2002b] R. Marín, J.S. Sanchez, P.J. Sanz. Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System. In IEEE International Conference on Robotics and Automation (ICRA). Washington, May, 2002.

[Marín et al., 2002c] R. Marín, J.S. Sanchez, P.J. Sanz. *Teleoperated Robot System Via Web: The UJI Telerobotic Training System*. accepted for publication in the Special Issue on Webbased Robotics and Automation of the International Journal of Robotics and Automation, Khaled Elleithy & Tarek Sobh, Actapress, 2002.

[Marín et al., 2002d] R. Marín, P.J. Sanz. *Augmented Reality to Teleoperate or Robot through the Web.* In 15th IFAC World Congress b'02, Barcelona (Spain), 2002.

[Morales et al., 1998] Morales A, Sanz PJ, del Pobil AP. *Computing Contact Stability Grasps of Unknown Objects by Means of Vision*. Proceedings of the IBERAMIA'98, pp. 241-252, Helder Coelho, Ediciones Colibrí. 1998.

[Morales et al., 2001] Morales A, Recatalá G., Sanz PJ, del Pobil AP. *Heuristic vision-based computation of planar antipodal grasps of unknown objects*. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 583-588, Seoul. 2001.

[MultimediaLab] Multimedia Research Group, Jaume I University of Castellón (Spain), (http://www.gm2.uji.es)

[Murphy, 2000] R. R. Murphy, *Introduction to AI Robotics*, The MIT Press, 2000.

[Nakamura, 1991] Nakamura Y. "Advanced Robotic Redundancy and Optimization". Adisson-Wesley series in electrical and computing engeneering. Control engineering. Reading Mass, Adisson-Wesley, 1991.

[Natarajan, 1991] B. Natarajan*, Machine Learning: A Theoretical Approach*, Morgan Kaufmann, 1991.

[Nguyen, 1988] Nguyen V-D. *Constructing Force-Closure Grasps*. The International Journal of Robotics Research. Vol.7, No 3, June 1988.

[PA10] (http://www.mitsubishitoday.com/1/PA10.htm)

[Papoulis, 1965] A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* New York: McGraw-Hill, 1965.

[Parker, 1994] J. R. Parker, "Practical computer vision using C", E.E.U.U, John Wiley & Sons Inc, 1994.

[Pobil et al., 1995] A. P. del Pobil, M. A. Serna, "Spatial Representation and Motion Planning", Springer-Verlag, Berlin, 1995.

[Polaroid Sonar] (http://www.acroname.com/robotics/info/articles/sonar/sonar.html)

[Ponce J et al., 1993] Ponce J, Stam D, Faverjon B. *On computing force-closure grasps of curved two dimensional objects*. Int. J. Robotics Res., Vol 12, No.3, pp. 263-273. June 1993.

[Proffit et al., 1979] D. Proffiet, D. Rossen, "Metrication errors and coding efficiency of chain-encoding schemes for the representation of lines and edges", Comp. Graph. Image Processing, 10: 318-32, 1979.

[QuickTime] (http://www.apple.com/quicktime/qtvr/index.html)

[Rabiner et al., 1993] Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Englewood Cliffs NJ: PTR Prentice Hall (Signal Processing Series), 1993.

[RoboticsLab] Robotic Intelligence Lab, Jaume I University of Castellón (Spain), http://robot.act.uji.es.

[Rosenfeld et al., 1973] Rosenfeld A, Johnston E.*Angle detection on digital curves*. IEEE Transaction on Computers. Vol C-22, pp. 875-878. September , 1973.

[Sánchez, 1998] J. S. Sánchez, "Aprendizaje y Clasificación basados en Criterios de Vecindad. Métodos Alternativos y Análisis Comparativo", Castellón (Spain), thesis dissertation, 1998.

[Sánchez, 2000] J. S. Sánchez, F. Pla, and F. J. Ferri, "Surrounding neighbourhood techniques for nearest-neighbour based classification and prototype selection", in *Recent Research Developments in Pattern Recognition*: Transworld Research Network, 2000, pp. 63-76.

[Sanz et al., 1996] Sanz PJ, Domingo J, del Pobil AP, Pelechano J. *An Integrated Approach to Position a Robot Arm in a System for Planar Part Grasping*. Advanced Manufacturing Forum, vol. 1, pp.137-148. Special Issue on Applications of Artificial Intelligence, 1996.

[Sanz et al., 1996b] Sanz PJ, Iñesta JM, del Pobil AP. Towards and Automatic Determination of Grasping points Through a Machine Vision Approach. In Proc. of the Ninth Intl. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'96), pp. 767-772. Fukuoka, Japan. 1996.

[Sanz et al., 1997] Sanz PJ, del Pobil AP, and Iñesta JM. *Curvature-Symmetry Fusion in Planar Grasping Characterization from 2D Images*. In *Proc*. of the Tenth Intl. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'97), pp. 45-52. Atlanta, U.S.A. 1997.

[Sanz et al., 1998] Sanz P.J., Adell S. An undergraduate Robotics Course via Web. Teleteaching'98 Conference, a part of the 15th IFIP World Computer Congress. Distance Learning, Training, and Education. Austrian Computer Society (book series of). Edit. by Gordon Davies, pp. 859-867, Viena, Austria. 1998.

[Sanzp et al., 1998] Sanz PJ, del Pobil AP, Iñesta JM, Recatalá G. *Vision-Guided Grasping of Unknown Objects for Service Robots*. In IEEE *Proc*. on Robotics and Automation (ICRA'98), pp. 3018-3025. Leuven, Belgium. May 1998.

[SAPI] Microsoft Speech SDK (http://www.microsoft.com/speech/speechSDK/sdkinfo.asp)

[Saucy et al., 2001] P. Saucy, F. Mondada. KhepOnTheWeb: One Year of Access to a Mobile Robot on the Internet: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001.

[Sayers, 1998] C. Sayers, Remote Control Robotics. New York: Springer Verlag, 1998.

[Sayers, 2001] C. Sayers. Fundamentals of Online Robots: Beyond webcams, and introduction to online robots, K. Goldberg and R. Siegwart (ed.), MIT Press, Massachussetts, 2001

[Sherindan, 1963] T. B. Sherindan and W. R. Ferrell, Remote Manipulative Control with Transmission Delay, IEEE Transactions on Human Factors in Electronics HFE-4(1963): 25-29

[Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

[Siegwart et al., 2001] R. Siegwart, P. Balmer, C. Portal, C. Wannaz, R. Blank, and G. Caprari. RobOnWeb: A Setup with Mobile Mini-Robots on the Web: Beyond webcams, and introduction to online robots, MIT Press, K. Goldberg and R. Siegwart (ed.), Massachussetts, 2001.

[Simmons et al., 2001] R. Simmons, R. Goodwin, S. Koenig, J. O'Sullivan, and G. Armstrong. Xavier: An Autonomous Mobile Robot on the Web: Beyond webcams, and introduction to online robots, MIT Press, K. Goldberg and R. Siegwart (ed.), Massachussetts, 2001.

[Sowizral et al., 2001] H. A. Sowizral, M. F. Deering, Projects in Virtual Reality: The Java 3D API and Virtual Reality, Sun Microsystems (http://java.sun.com), 2001.

[Spell, 2000] B. Spell, *Professional Java Programming*, Wrox Press Ltd, EEUU, 2000.

[Stein, 1994] M. R. Stein. Behavior-Based Control For Time Delayed Teleoperation. Ph.D. thesis, University of Pennsylvania MS-CIS-94-43, 1994.

[Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Telegarden] Telegarden webpage (http://www.usc.edu/dept/garden/).

[Thrun, 1995] S. Thrun and T. Mitchell, "Learning one more thing", in *Proc. of the 14th International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, 1995.

[ViaVoice] The IBM Viavoice Automatic Speech Recognition SDK (http://www-4.ibm.com/software/speech/dev/)

[VR Lab Michigan] University of Michigan, Virtual Reality Laboratory (http://www-vrl.umich.edu/)

[VRML] (http://www.web3d.org/vrml/vrml.htm)

[Web3D] The Web3D Consortium Repository (http://www.web3d.org/vrml/vrml.htm)

[Wilson, 1972] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data sets", *IEEE Trans. on Systems, Man and Cybernetics,* vol. 2, pp. 408-421, 1972.

[Wilson, 1997] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions", *Journal of Artificial Intelligence Research,* vol. 6, pp. 1-34, 1997.