

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR



DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

INGENIERÍA TÉCNICA INDUSTRIAL
ESPECIALIDAD ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

**DESARROLLO DE UN ENTORNO VIRTUAL
PARA LA EVALUACIÓN EXPERIMENTAL EN
ROBÓTICA ASISTENCIAL**

AUTOR: VIRGINIA FERNÁNDEZ DE TEJADA MENDIOLA

TUTOR: D. MARTIN FODSTAD STOELLEN

DIRECTOR: D. ALBERTO JARDÓN HUETE

LEGANÉS, MADRID

JULIO 2013

A mi familia, con todo mi cariño.

Índice General

Índice General	V
Lista de Figuras	IX
Lista de Tablas	XIII
Agradecimientos	XV
Resumen	XVII
Abstract	XIX
1 Introducción	1
1.1 Robótica Asistencial.....	1
1.2 El Robot Asistencial ASIBOT	3
1.3 Objetivos	5
1.4 Estructura del Documento.....	7
2 Metodologías para la Evaluación Experimental en Robótica Asistencial	9
2.1 Introducción	9
2.2 Selección de Tareas para la Evaluación Experimental	10
2.2.1 Tareas Simplificadas.....	10
2.2.1.1 Ley de Fitts.....	11
2.2.1.2 <i>Steering Law</i>	13
2.2.2 Relación entre Tareas Simplificadas y Tareas Robóticas	14
2.3 Evaluación Experimental en Robótica Asistencial.....	15

2.3.1	Estrategias para la Evaluación Experimental	17
2.3.2	Características principales de Diseño de los Experimentos	19
2.3.2.1	Robot Asistencial Simplificado	19
2.3.2.2	Entorno Virtual para la Evaluación Experimental	19
2.3.2.3	Tareas Seleccionadas y Representación de Objetivos	20
2.3.2.4	Participantes en el Estudio Piloto	21
3	Entorno Virtual de Experimentación	23
3.1	Introducción	23
3.2	Dispositivos Hardware	24
3.2.1	Dispositivo de Entrada al Ordenador	24
3.2.2	Entorno Físico de Experimentación	25
3.3	Herramientas Software	25
3.3.1	Plataforma Software	25
3.3.1.1	CMake	26
3.3.1.2	Eigen	26
3.3.1.3	KDL (Kinematics and Dynamics Library)	26
3.3.1.4	Boost	27
3.3.1.5	OpenRAVE (Open Robotics Automation Virtual Environment)	27
3.3.1.5.1	Capa <i>Core</i>	29
3.3.1.5.2	Capa <i>Plugins</i>	29
3.3.1.5.3	Capa <i>Scripting</i>	31
3.3.1.5.4	Capa <i>Robot DataBase</i>	31
3.3.2	Comunicación de Datos	32
3.3.2.1	ACE™ (Adaptative Communication Environment)	32
3.3.2.2	YARP (Yet Another Robot Platform)	32
3.4	Arquitectura Software del Entorno de Experimentación	35
3.4.1	Módulo <i>SpaceNavigator</i>	36
3.4.2	Módulo <i>Experiment Manager</i>	36
3.4.3	Módulo <i>Uci Proximity</i>	37
3.4.4	Módulo <i>Openrave Driver</i>	39
3.4.5	Módulo <i>Trajectory Player</i>	43
3.4.6	Módulo <i>Window Feedback</i>	43
3.4.7	Módulo <i>Trajectory Visualizer</i>	45
4	Estudio Piloto y Resultados	47
4.1	Introducción	47
4.2	Metodología Experimental	48

4.2.1	Procedimiento Experimental.....	48
4.2.2	Protección de Datos	51
4.2.3	Riesgos del Estudio.....	51
4.2.4	Beneficios del Estudio	52
4.2.5	Participación voluntaria	52
4.3	Resultados Experimentales	52
4.3.1	Trayectorias Representativas de los Usuarios.....	54
4.3.1.1	Usuario con clave de identificación id 184	58
4.3.1.2	Usuario con clave de identificación id 399	60
4.3.1.3	Usuario con clave de identificación id 624	62
4.3.2	Evaluación de las Estrategias de Control Compartido	64
5	Conclusiones y Trabajos Futuros	67
5.1	Conclusiones	67
5.1.1	Metodologías Experimentales Estudiadas	68
5.1.2	Especificaciones de Diseño de la Plataforma Virtual	68
5.1.3	Plataforma Virtual de Experimentación.....	69
5.1.4	Metodología Experimental Desarrollada	70
5.1.5	Evaluación y Validación de Resultados.....	71
5.2	Trabajos Futuros	71
5.2.1	Manipulador Robótico Asistencial	72
5.2.2	Interacción con el Entorno Virtual de Experimentación.....	72
5.2.3	Control Compartido Adaptativo y Planificación Supervisada	74
5.2.4	Usuarios Finales	74
5.3	Publicaciones Relacionadas	75
6	Presupuesto	77
	Referencias	79
	Anexo A: Manual de Puesta en Funcionamiento	83
	Anexo B: Formulario de Consentimiento	89

Lista de Figuras

Figura 1.1: Tecnologías robóticas asistenciales. <i>a)</i> Manipulador robótico Handy 1 con plataforma para alimentación acoplada. <i>b)</i> Brazo robótico MANUS montado sobre silla de ruedas eléctrica. <i>c)</i> Integración del robot KARES II mediante el uso de dos plataformas: sistema móvil con manipulador robótico KARES II y silla de ruedas eléctrica.....	3
Figura 1.2: El robot ASIBOT anclado a la silla de ruedas en el entorno de pruebas de la cocina para robots asistenciales de la UC3M. Se pueden observar dos estaciones de anclaje situadas en las paredes.....	4
Figura 2.1: Tarea estándar para la ley de Fitts.....	12
Figura 2.2: Tarea lineal estándar para <i>Steering law</i>	14
Figura 2.3: Ejemplo de una tarea robótica simplificada que consiste en el posicionamiento de una lata de refresco sobre un estante de un armario.....	14
Figura 2.4: <i>a)</i> Manipulador robótico Asistencial Manus ARM montado sobre silla de ruedas. <i>b)</i> Plataforma robótica asistencial denominada Raptor ARM.....	17
Figura 2.5: Diferentes tipos de estrategias para la evaluación experimental en robótica asistencial.....	18
Figura 2.6: Mano robótica simplificada sosteniendo una lata de refresco.....	19
Figura 2.7: <i>a)</i> Entorno de pruebas real correspondiente a la cocina para robots asistenciales de la UC3M. <i>b)</i> Entorno virtual de pruebas, utilizado en simulación, que representa el modelo real de la cocina.....	20
Figura 2.8: Representación del objetivo en forma de lata de refresco semitransparente y sobredimensionada sobre un estante del armario de la cocina.....	20
Figura 3.1: <i>a)</i> SpaceNavigator: ratón 3D desarrollado para manipulación de entornos tridimensionales. <i>b)</i> Descripción de los seis grados de libertad del SpaceNavigator, tres para posicionamiento y tres para orientación.....	24

Figura 3.2: Configuración del entorno físico de experimentación.....	25
Figura 3.3: La arquitectura de OpenRAVE se divide en cuatro capas principales: Núcleo, <i>Plugins</i> , Entorno de programación y Base de Datos de Robots.	28
Figura 3.4: Diversos tipos de plataformas robóticas cuya cinemática inversa puede ser resuelta con el algoritmo implementado en OpenRAVE (IKFast).....	30
Figura 3.5: La arquitectura de OpenRAVE con los protocolos de comunicación de red asociados.	31
Figura 3.6: Ejemplo de una red de puertos en YARP con protocolos de comunicación diferentes. Los puertos pertenecen a diversos procesos que pueden ser ejecutados en diferentes ordenadores con sistemas operativos distintos.....	34
Figura 3.7: Módulos software y conexiones YARP, utilizados para realizar experimentos virtuales, que involucran diferentes modos de control de actividades de la vida diaria.	35
Figura 3.8: Módulo <i>spacnavigator</i> que identifica los velocidades transferidas al robot por los usuarios.	36
Figura 3.9: Módulo <i>experiment manager</i> que ofrece información sobre las características del experimento en ejecución.	37
Figura 3.10: Distribución de los sensores infrarrojos por la mano robótica. El eje de color rojo indica la dirección de medida.	38
Figura 3.11: Módulo <i>uci proximity</i> , en ejecución, con información relativa a la velocidades enviadas por el usuario. El parámetro k se ajusta a un valor de 1.6 s. El resto de parámetros se corresponden con valores de depuración del módulo.....	39
Figura 3.12: Ubicación de las distintas posiciones de los objetivos en el entorno de pruebas de la cocina virtual.....	40
Figura 3.13: Posicionamiento de la lata de refresco en el interior del objetivo para cumplimentar la tarea.	40
Figura 3.14: Visor de simulación en OpenRAVE que muestra el entorno virtual de experimentación: cocina, mano robótica y objetivo.	41
Figura 3.15: Módulo <i>openrave driver</i> en ejecución, mostrando información relativa al desarrollo de los experimentos.	42
Figura 3.16: La información relativa a la detección de colisiones con el entorno virtual es mostrada por el módulo <i>openrave driver</i> durante su ejecución.	42
Figura 3.17: a) Ventana con marco de color azul que indica el inicio de la simulación. b) Ventana con marco verde que indica que la simulación está en progreso. c) Ventana con marco de color rojo que indica la existencia de una colisión con el entorno.	43

Figura 3.18: Configuración inicial de partida antes de comenzar con la simulación de la tarea. La ventana de tiempos se rotula en azul.	44
Figura 3.19: Módulo <i>window feedback</i> en ejecución, con información sobre la fuente de escritura seleccionada como parámetro de entrada.	44
Figura 3.20: <i>a)</i> Representación de todas las trayectorias del experimento para dos de los sujetos (azul y morado). <i>b)</i> Representación de las trayectorias de dos sujetos para una única tarea específica. <i>c)</i> Representación de las trayectorias de tres sujetos (azul, verde y morado) para dos tareas, con inclusión de ruido. <i>d)</i> Representación de las trayectorias representativas de un único sujeto para dos tareas.	45
Figura 4.1: El temporizador cambia su color a rojo cuando la mano robótica colisiona con el armario de la cocina.	49
Figura 4.2: <i>a)</i> Sesión de control. <i>b)</i> Sesión 1. <i>c)</i> Sesión 2. <i>d)</i> Sesión 3.	53
Figura 4.3: <i>a)</i> Trayectorias realizadas por los 3 sujetos durante la sesión de control, con 27 repeticiones sobre cada tarea. <i>b)</i> Regresión mixta gaussiana de las trayectorias generadas por los 3 sujetos para las correspondientes tareas.	56
Figura 4.4: <i>a)</i> Trayectorias generadas por un usuario (27 repeticiones) y GMR obtenida para la sesión de control, sin ruido gaussiano (nivel bajo). <i>b)</i> Trayectorias generadas por un usuario (27 repeticiones) y GMR obtenida para la sesión 1, con ruido gaussiano ajustado a nivel alto. El objetivo se corresponde con la tarea situada en la encimera y se muestra mediante un punto verde.	57
Figura 4.5: GMR y GMM de las trayectorias de la sesión 1 y 3 (usuario id 184).	58
Figura 4.6: GMR y GMM de las trayectorias de la sesión 1 y 3 (usuario id 399).	60
Figura 4.7: GMR y GMM de las trayectorias de la sesión 1 y 3 (usuario id 624).	62
Figura 4.8: <i>a)</i> Media de los tiempos (MT) para el primer intento realizado con éxito. <i>b)</i> Número total de errores. Para cada sesión se muestra la media para los tres sujetos, mientras que las barras de error indican la desviación estándar.	64
Figura 4.9: <i>a)</i> Media de los tiempos (MT) para el primer intento realizado, con o sin éxito. <i>b)</i> Media de los tiempos (MT) para todas las repeticiones. Para cada sesión se muestra la media para los tres sujetos, mientras que las barras de error indican la desviación estándar.	65
Figura 5.1: Dispositivo háptico modelo PHANTOM Omni®.	73

Lista de Tablas

Tabla 3.1: Pseudocódigo correspondiente al algoritmo utilizado para la implementación del control compartido.....	38
Tabla 4.1: Codificación probabilística del conjunto de datos de las trayectorias empleando la Regresión Mixta Gaussiana (GMR).	55
Tabla 6.1: Presupuesto global del Proyecto.	78
Tabla A.1: Script para ejecutar las conexiones de los puertos YARP.	87
Tabla A.2: Asociación entre las conexiones de los puertos YARP de entrada y salida y la etiqueta que los identifica con el flujo de datos.	88

Agradecimientos

A mi tutor, Martin F. Stoelen, por valorar sinceramente mi trabajo y esfuerzo, y ofrecerme continuamente su apoyo para seguir adelante en este mundo tan difícil, pero que tanto me apasiona. Gracias de nuevo por el aporte constante de motivación y acertado consejo durante todo este tiempo.

Quisiera también agradecer encarecidamente a mi director, Alberto Jardón, haber soportado serenamente mi mal humor y demostrar verdadera voluntad de ayuda a pesar de su escaso tiempo disponible.

No podría dejar de mencionar a las personas del RoboticsLab que me han dedicado su tiempo y prestado su inestimable ayuda, de forma reiterada y absolutamente desinteresada.

Por último, a todos los que no he mencionado pero que llevo siempre conmigo, en mi pensamiento, y que, de algún modo, forman parte de mí. Sin duda ellos saben quienes son.

Resumen

El desarrollo de sistemas robóticos asistenciales inteligentes es actualmente un campo de investigación activo en la comunidad robótica. Un ejemplo son los manipuladores robóticos asistenciales que pueden ayudar a las personas mayores y discapacitadas en su vida diaria. Estos robots generalmente se destinan a ser utilizados en actividades típicas de la vida diaria dentro de entornos dinámicos complejos, como el hogar del usuario. La naturaleza heterogénea de los usuarios se presenta como otro desafío, por ello se requiere flexibilidad y capacidad de adaptación por parte del robot asistencial. La Interacción Hombre-Robot (HRI) constituye un elemento fundamental para conseguir que estos sistemas robóticos sean útiles para los usuarios finales. Sin embargo, dada la complejidad y la diversidad de los sistemas robóticos, la interacción entre los usuarios y el robot no es fácil de evaluar experimentalmente de forma que pueda suponer una contribución durante el proceso de desarrollo. Por tanto, existe una falta de procedimientos universales para cuantificar el rendimiento debido a que cada entorno requiere sus propias medidas específicas. También es deseable utilizar medidas fiables que determinen el rendimiento durante la evaluación clínica para que proporcionen validez a los dispositivos robóticos asistenciales.

En este documento se exponen algunos de los trabajos realizados en este sentido en el proyecto ASIBOT, mediante la descripción de la arquitectura de software desarrollada para dicho propósito, así como los resultados preliminares en la creación de trayectorias representativas y la evaluación del control compartido sobre tareas virtuales de la vida cotidiana.

Abstract

The development of intelligent service robotic systems is currently an active field of research in the robotics community. For example assistive robot manipulators that can aid elderly and disabled people in their daily life. These robots are typically intended to be used in complex unstructured environment, like the user's home, and on typical Activities of Daily Living (ADLs). The heterogeneous nature of the user group is another challenge, requiring a flexible and adaptable assistive robot. The Human Robot Interaction (HRI) is an important part of making these robotic systems useful for the end-users. However, given the complexity and diversity of the robotic systems, the interaction between the users and the robot is not easy to evaluate experimentally in a way that can help drive the development process. Therefore, there is a lack of ubiquitous ways to measure performance because each domain needs specific performance measures. It is also desirable to use well established performance measures for clinical evaluation to lend validity to an assistive robot device.

This document outlines some of the work performed in this direction in the ASIBOT project, describing the software architecture developed for this purpose, as well as preliminary results in creating benchmark trajectories and evaluating shared control on virtual ADLs.

Introducción

1.1 Robótica Asistencial

Multitud de tareas de manipulación humana requieren el uso de las extremidades superiores. Cualquier deficiencia en esta parte del cuerpo provoca una pérdida de destreza y una disminución del rendimiento obtenido durante el proceso de manipulación. Estas dificultades afectan a las actividades diarias más comunes tales como el movimiento y posicionamiento de objetos, el uso de herramientas o utensilios, lavarse, vestirse, abrir y cerrar puertas, encender y apagar luces, así como las necesidades fisiológicas básicas como comer y beber. Los robots asistenciales tienen la capacidad de ayudar en estas tareas y pueden proporcionar asistencia personalizada, individualmente o en equipo (Mataric et al., 2007).

La robótica asistencial se está desarrollando actualmente como apoyo a las personas mayores y discapacitadas en actividades de la vida cotidiana realizadas en entornos no estructurados del mundo real. Una de las características principales en la interacción humano-robot (HRI), en robótica asistencial, es la diversidad de los potenciales usuarios, con diferentes tipos y grados de discapacidad, tanto física como

mental. Los manipuladores robóticos asistenciales presentan la capacidad de mejorar la calidad de vida de los usuarios finales mediante el aumento del nivel de independencia en la ejecución de dichas actividades, que abarcan desde el posicionamiento y emplazamiento de objetos hasta tareas que involucran el cuidado personal (A. Jardón et al., 2011).

Las tecnologías robóticas asistenciales tradicionales se han centrado, principalmente, en el desarrollo de tres conceptos:

- Sistemas estáticos que operan en ambientes estructurados.
- Sistemas robóticos montados sobre silla de ruedas.
- Manipuladores móviles que acompañan al usuario y desempeñan aplicaciones de cuidado personal.

El primer tipo de sistema robótico es recomendable cuando el usuario necesita asistencia para desarrollar siempre las mismas aplicaciones, dentro de un mismo entorno doméstico de espacio reducido. Estas tareas se corresponden, a modo de ejemplo, con las relacionadas con la higiene personal, comer, beber, etc. El manipulador robótico Handy 1 (Topping, 1993), mostrado en la Figura 1.1 a), es un ejemplo de sistema robótico estático de bajo coste para la asistencia y el cuidado personal. No obstante, los sistemas robóticos estáticos presentan el gran inconveniente que implica el cambio de emplazamiento de la plataforma robótica. En ocasiones, este cambio puede resultar muy complicado o imposible.

Otro tipo de robots utilizados en rehabilitación son los sistemas robóticos montados sobre silla de ruedas. El líder actual del mercado, en lo que concierne a esta modalidad de robots, es el sistema robótico MANUS¹ con cientos de unidades vendidas hasta el momento. Este sistema robótico consta de un brazo robótico, dotado de seis grados de libertad, que permanece permanente anclado en el lado derecho o izquierdo de una silla de ruedas eléctrica. Sin embargo, esta localización asimétrica del manipulador puede convertirse en un inconveniente durante la ejecución de ciertas

¹ <http://www.exactdynamics.nl/site/> (Última revisión: 02-07-2013)

tareas y, asimismo, producir problemas de movilidad cuando se atraviesan puertas o existen escaleras. En la Figura 1.1 *b)* se puede observar el sistema robótico MANUS.

El tercer concepto se refiere a un sistema móvil con manipulador robótico integrado que permite realizar el seguimiento de la silla de ruedas utilizada por el usuario. Este concepto tiene inconvenientes similares a los ya expuestos con anterioridad. La movilidad dentro de un entorno doméstico no es siempre ideal debido a la existencia de escalones u otros obstáculos. Sin embargo, introduce una nueva ventaja que no presentan los modelos anteriores, el robot tiene la capacidad de desplazarse de forma independiente con respecto a la silla de ruedas y al usuario. Un ejemplo bastante conocido de este tipo de sistema robótico es el manipulador móvil KARES II (Bien, Z. et al., 2004), mostrado en la Figura 1.1 *c)*.

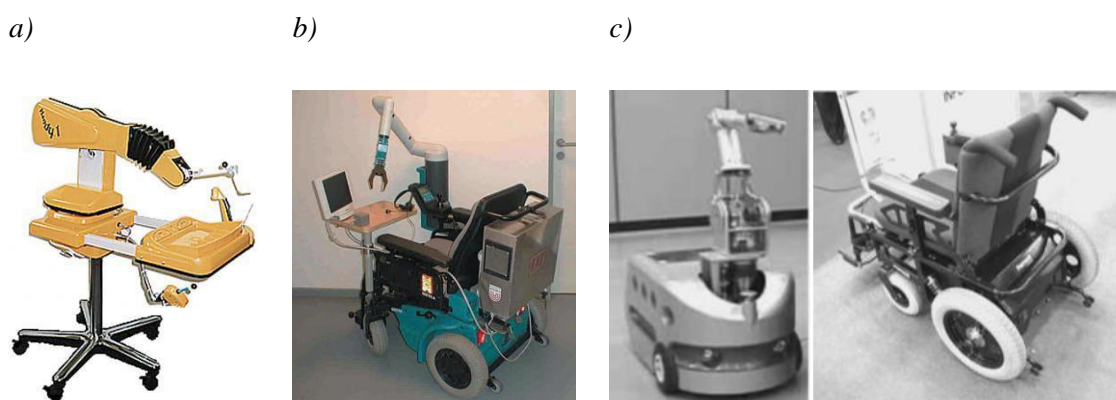


Figura 1.1: Tecnologías robóticas asistenciales. *a)* Manipulador robótico Handy 1 con plataforma para alimentación acoplada. *b)* Brazo robótico MANUS montado sobre silla de ruedas eléctrica. *c)* Integración del robot KARES II mediante el uso de dos plataformas: sistema móvil con manipulador robótico KARES II y silla de ruedas eléctrica.

1.2 El Robot Asistencial ASIBOT

El robot asistencial ASIBOT ha sido desarrollado en la Universidad Carlos III de Madrid (UC3M) y presenta una serie de ventajas, inusuales en otros robots asistenciales. Debido a su ligero peso y carácter escalador, ASIBOT posee la capacidad de trasladarse entre diferentes puntos de anclaje situados en el entorno o montado sobre

una silla de ruedas y realizar tareas en zonas de difícil acceso desde una base móvil. Véase por ejemplo (Balaguer, C. et al., 2006). Estas estaciones de anclaje permiten no sólo el desplazamiento y la acometida de tareas a través de superficies normalmente no utilizadas, como pueden ser suelos y paredes, sino el suministro de la alimentación necesaria al robot. Cuando el robot se encuentra acoplado a la estación de anclaje de la silla de ruedas, ver Figura 1.2, las propias baterías de la silla de ruedas se encargan de suministrar la energía para su funcionamiento.



Figura 1.2: El robot ASIBOT anclado a la silla de ruedas en el entorno de pruebas de la cocina para robots asistenciales de la UC3M. Se pueden observar dos estaciones de anclaje situadas en las paredes.

El diseño simétrico de ASIBOT, dotado de cinco grados de libertad, se divide en dos partes diferenciadas: la estructura articulada del brazo y los extremos. Estos últimos incorporan mecanismos para acoplar el robot a las estaciones de anclaje, mediante conectores de alimentación de 24V, y también una pinza. La estructura articulada tiene dos eslabones centrales que contienen todos los equipos electrónicos y la unidad de control del brazo. Es importante reseñar que, debido a la simetría del brazo del robot, es posible fijar cualquiera de sus extremos, lo que resulta mucho más conveniente para lograr el éxito en el desarrollo de una tarea de teleoperación específica.

La portabilidad de ASIBOT está garantizada debido a su estructura de aluminio, reforzada con fibra de carbono, que le confiere un peso de 12 kilogramos. Su diseño permite un alcance de 1.3 metros, lo que implica que la relación peso/longitud es

extremadamente baja, y el manejo de carga de hasta 1 kilogramo en su configuración de máxima extensión. El peso y tamaño reducido disminuye los requisitos estructurales de los soportes de las estaciones de anclaje así como el par máximo del actuador. Estas características suponen un incremento de la seguridad cuando se trabaja en estrecha proximidad con el usuario, por ejemplo, cuando el robot está conectado a la silla de ruedas.

La interfaz actual hombre-máquina (HMI) está basada en un asistente personal digital (PDA) inalámbrico. Diferentes tipos de interacción son posibles dependiendo de las capacidades del usuario. Esto incluye un *joystick* para el control directo del robot y la posibilidad de interactuar mediante el uso de sencillos botones a través de la pantalla táctil de la PDA; además de un sistema de reconocimiento de voz que permite a los usuarios con mayor grado de discapacidad desplazarse por el menú de la interfaz, y dispositivos de entrada flexibles basados en aceleraciones e inclinaciones que pueden ser utilizados con distintas partes del cuerpo (Balaguer, C. et al., 2007).

El sistema ha sido ampliamente evaluado por los usuarios finales mediante procesos de investigación y desarrollo (Balaguer, C. et al., 2005). Psicólogos, terapeutas y médicos han estado involucrados en dichos procesos y han ayudado a estimar el beneficio que los usuarios han obtenido del sistema. El 86% de los sujetos implicados en los experimentos han expresado la capacidad que les otorga el sistema para lograr nuevos retos. La posibilidad de transferir el brazo robótico desde la silla de ruedas hacia las estaciones de anclaje y viceversa, así como agarrar y soltar objetos, se encuentran entre las tareas a las que los sujetos han respondido más positivamente.

1.3 Objetivos

Este proyecto tiene como propósito principal el desarrollo de una plataforma software que permita la evaluación experimental de metodologías aplicada al campo de los manipuladores robóticos asistenciales.

Dicha plataforma debe permitir la simulación de tareas cotidianas, específicamente diseñadas en el entorno virtual de experimentación, y posibilitar la validación experimental de los resultados obtenidos mediante la realización de estudios pilotos.

De forma más concisa, el conjunto de objetivos que se pretende alcanzar mediante la elaboración del presente proyecto se detallan a continuación:

- Realización de un estudio del estado del arte en metodologías que cuantifican el rendimiento en la evaluación experimental de dispositivos de entrada al ordenador y, de forma más extendida, en robótica asistencial.
- Selección de las especificaciones de diseño necesarias para el desarrollo del entorno virtual de experimentación.
- Implementación del módulo software destinado a la simulación virtual del entorno de experimentación, que incluya los modelos tanto del entorno como del robot y el conjunto de tareas propuestas.
- Implementación de un módulo software que permita la visualización de las trayectorias generadas por los distintos usuarios para cada una de las tareas planteadas.
- Implementación de un módulo software generador de ruido gaussiano, filtrado a baja frecuencia, utilizado como modelo de discapacidad genérica.
- Definición de la metodología experimental empleada para la evaluación de un manipulador robótico asistencial en el desempeño de actividades que se realizan de forma cotidiana.
- Evaluación y validación de todos los procedimientos empleados y módulos software desarrollados, mediante un estudio piloto realizado por usuarios que no presentan ninguna discapacidad.

1.4 Estructura del Documento

En este primer capítulo se ha realizado una breve introducción al campo de la robótica asistencial y una descripción detallada de las características del robot asistencial ASIBOT, desarrollado en la Universidad Carlos III de Madrid. Asimismo, se exponen los objetivos principales a cumplimentar y la estructura del proyecto.

En el segundo capítulo se explican diversas metodologías para la realización de estudios de evaluación experimental, referentes a la interacción hombre-robot (HRI) en robótica asistencial. Se explica también la importancia en la selección de tareas apropiadas y las especificaciones necesarias para el diseño de los experimentos.

En el tercer capítulo se describe primordialmente la implementación de la arquitectura software diseñada para la realización de experimentos. Se detallan las herramientas hardware empleadas, así como los programas software específicos que resultan necesarios para la utilización y el correcto funcionamiento de la plataforma de experimentación.

El cuarto capítulo contiene los resultados experimentales relativos al estudio piloto realizado y se definen los procedimientos empleados durante la evaluación experimental de las tareas cotidianas, en el entorno virtual de experimentación diseñado. Asimismo se muestran los datos de simulación, obtenidos para cada usuario, correspondientes a una tarea específica implementada en el simulador.

Las conclusiones que se han obtenido durante el desarrollo del proyecto se exponen en el quinto capítulo, junto con la propuesta de posibles trabajos futuros y líneas de investigación viables que se pueden desarrollar.

Al final del documento, se incluyen el capítulo correspondiente al presupuesto del proyecto, las referencias utilizadas en este trabajo y los anexos.

Capítulo 2

Metodologías para la Evaluación Experimental en Robótica Asistencial

2.1 Introducción

Los experimentos son esenciales en la ciencia, tanto para confirmar o refutar una teoría como para elegir entre hipótesis alternativas. Algunos de los principios experimentales utilizados en la ciencia son: la comparación, la reproducibilidad/repetitividad y la justificación/explicación. El principio de comparación implica el conocimiento de lo que se ha hecho en el pasado para evitar la repetición de experimentos no interesantes y discernir cuáles podrían ser las líneas de investigación más fructíferas. Reproducibilidad y repetitividad representan conceptos que a menudo son objeto de confusión. El primer concepto se refiere a la posibilidad de verificar de forma independiente un experimento, mientras que el último significa que es necesario realizar varios ensayos, preferentemente en condiciones diferentes, para obtener resultados representativos. Finalmente, existe una necesidad de obtener conclusiones que resulten justificadas y traten de explicar los datos recogidos (Amigoni, F., Reggiani, M. and Schiaffonati, V., 2009).

El desarrollo de la interacción hombre-robot (HRI) en robótica asistencial requiere la realización de ensayos para evaluar eficientemente los cambios que se producen en el sistema. De igual modo, resulta necesario documentar de forma adecuada el beneficio potencial de la interfaz para el resto de la comunidad investigadora.

2.2 Selección de Tareas para la Evaluación Experimental

Generalmente, los robots asistenciales se destinan a ser utilizados dentro del entorno diario en el que se desenvuelven los usuarios. Este ambiente varía de un usuario a otro y es difícil de especificarlo durante la etapa de diseño, lo que hace que sea complicado llegar a un conjunto representativo de tareas para realizar una comparación de la interacción hombre-robot (HRI) en robótica asistencial. También parecen existir limitaciones en las conclusiones que pueden extraerse de los resultados obtenidos con la aplicación de tareas específicas.

No se puede asegurar que la interfaz con el mejor rendimiento en realizar una tarea de posicionamiento de un objeto sea la más óptima para efectuar la apertura de una puerta. Además, las tareas de aplicación específica pueden incluir, en muchos casos, una componente subjetiva de interpretación por parte del usuario. En conclusión, resulta complicado inferir cuanto de generalizables son los resultados logrados, o si el rendimiento de una interfaz viene determinado, en mayor o menor medida, por la interpretación personal de los diferentes usuarios e investigadores.

Un enfoque consistiría en utilizar una tarea más general, que sea similar a la deseada, manteniendo unas medidas confiables de rendimiento que representen la dificultad que supone la ejecución de la tarea.

2.2.1 Tareas Simplificadas

Las tareas de seguimiento se utilizan usualmente para evaluar el rendimiento de los sistemas que involucran a seres humanos. Este tipo de tareas, comúnmente, implica el control de un objeto, el cursor, para realizar un seguimiento lo más próximo posible a

otro objeto, el objetivo. Por lo general, los dos objetos son virtuales, ambos se mueven sobre una pantalla de ordenador, existiendo dos tipos principales de tareas de seguimiento, búsqueda y compensación. Para la primera tarea el objetivo se mueve por toda la pantalla, y el cometido del usuario es realizar su seguimiento tan cerca como sea posible. En el último caso, la tarea consiste en reducir el error entre el cursor y un objeto fijo, por ejemplo, una línea en el centro de la pantalla.

Los movimientos a seguir pueden variar dependiendo de la dificultad deseada y la finalidad de las pruebas. Esto implica simples movimientos sinusoidales (Ware, 1972) y (Shook, L. & Akin, D., 2001) o movimientos más complejos que introducen el uso de ruido blanco filtrado a baja frecuencia (Chan, R. & Childress, D., 1990a).

Sin embargo, muchas de las tareas que realizan los robots asistenciales involucran movimientos discretos, por ejemplo, coger o posicionar un objeto. Por lo tanto, también puede resultar beneficioso contemplar tareas simplificadas que no tienen un carácter continuo.

2.2.1.1 Ley de Fitts

Desde su publicación original, la ley de Fitts (Fitts, 1954) se ha convertido en una herramienta importante en la modelación de la compensación velocidad/precisión en movimientos humanos simples. Esta ley ha sido ampliamente aplicada en el campo de la interacción hombre-computadora (HCI) para diseñar y cuantificar el rendimiento de la interfaz gráfica del usuario.

Como se puede apreciar en la Ecuación 2.1, el modelo que predice el tiempo medio (MT) para completar un movimiento varía linealmente con el índice de dificultad (ID). Este índice es una función de la distancia completada y de la precisión requerida, o de la tolerancia impuesta en el movimiento. Estas propiedades se denotan como la distancia recorrida durante el movimiento, D , y la anchura de la zona objetivo, W , respectivamente. La formulación estándar para el índice de dificultad, ID , se muestra en la siguiente ecuación:

$$MT = a + b \cdot ID,$$

donde :

(2.1)

$$ID = \log_2 \left(\frac{D}{W} + 1 \right).$$

El parámetro ID tiene unidades de bits y se fundamenta en la Teoría de la Información. De hecho, dicha ley se interpretó originalmente como una medida de la capacidad de información del sistema motor humano. Normalmente la aplicación de la ley de Fitts se circunscribe a desplazamientos simples de la mano, de izquierda a derecha, tal y como puede observarse en el ejemplo representado en la Figura 2.1.

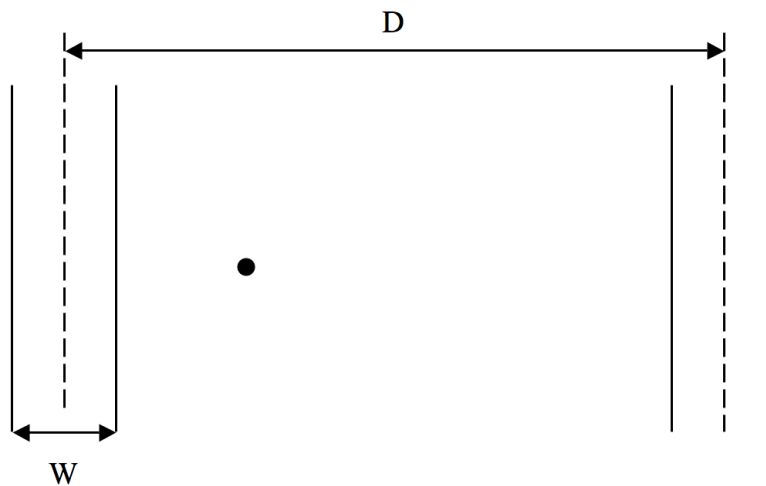


Figura 2.1: Tarea estándar para la ley de Fitts.

Los coeficientes a y b se determinan experimentalmente mediante un análisis de regresión lineal. La pendiente, coeficiente b , se convierte en una medida de la tasa de variación del tiempo necesario para concluir la tarea y varía con el nivel de dificultad de la misma. Su inverso, $1/b$, se conoce como índice de rendimiento (IP) y tiene unidades de bits/segundo. En otras palabras, el rendimiento humano para una tarea, dada una distancia e impuesto un requisito de tolerancia, se puede predecir en base a las observaciones de dichas combinaciones.

La versión de la Ley de Fitts utilizada anteriormente fue propuesta por primera vez por Mackenzie (Mackenzie, 1989) y constituye la base para las pruebas de rendimiento de la norma ISO 9241-9, que abarca los requisitos ergonómicos para dispositivos de entrada al ordenador excluyendo los teclados.

Esta ley se utiliza normalmente en las comparaciones de los dispositivos de entrada, puesto que proporciona la capacidad de generalizar los resultados más allá de una tarea específica. De hecho, la ley de Fitts constituye una de las pocas herramientas cuantitativas existentes disponibles para los diseñadores de interfaces hombre-máquina. Sin embargo, hoy en día se considera más una ley empírica que un modelo para describir los mecanismos subyacentes del movimiento humano.

2.2.1.2 *Steering Law*

Steering Law (Ley de Seguimiento), definida mediante la Ecuación 2.2, supone una extensión de la ley de Fitts transformando los movimientos recíprocos o discretos de dicha ley a trayectorias continuas (Accot, J. & Zhai, S., 1997).

$$ID_{Steering} = \frac{d}{w},$$

donde :

$$w = k - b. \quad (2.2)$$

En esencia, al aplicar un número infinito de objetivos a una tarea de la Ley de Fitts a lo largo de una trayectoria dada, el rendimiento humano, durante el recorrido en 2D de pasillos de diferentes formas, utilizando dispositivos de entrada al ordenador, puede ser modelado. En la Figura 2.2 se muestra una tarea estándar en línea recta para *Steering Law*.

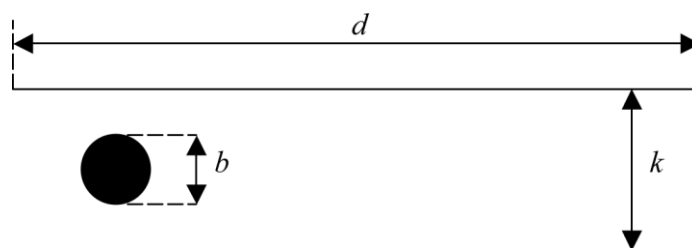


Figura 2.2: Tarea lineal estándar para *Steering law*.

2.2.2 Relación entre Tareas Simplificadas y Tareas Robóticas

Los modelos presentados anteriormente se han utilizado con éxito en una amplia gama de tareas simples y para un amplio rango de usuarios. No obstante, no es evidente que estos modelos sean también representativos de los movimientos requeridos para los robots asistenciales. La primera cuestión que surge es si una tarea compleja como la colocación de una lata de refresco en un armario de una cocina puede ser representada aproximadamente por un conjunto de dichos movimientos primitivos. Dado que tanto la Ley de Fitts como *Steering Law* se aplican normalmente a entornos planos, se muestra como tarea robótica representativa la expuesta en la Figura 2.3.

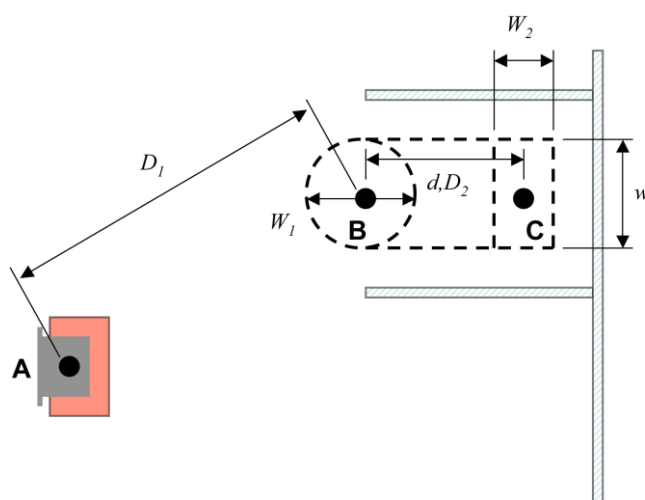


Figura 2.3: Ejemplo de una tarea robótica simplificada que consiste en el posicionamiento de una lata de refresco sobre un estante de un armario.

Asumiendo que existen pocos objetos, la tarea mostrada anteriormente puede considerarse plana debido a que las restricciones fuera de plano son relativamente menores con respecto a las que se encuentran dentro del plano. Una interpretación de la tarea podría ser la de un movimiento tosco de la Ley de Fitts hasta el punto de entrada situado entre los dos estantes (desde A hasta B). A continuación, le sucedería un movimiento basado en *Steering Law* entre los dos estantes y, finalmente, el posicionamiento de la lata sin colisionar con el extremo del armario como requisito de la ley de Fitts (desde B hasta C).

Existen pocas garantías de que el tiempo necesario para realizar un movimiento complejo se pueda modelar con precisión mediante su descomposición en movimientos primitivos más simples. Sin embargo, incluso un modelo aproximado resulta probablemente más adecuado que la alternativa de comparar el rendimiento obtenido en la realización de las tareas sin incluir ninguna medida de la dificultad de las mismas. Cuanto mejor sea el modelo de la tarea, menos variabilidad no relacionada con la tarea se observará, reduciendo el efecto de factores externos en los resultados y proporcionando un mayor control sobre los experimentos.

2.3 Evaluación Experimental en Robótica Asistencial

Tradicionalmente, los desarrolladores de sistema robóticos han llevado a cabo experimentos exclusivamente sobre los robots físicos y sus algoritmos de control. Por tanto, se han mostrado principalmente interesados en medidas de rendimiento relacionadas con el tiempo para finalizar la tarea, precisión y consumo de energía. En consecuencia, los experimentos realizados se centran en torno a dichas métricas. Conforme los robots se van incorporando al mundo real, resulta necesario realizar experimentos que involucren a personas ajenas a los propios desarrolladores del sistema robótico.

Los experimentos controlados se utilizan para comparar las condiciones en las que se han desarrollado los experimentos mediante medidas cuantitativas de rendimiento. Asimismo, los estudios piloto son versiones a escala reducida de un experimento. Durante el ciclo de evaluación experimental de los robots asistenciales, los estudios

piloto se deben realizar primero con personas sin discapacidad, con las capacidades cognitivas plenamente funcionales. Este tipo de participantes puede proporcionar medidas de normalización o un límite superior de rendimiento en el desempeño de la ejecución de la tarea para el usuario final. Los estudios piloto presentan la finalidad de comprobar el protocolo experimental y verificar que los datos resultantes obtenidos concuerden con las medidas de rendimiento previstas (K.M. Tsui, and Yanco H.A., 2009). Estos datos pueden estar referidos tanto a medidas cuantitativas como, por ejemplo, el tiempo de ejecución del ensayo o el nivel de atención, como a medidas cualitativas relacionadas con el grado de aceptación de una determinada interfaz o sugerencias de mejoras para futuros desarrollo.

De este modo, controlar un robot asistencial se presenta como una tarea mucho más compleja que los modelos expuestos en las secciones anteriormente. El campo de la robótica asistencial ha sido desarrollado para diversos entornos, que abarcan desde el cuidado de personas mayores hasta la rehabilitación clínica, donde puede tener beneficios terapéuticos. Sin embargo, trasladarse de un entorno de investigación a aplicaciones clínicas implica un desafío significativo (Tsui, K., Yanco, H., Kontak, D. & Beliveau, L., 2008). Por lo tanto, es importante para las tecnologías robóticas asistenciales incluir a los usuarios finales en el diseño y evaluación de los experimentos con la finalidad de abordar adecuadamente sus necesidades.

Debido a la amplia variedad de aplicaciones asociadas a la robótica asistencial, existe una falta de procedimientos universales para medir el rendimiento, puesto que cada campo de aplicación requiere sus propias medidas específicas. La mayoría de las métricas no se adaptan a todos los dominios. También es deseable utilizar medidas fiables para determinar el rendimiento durante la evaluación clínica que proporcionen validez a los dispositivos robóticos asistenciales (K. Tsui, H. Yanco, D. J. Feil-Seifer, and M. J. Mataric, Aug 2008).

Otra dificultad añadida consiste en lograr una cantidad suficiente de datos con el hardware real y sujetos potenciales heterogéneos. Por tanto, resulta complicado determinar cuál es la mejor metodología para la evaluación experimental en robótica asistencial, en lo referente a la interacción hombre-robot (HRI).

No obstante, aunque muchos proyectos se encuentran todavía en fase de desarrollo, varias plataformas robóticas han realizado la transición desde los laboratorios de investigación hasta el entorno clínico. Un ejemplo lo constituyen dos manipuladores robóticos comerciales montados sobre silla de ruedas: la plataforma robótica asistencial denominada Manus ARM, mostrada en la Figura 2.4 a), y el sistema robótico asistencial conocido como Raptor ARM, observar Figura 2.4 b).

a)



b)



Figura 2.4: a) Manipulador robótico Asistencial Manus ARM montado sobre silla de ruedas. b) Plataforma robótica asistencial denominada Raptor ARM.

2.3.1 Estrategias para la Evaluación Experimental

Existen diversos tipos de estrategias para la aplicación de metodologías experimentales en el campo de la robótica asistencial. Estas estrategias comprenden desde el desarrollo de experimentos controlados sobre dispositivos de entrada hasta estudios de utilidad del sistema robótico completo. Un estadio intermedio consiste en el desarrollo de un entorno virtual y la utilización de un robot simplificado para la realización de tareas cotidianas por usuarios reales. La Figura 2.5 muestra los diferentes tipos de estrategias utilizadas para el diseño de los experimentos, desde aplicaciones más realistas hasta tareas que ofrecen un mayor control de las diferentes variables de evaluación experimental.

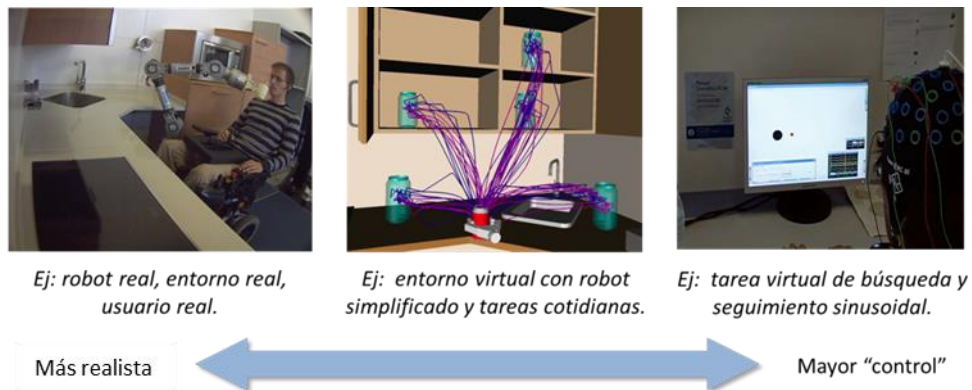


Figura 2.5: Diferentes tipos de estrategias para la evaluación experimental en robótica asistencial.

Las principales características asociadas a estas tres distintas estrategias se exponen a continuación:

- **Experimentos controlados:** Este tipo de experimentos se corresponden con extensiones de la ley de Fitts y *Steering Law*. Típicamente se restringen a entornos unidimensionales o bidimensionales (1D-2D), donde el tiempo medio para completar un movimiento se encuentra ampliamente cuantificado por los parámetros propios de la tarea (d , w , D , W).
- **Robot y entorno simplificados:** Se caracterizan por ser entornos de oclusión visual reducida y la utilización de un robot con cinemática y dinámica simplificada. No obstante, la falta de percepción de la profundidad puede convertirse en un factor limitante.
- **Experimentos realistas:** Para la evaluación de dichos experimentos se requiere la utilización de un sistema robótico completo y la inclusión de usuarios finales reales resulta imperativa. La realización de tareas complejas mediante la plataforma robótica física implica baja reproducibilidad y repetitividad de los experimentos.

2.3.2 Características principales de Diseño de los Experimentos

Este trabajo se fundamenta en el desarrollo de una plataforma virtual de simulación que permita la evaluación experimental en el ámbito de la robótica asistencial. Las características principales de diseño que se han seleccionado se corresponden con las de un estadio intermedio entre los experimentos controlados y los estudios de utilidad de sistemas robóticos. Estas características son las que se exponen a continuación.

2.3.2.1 Robot Asistencial Simplificado

Para facilitar el control en el manejo del robot por parte del usuario final, se ha optado por la utilización de una mano robótica simplificada, como se observa en la Figura 2.6. Esta mano robótica sostiene de forma permanente una lata de refresco con la finalidad de posicionarla en el emplazamiento requerido para cada tarea.

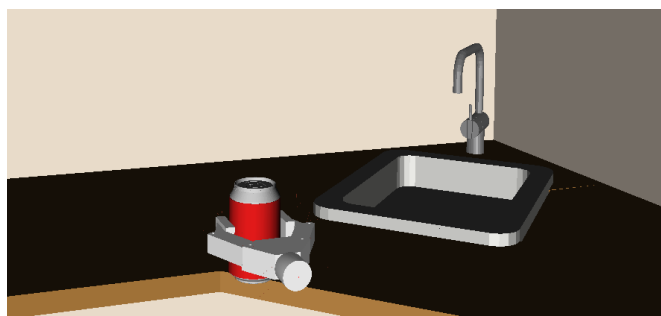


Figura 2.6: Mano robótica simplificada sosteniendo una lata de refresco.

2.3.2.2 Entorno Virtual para la Evaluación Experimental

El entorno virtual de simulación se modela teniendo en cuenta la estructura y dimensiones del entorno de pruebas de la cocina para robots asistenciales que se dispone en la Universidad Carlos III de Madrid. La Figura 2.7 a) muestra la distribución del mobiliario en el entorno real de pruebas, correspondiente a una cocina de uso doméstico; mientras que la Figura 2.7 b) representa el modelo virtual de dicha cocina, utilizado para el entorno de pruebas en simulación.

a)



b)



Figura 2.7: a) Entorno de pruebas real correspondiente a la cocina para robots asistenciales de la UC3M. b) Entorno virtual de pruebas, utilizado en simulación, que representa el modelo real de la cocina.

2.3.2.3 Tareas Seleccionadas y Representación de Objetivos

Las tareas escogidas, que representan actividades diarias comunes, consisten en el movimiento y posicionamiento de una lata de refresco sobre un estante del armario o sobre la encimera de la cocina. El emplazamiento final, para el posicionamiento de dicha lata, está representado por un objetivo con la misma geometría de la lata de refresco, semitransparente y con proporciones sobredimensionadas, tal y como se aprecia en la Figura 2.8.

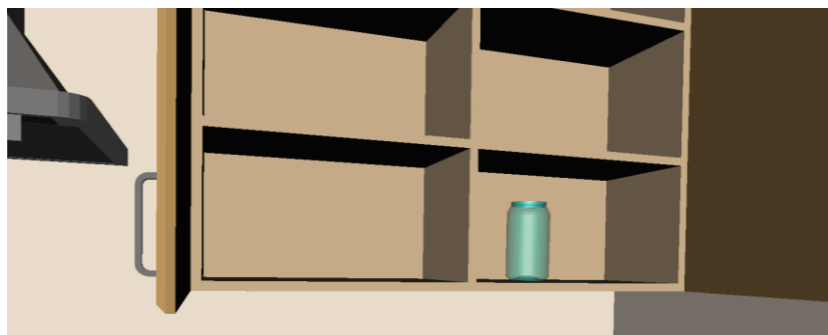


Figura 2.8: Representación del objetivo en forma de lata de refresco semitransparente y sobredimensionada sobre un estante del armario de la cocina.

2.3.2.4 Participantes en el Estudio Piloto

Todos los usuarios finales involucrados en el estudio piloto realizado no presentan ningún tipo de discapacidad física o psíquica. Los movimientos no intencionados durante la ejecución de las tareas, asociados a una discapacidad genérica, se introducen mediante la utilización del módulo software generador de ruido gaussiano.

Capítulo 3

Entorno Virtual de Experimentación

3.1 Introducción

Las metodologías expuestas en el capítulo anterior, aunque resultan útiles para la evaluación del rendimiento de los dispositivos de entrada utilizados por los usuarios, no son adecuadas cuando se pretende evaluar un sistema de mayor complejidad que comprenda al usuario y al robot asistencial. Un ejemplo de estos sistemas son los que incluyen esquemas de control compartido, en los que el robot toma decisiones basándose tanto en la entrada que recibe por parte del usuario como en su propia interpretación de la tarea. En este caso es conveniente evaluar el sistema en alguna plataforma que se aproxime al entorno real y a las actividades de la vida diaria deseadas, manteniendo al mismo tiempo el máximo control posible sobre la metodología del experimento (M.F. Stoelen et al., 2011).

En este capítulo se describe la arquitectura software de la plataforma virtual de experimentación, desarrollada para dicho propósito, junto con los dispositivos hardware empleados y los programas software necesarios.

3.2 Dispositivos Hardware

En este apartado se realiza una descripción de los sistemas hardware empleados mediante los cuales se ha logrado implementar la arquitectura virtual de experimentación desarrollada en este proyecto. Estos elementos hardware se enumeran a continuación.

3.2.1 Dispositivo de Entrada al Ordenador

Como dispositivo de entrada al ordenador se utiliza un ratón 3D denominado SpaceNavigator², el cual se muestra en la Figura 3.1 a). El SpaceNavigator es un dispositivo 3D, que actúa como interfaz humana, diseñado especialmente para la manipulación y navegación en entornos tridimensionales complejos. Dispone de seis grados de libertad, de los cuales tres permiten definir la posición y los otros tres la orientación, como se puede observar en la Figura 3.1 b). Además posee dos botones para producir los eventos típicos de los ratones tradicionales.

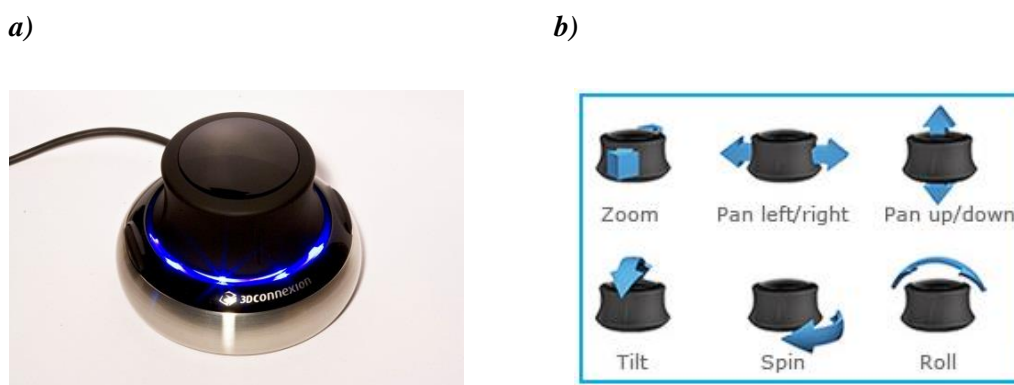


Figura 3.1: a) SpaceNavigator: ratón 3D desarrollado para manipulación de entornos tridimensionales. b) Descripción de los seis grados de libertad del SpaceNavigator, tres para posicionamiento y tres para orientación.

² <http://www.3dconnexion.com/products/spacenavigator.html> (Última revisión: 02-07-2013)

3.2.2 Entorno Físico de Experimentación

La configuración del entorno físico de experimentación puede observarse en la Figura 3.2, donde el dispositivo de entrada utilizado se corresponde con el SpaceNavigator. La simulación del robot simplificado en el entorno virtual se ha realizado empleando una CPU estándar dotada de dos procesadores AMD Opteron™ 258 a 2.20 GHz y 2.75 GB útiles de memoria RAM DDR3, utilizando toda su capacidad de rendimiento durante la ejecución de los experimentos. El entorno de simulación virtual y el conjunto de datos relativos al experimento en curso se visualizan mediante dos monitores TFT convencionales de 22 y 19 pulgadas, respectivamente.



Figura 3.2: Configuración del entorno físico de experimentación.

3.3 Herramientas Software

En este apartado se realiza una descripción de los programas software empleados mediante los cuales se ha logrado la correcta implementación de la arquitectura virtual de experimentación que se expone en este proyecto. Todos los programas software necesarios se detallan a continuación.

3.3.1 Plataforma Software

La implementación software de los distintos módulos que integran la arquitectura de experimentación lleva asociada la utilización de los siguientes programas.

3.3.1.1 CMake

CMake³ es una herramienta multiplataforma de generación o automatización de código. Diseñada para construir, probar y empaquetar software, se utiliza para controlar el proceso de compilación del software usando ficheros de configuración sencillos e independientes de la plataforma.

CMake genera espacios de trabajo que pueden usarse en el entorno de desarrollo deseado y soporta la generación de ficheros para los sistemas operativos Linux, Windows y Mac OS X. Esta característica facilita el mantenimiento y elimina la necesidad de tener varios conjuntos de ficheros para cada plataforma.

3.3.1.2 Eigen

Eigen⁴ es una librería de plantillas de código fuente en C++ destinada a aplicaciones de álgebra lineal. Compatible con todos los tamaños de matrices y tipos numéricos estándar, proporciona versatilidad y rapidez de cómputo.

Dispone de algoritmos seleccionados por su fiabilidad y funciones especiales, tales como optimizadores no lineales, solucionadores de algoritmos, FFT, etc.

3.3.1.3 KDL (Kinematics and Dynamics Library)

KDL⁵ es una librería dedicada a la cinemática y la dinámica, distribuida por el proyecto Orocos⁶. Desarrolla un marco de aplicación independiente para el modelado y el cálculo de cadenas cinemáticas, tales como robots, modelos biomecánicos humanos, figuras animadas por ordenador, herramientas computacionales, etc.

³ <http://www.cmake.org/> (Última revisión: 02-07-2013)

⁴ <http://eigen.tuxfamily.org/> (Última revisión: 02-07-2013)

⁵ <http://www.orocos.org/kdl> (Última revisión: 02-07-2013)

⁶ <http://www.orocos.org/> (Última revisión: 02-07-2013)

Además, proporciona librerías de clases para objetos geométricos que ofrecen un excelente soporte para trabajar con vectores, puntos y sistemas de referencia, así como con cadenas cinemáticas de diversos tipos (serie, humanoide, paralelo o móvil) y sus especificaciones de movimiento e interpolación.

3.3.1.4 Boost

Boost⁷ es un conjunto de librerías de código fuente abierto preparadas para extender las capacidades del lenguaje de programación C++. Su licencia permite que sea utilizada en cualquier tipo de proyectos, ya sean comerciales o no.

El diseño e implementación de Boost permiten que sea utilizada en un amplio espectro de aplicaciones y plataformas, con multitud de compiladores distintos. Abarca desde librerías de propósito general hasta abstracciones del sistema operativo. Con el objetivo de alcanzar el mayor rendimiento y flexibilidad se realiza un uso intensivo de plantillas.

3.3.1.5 OpenRAVE (Open Robotics Automation Virtual Environment)

OpenRAVE⁸ (Diankov, 2010), desarrollado en el Instituto de Robótica (Universidad Carnegie Mellon), se presenta como una arquitectura software multiplataforma de código abierto focalizada en el desarrollo de aplicaciones para el mundo real de robots autónomos.

OpenRAVE proporciona un entorno para la realización de pruebas, desarrollo e implementación de algoritmos de planificación de movimientos para aplicaciones robóticas, e incluye una robusta integración entre simulación, visualización, algoritmos de planificación y control, y lenguajes de programación. Fundamentalmente se centra en la simulación y el análisis de la información cinemática y geométrica relacionada

⁷ <http://www.boost.org/> (Última revisión: 02-07-2013)

⁸ <http://openrave.org/> (Última revisión: 02-07-2013)

con la planificación de movimientos. Presenta las siguientes características generales en su diseño:

- Arquitectura basada en múltiples *plugins* que le confiere mayor funcionalidad y permite que su núcleo sea lo más sencillo posible.
- Un núcleo que puede ser utilizado tanto como entorno físico de simulación 3D, como módulo controlador de las operaciones cinemáticas y dinámicas asociadas a los robots, o como módulo de planificación para tareas de manipulación.
- Un diseño integrado para el control y monitorización de los movimientos ejecutados por los robots en tiempo real.
- Un protocolo de red que proporciona la posibilidad de interactuar con lenguajes de programación interpretados como Matlab, Octave y Python.

La arquitectura de OpenRAVE, mostrada en la Figura 3.3, simplifica la implementación y prueba de algoritmos de planificación. Asimismo, permite a los nuevos módulos creados por los usuarios interactuar de forma sencilla con otros módulos existentes a través de una interfaz de programación de aplicaciones (API). La API principal está realizada en C++ y utiliza las librerías Boost C++ como una base robusta para las estructuras de almacenamiento y la gestión a bajo nivel.

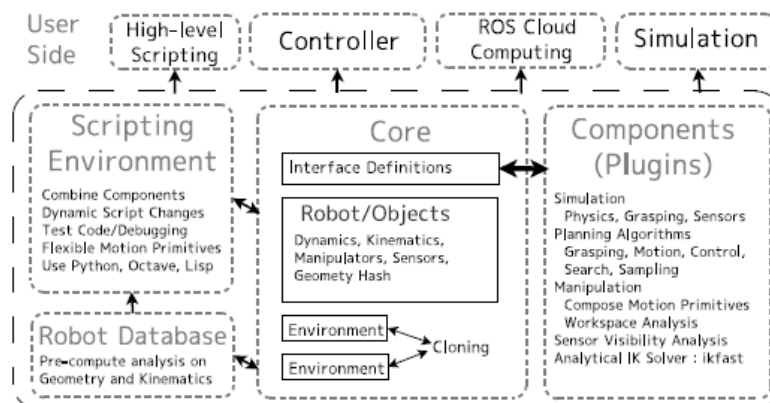


Figura 3.3: La arquitectura de OpenRAVE se divide en cuatro capas principales: Núcleo, *Plugins*, Entorno de programación y Base de Datos de Robots.

La estructura de los sistemas operativos, lenguajes de programación y herramientas de depuración ha motivado la división de OpenRAVE en cuatro capas principales: la capa del núcleo de la API, la capa de *plugins* para extender la funcionalidad, una capa para el análisis de secuencias de comandos en tiempo de ejecución y una capa de base de datos para la generación y la rápida obtención de parámetros característicos de diversos robots.

3.3.1.5.1 Capa Core

Está compuesta por un conjunto de clases, que a modo de interfaz, definen cómo los *plugins* intercambian información. También proporciona un entorno que combina motores físicos, detectores de colisión, visores de simulación, propiedades cinemáticas y dinámicas que determinan, tanto el estado de los robots como el de los objetos con los que interactúan.

3.3.1.5.2 Capa Plugins

La arquitectura basada en *plugins* permite la continua creación de nuevos componentes que mejoran las especificaciones generales. Cada *plugin*, que representa una interfaz estándar, puede ser distribuido y cargado de forma dinámica sin necesidad de recompilar el núcleo. Entre los diferentes tipos de *plugins* que pueden ser implementados se encuentran los que se mencionan a continuación:

- **Planificadores.** Su propósito básico es el de encontrar un camino desde una configuración inicial del robot hasta alcanzar un estado final, atendiendo ciertas restricciones (equilibrio dinámico, obstáculos, etc.).
- **Controladores.** Conectados a los robots proporcionan la funcionalidad para establecer una trayectoria y obtener el estado actual del robot en el mundo real o virtual.

- **Motor Físico.** Proporciona la capacidad de utilizar diversos tipos de librerías físicas personalizadas o existentes (Bullet, ODE, etc.). Vinculadas a un entorno de simulación, permiten mover correctamente los objetos de la escena, dotándolos con características cinemáticas y dinámicas.
- **Detectores de Colisión.** Aportan una amplia gama de funciones de proximidad y colisión que presentan información sobre puntos y eslabones de contacto, número de colisiones, profundidad de penetración, etc.
- **Cinemática Inversa.** Resuelve de forma analítica la cinemática inversa de diversos tipos de robots, observar Figura 3.4. Atendiendo al subconjunto de eslabones que lo conforman, proporciona soluciones que pueden ser utilizadas como entrada para los planificadores de manipulación.

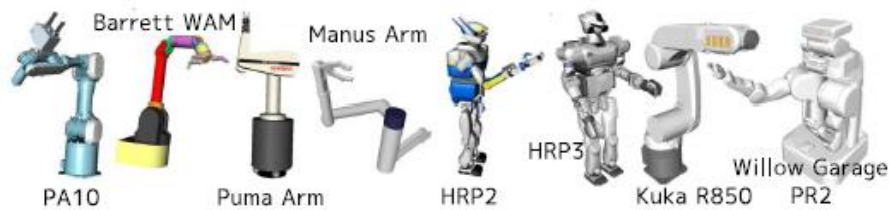


Figura 3.4: Diversos tipos de plataformas robóticas cuya cinemática inversa puede ser resuelta con el algoritmo implementado en OpenRAVE (IKFast).

- **Interfaz gráfica.** Ofrece una visualización en 3D del estado actual del entorno de simulación y permite al usuario cambiar la posición de los objetos, interactuar con las articulaciones de los robots, rotar vistas, etc.
- **Sensores.** Miden las propiedades físicas del entorno y devuelven la información en un formato estándar, dependiendo del tipo de sensor (cámara, láser, fuerza-par, etc.). Cada sensor está asociado con una posición particular en el espacio y puede ser conectado a cualquier parte del robot.

3.3.1.5.3 Capa Scripting

Proporciona entornos de secuencias de comandos (*scripts*) para Python y Octave/Matlab. La API de Python se comunica directamente con el núcleo mediante llamadas a memoria, confiriéndole una extrema rapidez, mientras que el protocolo de *scripts* de Octave/Matlab envía los comandos a través de conexiones TCP/IP, tal y como se refleja en la Figura 3.5. Los *scripts* permiten realizar modificaciones sobre cualquier aspecto del entorno en tiempo real, sin necesidad de detener la ejecución de los algoritmos. La utilización de estos lenguajes debe considerarse como parte integral de todo el sistema, no como una sustitución de la API de C++.

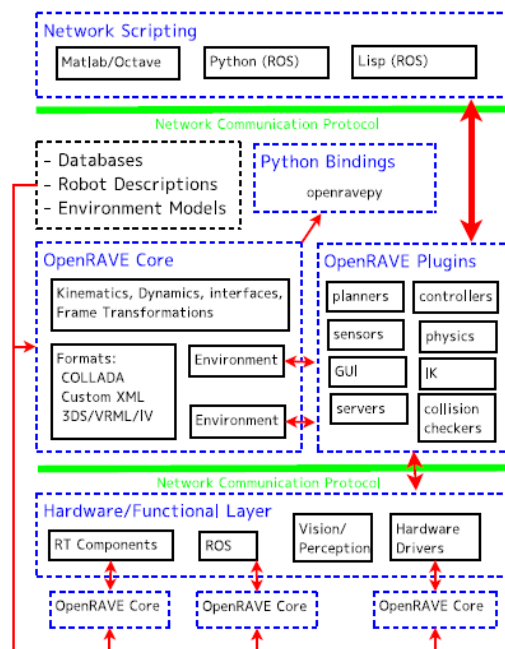


Figura 3.5: La arquitectura de OpenRAVE con los protocolos de comunicación de red asociados.

3.3.1.5.4 Capa Robot DataBase

Implementa una planificación de base de datos de robots y proporciona una sencilla interfaz para el acceso y generación de sus parámetros característicos. La base de datos consiste principalmente en el análisis de propiedades cinemáticas, cuasi-estáticas, dinámicas y geométricas del robot, así como las propiedades la tarea asociada.

Estas cuatro capas combinadas entre si ofrecen un conjunto de herramientas que, incluso los usuarios no expertos, pueden utilizarlas para el análisis de sus respectivos problemas sin necesidad de poseer un profundo conocimiento de los sistemas software, algoritmos de planificación y manipulación o motores físicos.

La naturaleza dual de OpenRAVE permite ser utilizado tanto como controlador, como simulador o ambos simultáneamente. Su diseño integra interfaces para el control en tiempo real de sistemas robóticos. Todos los algoritmos y *scripts* diseñados para trabajar en el entorno virtual pueden, teóricamente, funcionar en el mundo real simplemente sustituyendo los controladores virtuales por reales.

3.3.2 Comunicación de Datos

Para las comunicaciones de datos entre los distintos módulos que componen la plataforma virtual de experimentación se utilizan las siguientes herramientas:

3.3.2.1 ACE™ (Adaptative Communication Environment)

El Ambiente de Comunicación Adaptable (ACE⁹) simplifica varios aspectos de la programación en red. Ofrece un juego de objetos de alto rendimiento orientados a clases de C++, diseñadas para dirigir las complejidades inherentes y desafíos de la programación en red previniendo errores comunes.

3.3.2.2 YARP (Yet Another Robot Platform)

La librería YARP¹⁰ (Fitzpatrick, P., Metta, G. and Natale, L., 2007) se describe como una arquitectura software de código abierto para robots que ayuda a organizar la

⁹ <http://www.cs.wustl.edu/~schmidt/ACE.html> (Última revisión: 02-07-2013)

¹⁰ <http://eris.liralab.it/yarp/> (Última revisión: 02-07-2013)

comunicación entre los sensores, procesadores y actuadores, permitiendo un acoplamiento flexible entre ellos.

Incluye un modelo de comunicación que permite un transporte neutral, de modo que el flujo de datos se desacopla de las características intrínsecas de las redes subyacentes y protocolos en uso.

YARP está diseñada para operar conjuntamente con otras arquitecturas, lo que resulta importante para su longevidad a largo plazo. El uso de YARP facilita el intercambio de código entre investigadores, principalmente cuando el tiempo necesario para desarrollar una plataforma robótica y usarla para investigación es un factor crítico.

YARP fue inicialmente desarrollada por el MIT y LiraLab (Universidad de Génova) para plataformas robóticas humanoides y cuenta también con el soporte del Instituto Italiano de Tecnología (IIT). Presenta como características principales ser una librería liviana y fácil de usar para los principiantes, a la vez que dispone de un código complejo y especialmente eficiente para ser utilizado por los usuarios más expertos. YARP es compatible con Linux, Windows y Mac OS X y depende exclusivamente de ACE para su implementación.

La interfaz nativa de YARP está basada en el lenguaje de programación orientado a objetos C++. No obstante, mediante la herramienta de libre distribución denominada SWIG¹¹ es posible generar *wrappers* para el código fuente en C++, lo que permite su utilización con distintos lenguajes de programación como Python, Java, Matlab (vía Java), TCL, Lisp, C#, etc.

La comunicación mediante YARP generalmente sigue el patrón de diseño *Observer*. Dispone de objetos especiales, denominados puertos, que entregan mensajes a cualquier número de observadores (otros puertos), empleando cualquier número de procesos distribuidos a través de diferentes máquinas físicas. Dichos procesos pueden incluso ser ejecutados bajo distintos sistemas operativos y ser implementados utilizando distintos lenguajes de programación, observar Figura 3.6.

¹¹ <http://www.swig.org/> (Última revisión: 02-07-2013)

YARP soporta el uso de los siguientes tipos de protocolo de comunicación, lo que permite un mejor aprovechamiento de las cualidades más destacadas de cada uno de ellos:

- TCP/IP: protocolo fiable que puede ser utilizado para garantizar la recepción de los mensajes enviados.
- UDP: es un protocolo más rápido que el TCP/IP, utilizado para conexiones punto a punto, pero que no ofrece garantías en la recepción de mensajes.
- Multicast: protocolo utilizado para crear conexiones entre múltiples puntos; resulta eficaz para distribuir la misma información a un gran número de objetivos, por ejemplo, las imágenes de una cámara.
- Memoria compartida: empleado para conexiones locales. Este protocolo se selecciona automáticamente siempre que sea posible, sin la necesidad de ninguna intervención por parte del programador.

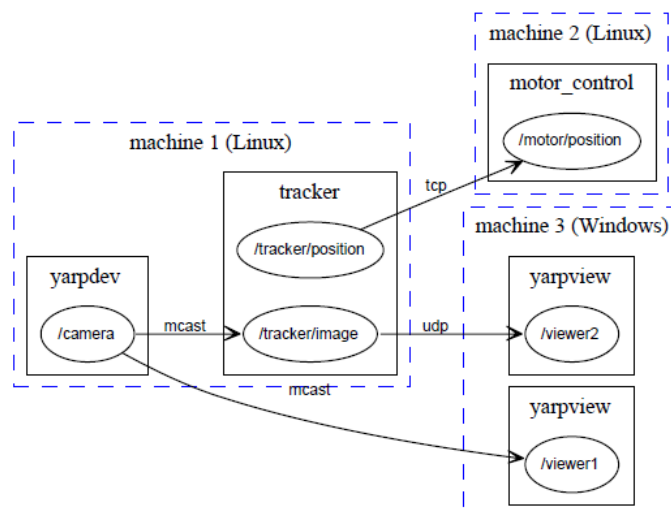


Figura 3.6: Ejemplo de una red de puertos en YARP con protocolos de comunicación diferentes. Los puertos pertenecen a diversos procesos que pueden ser ejecutados en diferentes ordenadores con sistemas operativos distintos.

YARP presenta herramientas útiles aplicadas a la línea de comandos para realizar operaciones con la red de puertos, así como utilidades gráficas y basadas en la web.

3.4 Arquitectura Software del Entorno de Experimentación

La arquitectura software de los experimentos se compone fundamentalmente de siete módulos programados en código fuente C++ e interconectados entre si mediante YARP, utilizando el protocolo de comunicación TCP/IP. Los módulos integrantes se denominan: *spacnavigator*, *experiment manager*, *uci proximity*, *openrave driver*, *trajectory player*, *window feedback* y *trajectory visualizer*.

La Figura 3.7 muestra el diagrama de bloques de la plataforma virtual de experimentación. La implementación y verificación del funcionamiento adecuado de todos los módulos se ha realizado bajo el sistema operativo Ubuntu 10.10 Desktop (Linux) de 64 bits.

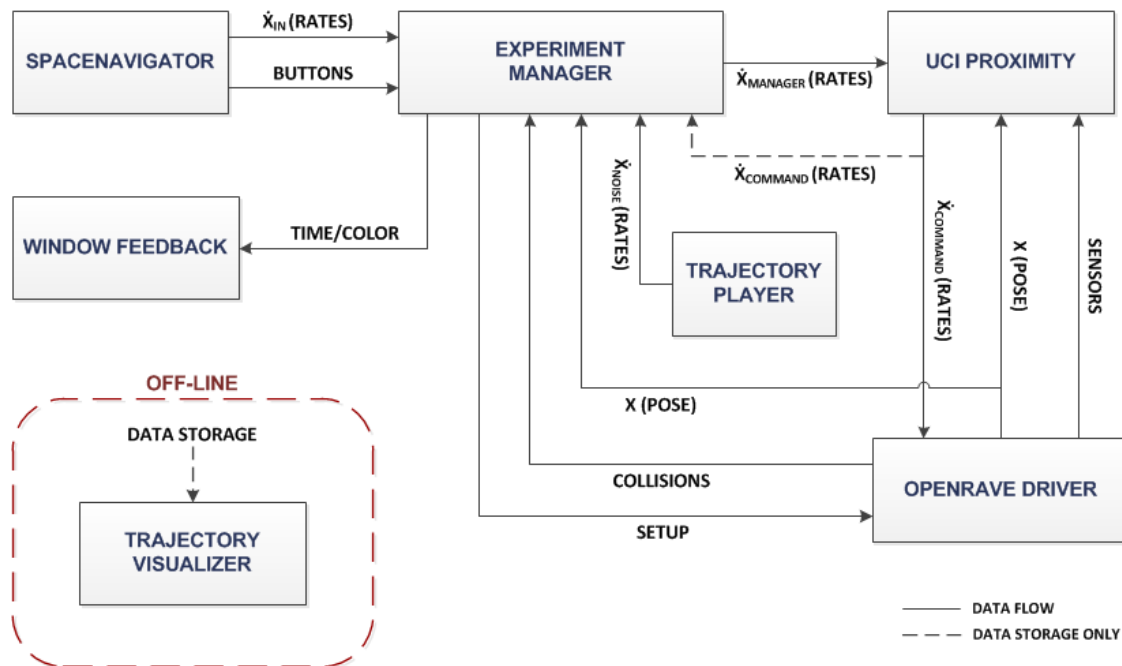


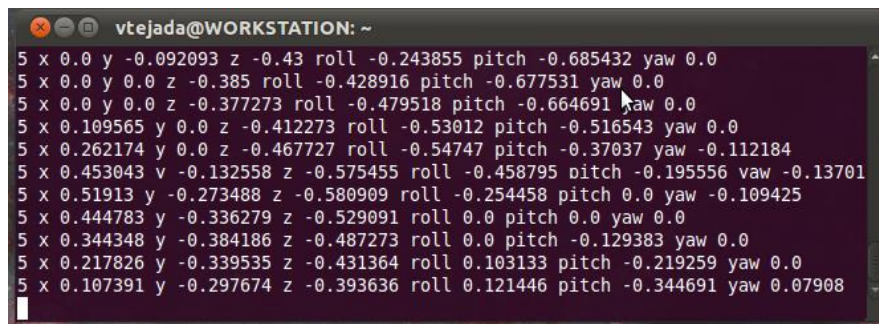
Figura 3.7: Módulos software y conexiones YARP, utilizados para realizar experimentos virtuales, que involucran diferentes modos de control de actividades de la vida diaria.

La secuencia de instrucciones para ejecutar la arquitectura software del entorno de experimentación, la identificación de los puertos YARP asociados al flujo de datos y su modo de conexión se recogen en el Anexo A del documento.

3.4.1 Módulo *SpaceNavigator*

Este módulo se identifica con el dispositivo electrónico de entrada y se corresponde con el ratón 3D, SpaceNavigator, que permite a los usuarios realizar movimientos precisos de posicionamiento y orientación en el espacio.

Dicho módulo se comunica unidireccionalmente con el administrador de experimentos mediante dos puertos YARP de salida, \dot{X}_{in} y *buttons*. Estos puertos transfieren las velocidades transmitidas por los sujetos (*x, y, z, roll, pitch, yaw*) y envían los eventos producidos cuando se presionan los botones por parte de los usuarios, como se puede observar en la Figura 3.8.

A terminal window titled 'vtejada@WORKSTATION: ~' displays a list of 13 lines of data. Each line represents a set of velocity parameters for a subject, including x, y, z coordinates, roll, pitch, and yaw values. The data is as follows:

```
5 x 0.0 y -0.092093 z -0.43 roll -0.243855 pitch -0.685432 yaw 0.0
5 x 0.0 y 0.0 z -0.385 roll -0.428916 pitch -0.677531 yaw 0.0
5 x 0.0 y 0.0 z -0.377273 roll -0.479518 pitch -0.664691 yaw 0.0
5 x 0.109565 y 0.0 z -0.412273 roll -0.53012 pitch -0.516543 yaw 0.0
5 x 0.262174 y 0.0 z -0.467727 roll -0.54747 pitch -0.37037 yaw -0.112184
5 x 0.453043 y -0.132558 z -0.575455 roll -0.458795 pitch -0.195556 yaw -0.13701
5 x 0.51913 y -0.273488 z -0.580909 roll -0.254458 pitch 0.0 yaw -0.109425
5 x 0.444783 y -0.336279 z -0.529091 roll 0.0 pitch 0.0 yaw 0.0
5 x 0.344348 y -0.384186 z -0.487273 roll 0.0 pitch -0.129383 yaw 0.0
5 x 0.217826 y -0.339535 z -0.431364 roll 0.103133 pitch -0.219259 yaw 0.0
5 x 0.107391 y -0.297674 z -0.393636 roll 0.121446 pitch -0.344691 yaw 0.07908
```

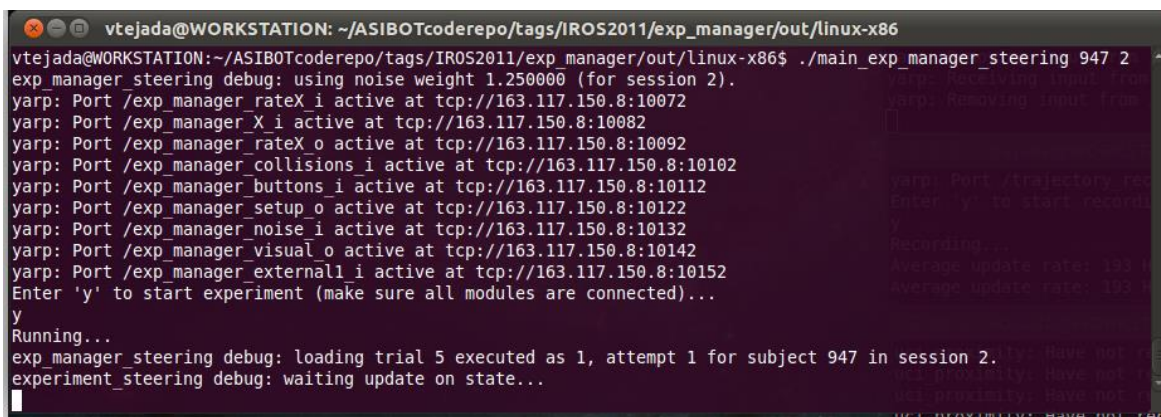
Figura 3.8: Módulo *spacenavigator* que identifica los velocidades transferidas al robot por los usuarios.

3.4.2 Módulo *Experiment Manager*

El módulo denominado *experiment manager* tiene como finalidad principal la gestión de las variables seleccionadas para la correcta evaluación del rendimiento de los ensayos: velocidad de traslación, velocidad de rotación, posición, orientación, tiempo de simulación, ruido gaussiano y colisiones. Dichas variables de estudio interactúan con los diferentes módulos que constituyen la arquitectura y determinan el funcionamiento global de todos los procesos involucrados en la ejecución del experimento.

No obstante, también presenta como cometido el almacenamiento de la métrica necesaria para representar, mediante el módulo *trajectory visualizer*, las trayectorias realizadas por cada sujeto y su posterior análisis.

Requiere, como parámetros de entrada, el identificador de cada usuario y la sesión correspondiente a la cuantía de ruido gaussiano que se desea introducir. Ofrece información en cada instante del ensayo en curso, su número de ejecución e intentos que se realizan hasta alcanzar el objetivo, como se puede observar en la Figura 3.9



```
vtejada@WORKSTATION: ~/ASIBOTcoderepo/tags/IROS2011/exp_manager/out/linux-x86
vtejada@WORKSTATION:~/ASIBOTcoderepo/tags/IROS2011/exp_manager/out/linux-x86$ ./main_exp_manager_steering 947 2
exp_manager_steering debug: using noise weight 1.250000 (for session 2).
yarp: Port /exp_manager_rateX_i active at tcp://163.117.150.8:10072
yarp: Port /exp_manager_X_i active at tcp://163.117.150.8:10082
yarp: Port /exp_manager_rateX_o active at tcp://163.117.150.8:10092
yarp: Port /exp_manager_collisions_i active at tcp://163.117.150.8:10102
yarp: Port /exp_manager_buttons_i active at tcp://163.117.150.8:10112
yarp: Port /exp_manager_setup_o active at tcp://163.117.150.8:10122
yarp: Port /exp_manager_noise_i active at tcp://163.117.150.8:10132
yarp: Port /exp_manager_visual_o active at tcp://163.117.150.8:10142
yarp: Port /exp_manager_external1_i active at tcp://163.117.150.8:10152
Enter 'y' to start experiment (make sure all modules are connected)...
y
Running...
exp_manager_steering debug: loading trial 5 executed as 1, attempt 1 for subject 947 in session 2.
experiment_steering debug: waiting update on state...
```

Figura 3.9: Módulo *experiment manager* que ofrece información sobre las características del experimento en ejecución.

3.4.3 Módulo *Uci Proximity*

El módulo que implementa el control compartido, denotado mediante *uci proximity*, ejerce un control sobre la velocidad cuando existe un riesgo evidente de colisión, con el objetivo de mejorar el rendimiento de los usuarios.

Su funcionamiento se basa en la determinación del sensor que suministra el rango de medidas más próximo a un posible obstáculo. Se obtiene la proyección del vector de datos en relación con la velocidad de traslación de la mano robótica y se ralentiza dicha velocidad en función de un tiempo de colisión preestablecido.

La implementación del control compartido se basa en un conjunto de 20 sensores infrarrojos de proximidad distribuidos por toda la mano robótica virtual, ver Figura 3.10.

El rango de los sensores infrarrojos oscila entre 40 y 300 milímetros, a los que se les añade ruido gaussiano con una desviación estándar de 10 milímetros para asemejarlos a los sensores reales.

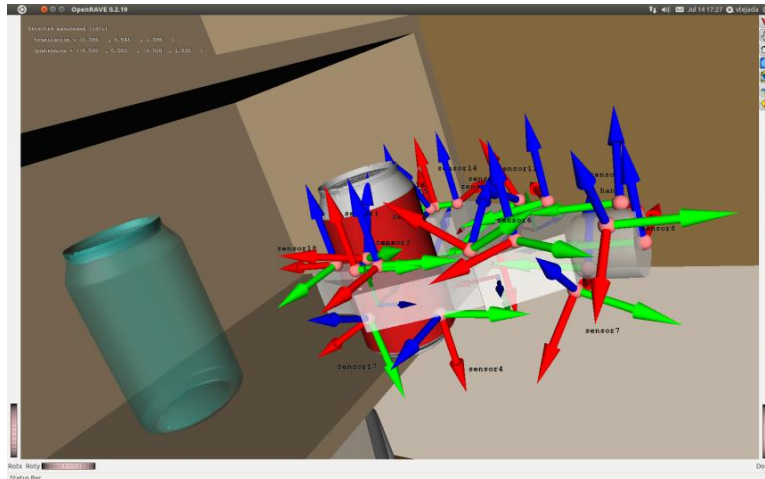


Figura 3.10: Distribución de los sensores infrarrojos por la mano robótica. El eje de color rojo indica la dirección de medida.

El esquema de control compartido implementado se muestra en la Tabla 3.1.

Algorithm 1 Simple shared control algorithm used for limiting translational velocities based on proximity sensors.

```

 $\Delta \vec{p} = k \vec{v}_{in}$ 
for  $i = 1$  to  $n$  do
   $proj_i = \frac{\Delta \vec{p} \cdot \vec{d}_i}{\|\vec{d}_i\|}$ 
   $ratio_i = \frac{proj_i}{\|\vec{d}_i\|}$ 
end for
 $ratio_{max} = \max_i(ratio_i)$ 
 $\vec{v}_{out} = \begin{cases} \vec{v}_{in} & \text{if } ratio_{max} \leq 1, \\ \frac{\vec{v}_{in}}{ratio_{max}} & \text{otherwise.} \end{cases}$ 

```

Tabla 3.1: Pseudocódigo correspondiente al algoritmo utilizado para la implementación del control compartido.

Este algoritmo limita la velocidad de la mano robótica comandada por el usuario, \vec{v}_{in} , en la dirección en la que los obstáculos son detectados, proporcionalmente a la distancia medida. La lectura \vec{d}_i del total de los n sensores que tenga la mayor

proyección cuando se produce un cambio de posición de la mano robótica ($\Delta\vec{p}$), en comparación con su propia magnitud, se utiliza en cada instante. La dirección de la velocidad transmitida se mantiene sin alteraciones.

Esto proporciona una primera aproximación rápida y razonable para conseguir una limitación de velocidad, en el espacio de trabajo, que cumpla con la finalidad de los experimentos.

Este módulo utiliza tanto las velocidades enviadas por el usuario, a través del puerto YARP, $\dot{X}_{manager}$, como la información proporcionada por los sensores para generar comandos para el robot. Dicha información es enviada por el módulo denominado *openrave driver* mediante el puerto YARP, *sensors*. Además, requiere como parámetro de entrada la elección del tiempo mínimo para que se produzca una colisión, k . La Figura 3.11 muestra el módulo *uci proximity* durante su ejecución.

```
vtejada@WORKSTATION:~/ASIBOTcoderepo/tags/IR052011/uci_proximity/out/linux-x86$ ./uci_proximity 1.6 0 0
TTC_min: 1.600000.
yarp: Port /uci_user_i active at tcp://163.117.150.8:10182
yarp: Port /uci_prox_i active at tcp://163.117.150.8:10192
yarp: Port /uci_trans_i active at tcp://163.117.150.8:10202
yarp: Port /uci_robo_o active at tcp://163.117.150.8:10212
uci_proximity: Have not received user input for more than 0.1 seconds, setting to zero.
uci_proximity: Have not received user input for more than 0.1 seconds, setting to zero.
uci_proximity: Have not received user input for more than 0.1 seconds, setting to zero.
uci_proximity: Have not received user input for more than 0.1 seconds, setting to zero.
```

Figura 3.11: Módulo *uci proximity*, en ejecución, con información relativa a la velocidades enviadas por el usuario. El parámetro k se ajusta a un valor de 1.6 s. El resto de parámetros se corresponden con valores de depuración del módulo.

3.4.4 Módulo *Openrave Driver*

El módulo *openrave driver* se identifica propiamente con el simulador dentro de la arquitectura de experimentos. Este módulo simula tareas simplificadas de la vida diaria en el entorno virtual y el robot asistencial mediante OpenRAVE.

Integra como entorno de experimentación el mobiliario correspondiente a la cocina de pruebas y un objetivo que ocupa, para cada ensayo, distintas posiciones con diferente grado de complejidad, como se puede observar en la Figura 3.12.

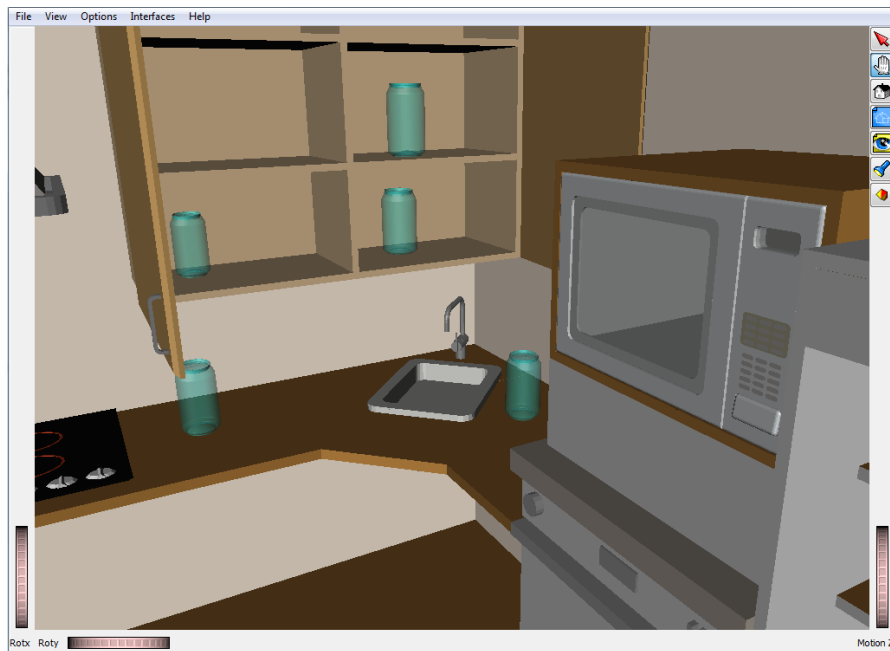


Figura 3.12: Ubicación de las distintas posiciones de los objetivos en el entorno de pruebas de la cocina virtual.

El propósito fundamental del experimento consiste en el posicionamiento de una lata de refresco, sostenida por una mano robótica, en el interior de un objetivo representado por una lata semitransparente de mayores dimensiones, tal y como se observa en la Figura 3.13.

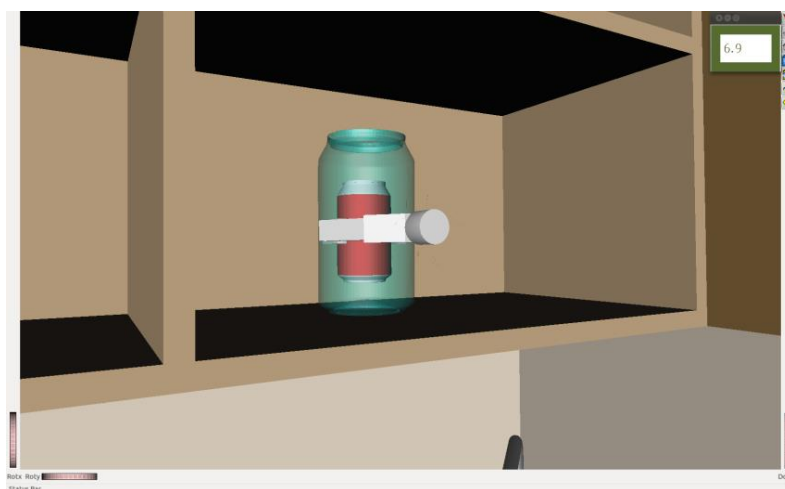


Figura 3.13: Posicionamiento de la lata de refresco en el interior del objetivo para cumplimentar la tarea.

En el experimento piloto realizado, para la validación de la arquitectura, se pretende que cada tarea se realice en el menor tiempo posible evitando cualquier tipo de colisión con el entorno. Para la consecución de dicho fin, se envían asiduamente las medidas correspondientes con los valores de posición y orientación de la lata, los datos adquiridos por sensores y colisiones detectadas con el entorno. Asimismo, se le proporciona a la mano robótica tanto las velocidades de desplazamiento como los datos de su posición de partida para cada ensayo.

La Figura 3.14 muestra el visor de simulación que OpenRAVE ofrece a los usuarios para la realización de los experimentos.



Figura 3.14: Visor de simulación en OpenRAVE que muestra el entorno virtual de experimentación: cocina, mano robótica y objetivo.

Este módulo puede ser configurado por el módulo *experiment manager* mediante la utilización del puerto YARP, *setup*, que devuelve al robot a su posición inicial. Incluye las funcionalidades necesarias para el envío de las medidas suministradas por los sensores al módulo que implementa el control compartido (*uci proximity*) a través del puerto YARP, *sensors*. La información relativa a la detección de colisiones de la mano robótica con el entorno se envía al módulo *experiment manager*, mediante el puerto YARP, *collisions*, y se utiliza típicamente para reiniciar el experimento.

En la Figura 3.15 se puede observar la información que proporciona este módulo durante la ejecución de los experimentos.

```
vtejada@WORKSTATION: ~/ASIBOTcoderepo/tags/IROS2011/exp_openrave_driver/
vtejada@WORKSTATION:~/ASIBOTcoderepo/tags/IROS2011/exp_openrave_driver/out/linux-x86 ~
./exp_openrave_driver
yarp: Port /exp_driver_rate_i active at tcp://163.117.150.8:10022
yarp: Port /exp_driver_setup_i active at tcp://163.117.150.8:10032
yarp: Port /exp_driver_collisions_o active at tcp://163.117.150.8:10042
yarp: Port /exp_driver_sensors_o active at tcp://163.117.150.8:10052
yarp: Port /exp_driver_pose_o active at tcp://163.117.150.8:10062
Loading file...
[environment-core.h:619] setting ode physics engine
[KinBody.cpp:3873] _ComputeInternalInformation manushand in 0.036996s
[KinBody.cpp:3873] _ComputeInternalInformation tool in 0.341243s
[KinBody.cpp:3873] _ComputeInternalInformation kitchen in 0.085519s
[KinBody.cpp:3873] _ComputeInternalInformation target1 in 0.341935s
[KinBody.cpp:3873] _ComputeInternalInformation target2 in 0.345476s
[KinBody.cpp:3873] _ComputeInternalInformation target3 in 0.411529s
[KinBody.cpp:3873] _ComputeInternalInformation target4 in 0.362589s
[KinBody.cpp:3873] _ComputeInternalInformation target5 in 0.364203s
[environment-core.h:668] setting ode collision checker
Running...
Average update rate: 56 Hz
Average update rate: 59 Hz
Average update rate: 38 Hz
Average update rate: 71 Hz
yarp: Sending output from /exp_driver_collisions_o to /exp_manager_collisions_i using tcp
yarp: Sending output from /exp_driver_pose_o to /exp_manager_X_i using tcp
yarp: Sending output from /exp_driver_pose_o to /uci_trans_i using tcp
yarp: Receiving input from /exp_manager_setup_o to /exp_driver_setup_i using tcp
yarp: Receiving input from /uci_robo_o to /exp_driver_rate_i using tcp
yarp: Sending output from /exp_driver_sensors_o to /uci_prox_i using tcp
Average update rate: 53 Hz
[KinBody.cpp:3873] _ComputeInternalInformation target5 in 0.627270s
target5 has been successfully added.
```

Figura 3.15: Módulo *openrave driver* en ejecución, mostrando información relativa al desarrollo de los experimentos.

La Figura 3.16 muestra la información ofrecida por el módulo cuando se produce una colisión con el entorno.

```
vtejada@WORKSTATION: ~/ASIBOTcoderepo/tags/IROS2011/exp_openrave_driver/
yarp: Sending output from /exp_driver_sensors_o to /uci_prox_i using tcp
Average update rate: 56 Hz
Average update rate: 50 Hz
[KinBody.cpp:3873] _ComputeInternalInformation target5 in 0.539700s
target5 has been successfully added.
Running...
Average update rate: 59 Hz
Average update rate: 48 Hz
ERROR: collision has been detected with kitchen.
target5 has been safely removed.
[KinBody.cpp:3873] _ComputeInternalInformation target5 in 0.382521s
target5 has been successfully added.
Running...
Average update rate: 45 Hz
Average update rate: 33 Hz
```

Figura 3.16: La información relativa a la detección de colisiones con el entorno virtual es mostrada por el módulo *openrave driver* durante su ejecución.

3.4.5 Módulo *Trajectory Player*

Para distorsionar las medidas de velocidad introducidas por los usuarios (x , y , z , $roll$, $pitch$, yaw) y provocar una componente no intencional en sus movimientos, se ha incorporado el módulo nombrado como *trajectory player*.

Mediante este módulo se provee a la arquitectura de un generador de ruido gaussiano pregrabado y filtrado a baja frecuencia, $2Hz$, que interactúa con las trayectorias realizadas por los usuarios mediante el puerto YARP de salida, \dot{X}_{noise} . El ruido es añadido a la entrada proporcionada por el usuario, multiplicado por una ganancia que aumenta proporcionalmente con la magnitud de la velocidad cartesiana de traslación comandada.

Este módulo es utilizado como una sencilla aproximación de una discapacidad genérica. Provoca una disminución de las capacidades de los usuarios, puesto que los movimientos que realizan los usuarios con el SpaceNavigator se ven afectados.

3.4.6 Módulo *Window Feedback*

El cometido principal del módulo *window feedback* consiste en la visualización del tiempo de progreso de la simulación. Se representa mediante una ventana que muestra continuamente el tiempo de la simulación, rodeada de un marco cuyo color se modifica en función del estado de la simulación, observar Figura 3.17.

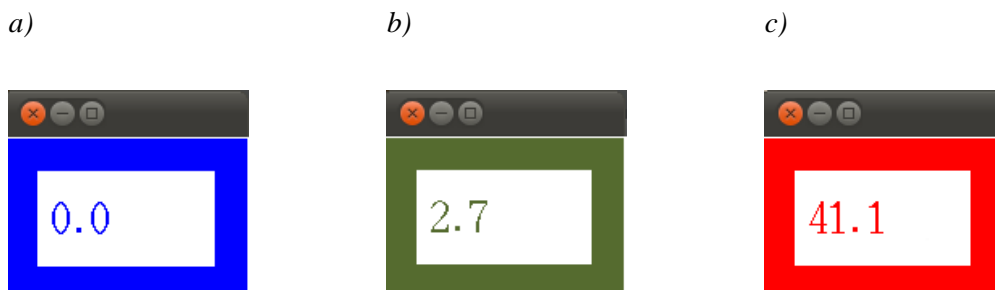


Figura 3.17: a) Ventana con marco de color azul que indica el inicio de la simulación. b) Ventana con marco verde que indica que la simulación está en progreso. c) Ventana con marco de color rojo que indica la existencia de una colisión con el entorno.

Asimismo, proporciona una realimentación visual a los usuarios, cuando se produce una colisión de la mano robótica con el entorno durante el proceso de simulación, cambiando el color del marco de la ventana de verde a rojo, como se observa en la Figura 3.17 a) y la Figura 3.17 b). El tiempo de simulación y el color apropiado del marco de la ventana se suministran a través del puerto YARP, *time/color*, por el módulo *experiment manager*.

La Figura 3.18 muestra el instante inicial antes pulsar el botón izquierdo del SpaceNavigator y comenzar con la simulación. El robot simplificado se encuentra en su posición inicial de partida y la ventana de tiempos muestra el marco en color azul, con el cronómetro a cero.



Figura 3.18: Configuración inicial de partida antes de comenzar con la simulación de la tarea. La ventana de tiempos se rotula en azul.

Este módulo ofrece la posibilidad de seleccionar el formato de la fuente de escritura deseada e introducirlo como parámetro de entrada, como se aprecia en la Figura 3.19.

```
vtejada@WORKSTATION: ~/ASIBOTcoderepo/tags/IROS2011/exp_manager/out/linux-x86
vtejada@WORKSTATION:~/ASIBOTcoderepo/tags/IROS2011/exp_manager/out/linux-x86$ ./main_window_feedback
-sony-fixed-medium-r-normal--24-230-75-75-c-120-jisx0201.1976-0
yarp: Port /window_feedback_i active at tcp://163.117.150.8:10162
Running...
yarp: Receiving input from /exp_manager_visual_o to /window_feedback_i using tcp
```

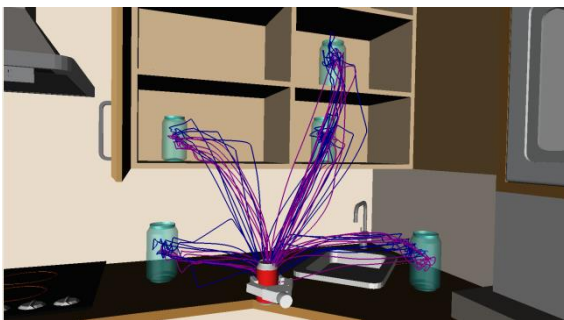
Figura 3.19: Módulo *window feedback* en ejecución, con información sobre la fuente de escritura seleccionada como parámetro de entrada.

3.4.7 Módulo *Trajectory Visualizer*

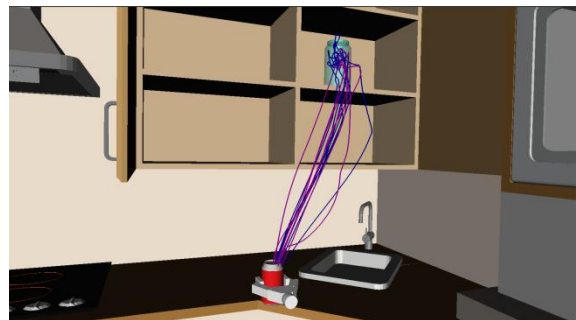
El módulo *trajectory visualizer* permite la visualización de las trayectorias generadas por los usuarios, basadas fundamentalmente en las medidas de posición, orientación, velocidad y tiempo grabadas previamente por el módulo *experiment manager*. También puede proporcionar la visualización de las diferentes trayectorias representativas de cada usuario, a modo de evaluación comparativa del rendimiento para una tarea determinada.

Además, permite la posibilidad de realizar múltiples tipos de representación de trayectorias, seleccionando los usuarios y las tareas que se deseen visualizar, como se puede apreciar en la Figura 3.20.

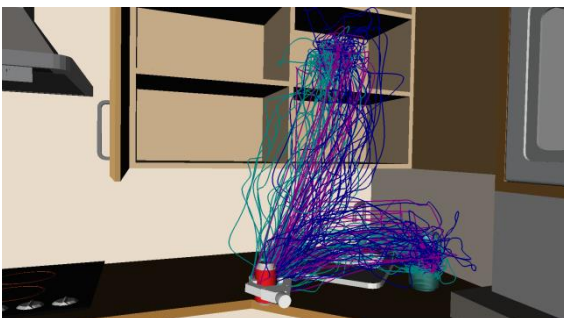
a)



b)



c)



d)

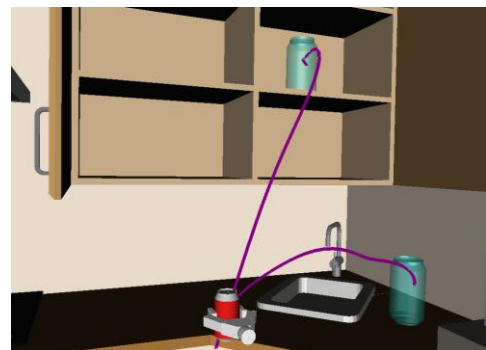


Figura 3.20: a) Representación de todas las trayectorias del experimento para dos de los sujetos (azul y morado). b) Representación de las trayectorias de dos sujetos para una única tarea específica. c) Representación de las trayectorias de tres sujetos (azul, verde y morado) para dos tareas, con inclusión de ruido. d) Representación de las trayectorias representativas de un único sujeto para dos tareas.

Capítulo 4

Estudio Piloto y Resultados

4.1 Introducción

Un sencillo estudio piloto se ha realizado para explorar diferentes modos de evaluar experimentalmente el rendimiento, sobre tareas simplificadas de la vida cotidiana, en un entorno virtual.

Las tareas virtuales simplificadas consisten en mover una lata, desde una posición inicial situada frente al usuario, hasta la posición de un objetivo ubicado en la encimera o el armario de una cocina.

Todos los usuarios, implicados en la realización del estudio piloto, recibieron un formulario de consentimiento donde se detallaban los procedimientos relativos a la participación en el mismo. Dicho formulario, mostrado en el Anexo B del documento, debía ser firmado por cada usuario previamente a su participación en los experimentos.

Este procedimiento es el que se emplea normalmente en estudios experimentales que involucran usuarios, como medida de protección tanto para los propios participantes como para los investigadores que supervisan el estudio piloto.

En este capítulo se describe la metodología experimental que se ha seguido durante la realización del estudio piloto. Asimismo, se ofrece una selección de los resultados experimentales obtenidos mediante la utilización de la plataforma software, descrita en el capítulo anterior.

4.2 Metodología Experimental

El estudio de investigación detallado en este capítulo ha sido realizado por investigadores del *RoboticsLab* en la Universidad Carlos III de Madrid. Los participantes, en el proyecto de investigación, debían tener una edad comprendida entre los 18 y 60 años, ser diestros y no poseer ningún tipo de movilidad reducida en ninguno de los brazos o lesión incapacitante.

El propósito de este estudio ha sido validar un procedimiento experimental para cuantificar el rendimiento, en el control de un robot asistencial, sobre actividades de la vida cotidiana.

4.2.1 Procedimiento Experimental

Se solicitó a los participantes la utilización un dispositivo de entrada al ordenador para mover un objeto virtual en un monitor de ordenador. La duración total de la sesión, para cada participante, fue de alrededor de 50 minutos. El dispositivo de entrada al ordenador se correspondió con un ratón 3D del tipo SpaceNavigator.

Los participantes recibieron un resumen de los procedimientos de trabajo antes de comenzar la sesión de experimentos. La tarea consistió en mover una mano robótica que sostenía una lata de refresco, usando el dispositivo de entrada mencionado anteriormente, en el entorno virtual de la cocina. La mano comenzó siempre en la misma posición para cada ensayo. Un objetivo semitransparente, en forma de una lata grande, se mostró para cada ensayo.

La finalidad del ensayo consistió en colocar la lata, sostenida por la mano robótica, completamente dentro del objetivo semitransparente que, al ser incorporeo, no impedía los movimientos. Un temporizador se mostraba en la parte superior izquierda de la pantalla y cambiaba su color según el estado del ensayo en curso. Antes de iniciar cada ensayo, el temporizador se mostraba en color azul.

Para empezar a mover la mano, se debía presionar el botón del lado izquierdo del dispositivo de entrada, utilizando la mano izquierda. El temporizador cambiaba su color de azul a verde y empezaba a funcionar. A continuación, se debía intentar mover la lata de refresco, que se encontraba sostenida por la mano robótica, tan rápido como fuera posible para introducirla dentro del objetivo. Cuando se lograba alcanzar el objetivo con éxito, el temporizador se detenía automáticamente y se procedía a la siguiente prueba.

Sin embargo, si la mano robótica chocaba con el entorno virtual de la cocina, el temporizador cambiaba su color a rojo por un instante, ver Figura 4.1, y la mano retornaba a su posición inicial. No obstante, el temporizador continuaba funcionando y el usuario debía tratar de terminar la tarea a pesar del tiempo perdido a consecuencia de la colisión.

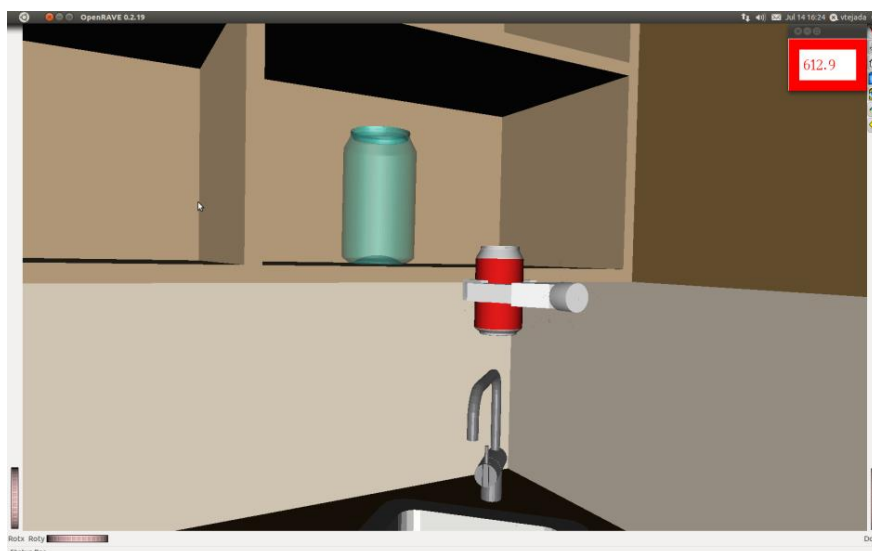


Figura 4.1: El temporizador cambia su color a rojo cuando la mano robótica colisiona con el armario de la cocina.

Previamente al inicio de cada sesión, se instaba a los usuarios a realizar los ensayos lo más rápido posible, teniendo en cuenta que las colisiones podían resultar muy costosas en términos de tiempo.

El participante con el mejor promedio de tiempo durante las sesiones de evaluación recibiría un premio de 10 euros, además de los 10 euros pagados por participar para aumentar su motivación. Ninguna otra información sobre la evaluación individual de cada usuario sería proporcionada.

En primer lugar, se realizaba una sesión de entrenamiento con 27 ensayos para familiarizarse con la configuración del experimento. Si los usuarios disponían de alguna pregunta, podían formularla a los supervisores durante la sesión de entrenamiento o en un descanso.

A continuación, se realizaban 4 sesiones principales, con 27 ensayos cada una, y un breve descanso entre cada una de ellas. Se les recordaba a los participantes que el temporizador no se detenía hasta haber completado con éxito un ensayo, es decir, cuando su marco volviese a ser de color azul. También se permitía a los usuarios hacer una pausa entre los ensayos.

La posición y el tamaño del objetivo cambiaba aleatoriamente entre los ensayos y el rendimiento de la mano robótica variaba entre las sesiones. Durante la sesión inicial de control, el rendimiento de la mano robótica era el mismo que el de la sesión de entrenamiento. Para las 3 sesiones de evaluación restantes, un sencillo sistema de control compartido intentaba ajustar el uso de los 20 sensores de proximidad, colocados en la mano robótica, para ayudar a evitar colisiones cuando ésta se aproximase a los obstáculos.

En definitiva, el cometido principal de todos los usuarios simplemente consistía en procurar conseguir el menor tiempo que les fuese posible en todos los ensayos, teniendo siempre en consideración que las colisiones resultaban muy costosas en términos de tiempo.

4.2.2 Protección de Datos

Se han realizado todos los esfuerzos necesarios para mantener la confidencialidad en la información personal de los usuarios. Para ayudar a proteger dicha confidencialidad, se han tomado las siguientes medidas:

- El nombre de los usuarios no se ha incluido en las encuestas, así como otros datos recogidos.
- Un código identificador se ha utilizado en lugar del nombre.
- Mediante el uso de una clave de identificación, los supervisores son capaces de vincular las encuestas a las identidades de los participantes.
- Sólo los supervisores tienen acceso a dicha clave de identificación.

Si se escribe algún informe o artículo sobre el estudio de investigación, las identidades de los usuarios se protegerán lo máximo posible.

4.2.3 Riesgos del Estudio

Podían existir algunos riesgos durante la participación en el estudio de investigación relacionado con la fatiga de las manos, las muñecas y los brazos, como consecuencia de movimientos repetitivos. Por ello, se efectuaban descansos entre ensayos para aliviar dichos síntomas.

Otro riesgo potencial se corresponde con la identificación de los resultados del estudio por personas ajenas a los supervisores. Esto se evitará en la medida de lo posible y todos los datos acerca del rendimiento experimental obtenido se han mantenido guardados de forma segura en ordenadores protegidos con contraseña. Los datos sólo se pueden vincular a un sujeto determinado a través de una clave de identificación, que se encuentra disponible en exclusiva para los supervisores.

4.2.4 Beneficios del Estudio

Este estudio de investigación no está diseñado para ayudar a los participantes personalmente. No obstante, los resultados obtenidos pueden ayudar a los investigadores a aprender más acerca de cómo el control compartido adaptativo puede resultar útil para los robots asistenciales.

4.2.5 Participación voluntaria

La participación en este estudio de investigación ha sido completamente voluntaria. Los usuarios podían optar por no participar en absoluto. Si decidían participar en el estudio, podían dejar de hacerlo en cualquier momento. En el caso de que decidiesen no participar en el estudio o, si dejaban de participar en cualquier momento, no se penalizaría la confidencialidad de sus datos personales.

La firma del formulario de consentimiento por parte de los usuarios, antes de realizar los experimentos, implicaba:

- Su edad era de, al menos, 18 años.
- Todas las cuestiones planteadas les habían sido respondidas.
- Habían elegido voluntariamente participar en este estudio de investigación.

4.3 Resultados Experimentales

El estudio piloto fue realizado por tres estudiantes varones con edades comprendidas entre los 27 y 31 años, que no poseían ninguna discapacidad. A cada sujeto se le proporcionó el punto de vista de una persona sentada en una silla de ruedas. Los objetivos se situaron en un estante del armario y en la encimera de la cocina. El dispositivo de entrada empleado fue un *joystick* del tipo SpaceNavigator y el experimento se realizó en el entorno virtual OpenRAVE. Los sujetos pudieron controlar

solamente las velocidades de traslación del efector final, utilizado como modelo simplificado de un robot asistencial.

Como se ha explicado en la sección anterior, el experimento consistió en una sesión de control y tres sesiones de evaluación. Para la sesión utilizada como control los usuarios realizaron los ensayos (27 repeticiones para cada tarea en orden aleatorio) sin ruido gaussiano añadido y con el control compartido desactivado. Para las tres sesiones principales de evaluación se añadió ruido gaussiano, incrementándose proporcionalmente con la magnitud de la velocidad de traslación cartesiana comandada. Para estas tres sesiones, el control compartido se modificó. Para la sesión 1, el control compartido fue desactivado. Para la sesión 2 y 3, la constante que relaciona la distancia medida con la velocidad máxima, k , se ajusta a un nivel bajo y alto, respectivamente. Esto implica que la velocidad de los usuarios se limitó más agresivamente en la sesión 3 que en la sesión 2.

La Figura 4.2 muestra las trayectorias ejecutadas por los tres usuarios para todas las sesiones, con diferentes niveles de ruido y valores de control compartido.

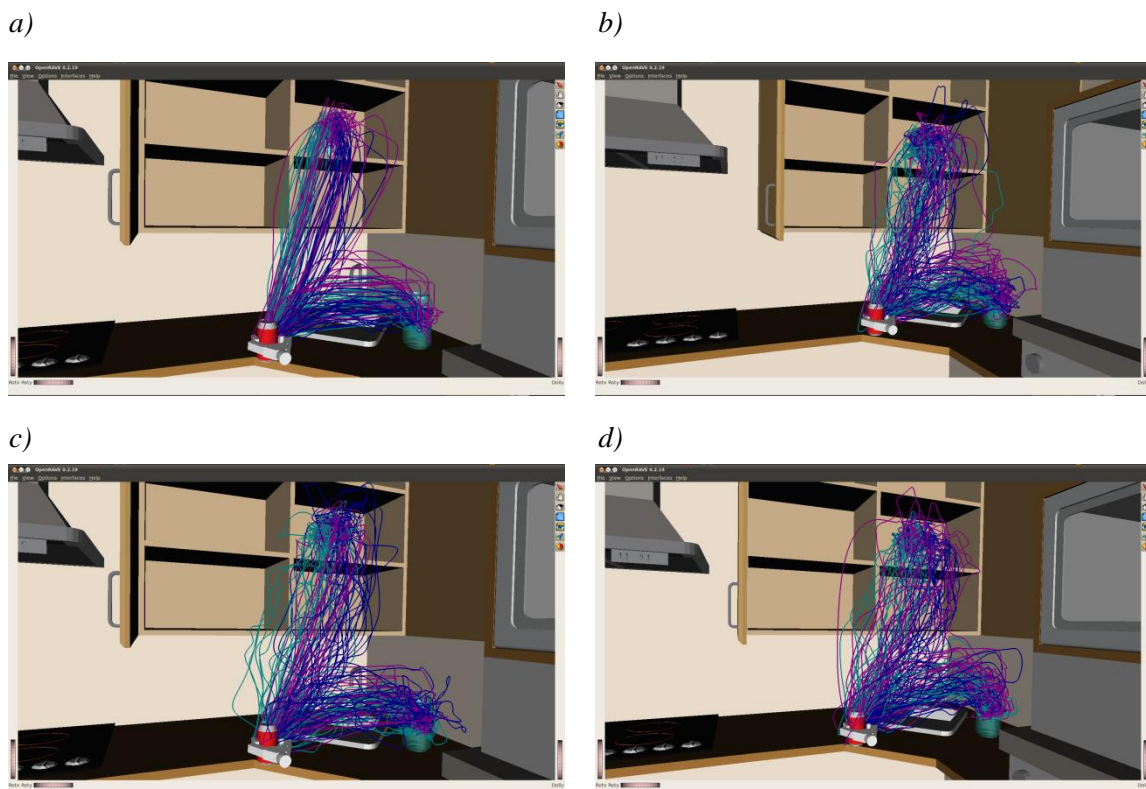


Figura 4.2: a) Sesión de control. b) Sesión 1. c) Sesión 2. d) Sesión 3.

4.3.1 Trayectorias Representativas de los Usuarios

Como se ha podido observar en la Figura 4.2, todos los sujetos realizan las tareas de forma razonablemente coherente para todas las repeticiones, e incluso, adoptan estrategias similares para las respectivas tareas.

El conjunto de trayectorias generadas puede servir para establecer las trayectorias representativas de los usuarios sobre cada tarea específica. Estas trayectorias se pueden utilizar como evaluación comparativa del rendimiento de los usuarios, proporcionando una base para el aprendizaje y la predicción de la intención en cada usuario.

Para explorar este concepto, una regresión basada en mezclas de funciones gaussianas se ha realizado sobre las trayectorias registradas para cada usuario. Para dicho propósito se ha utilizado la *toolbox* de Matlab GMM-GMR¹² v2.0.

La *toolbox* GMM-GMR es un conjunto de funciones de Matlab que permiten entrenar un modelo de mezclas de gaussianas (Gaussian Mixture Model) para obtener datos generalizados, a través de una regresión basada en combinación de gaussianas (Gaussian Mixture Regression). Permite codificar eficientemente cualquier conjunto de datos en el modelo de mezcla de componentes gaussianas (GMM), mediante el uso de algoritmos iterativos de aprendizaje de Expectación-Maximización (EM).

Mediante el uso de este modelo (GMM), la regresión de mezclas de funciones gaussianas (GMR) se puede utilizar para obtener datos parciales de salida especificando las entradas deseadas. En consecuencia, actúa como un proceso de generalización que calcula la probabilidad condicional con respecto a los datos parcialmente observados.

Los códigos fuente de esta *toolbox* son implementaciones de los algoritmos descritos en el libro¹³ denominado "*Robot Programming by Demonstration: A Probabilistic*

¹² <http://www.mathworks.com/matlabcentral/fileexchange/19630-gaussian-mixture-model-gmm-gaussian-mixture-regression-gmr> (Última revisión: 02-07-2013)

¹³ <http://programming-by-demonstration.org/book.php> (Última revisión: 02-07-2013)

Approach" (Calinon, 2009). La Tabla 4.1 resume los fundamentos matemáticos de dichos algoritmos, utilizados en la codificación de las trayectorias de los usuarios.

Un conjunto de datos $\xi = \{\xi_j\}_{j=1}^N$ se define mediante N observaciones $\xi_j \in \mathbb{R}^D$ de los datos sensoriales que cambian a lo largo del tiempo (por ejemplo, las trayectorias realizadas por los usuarios), donde cada punto de datos $\xi_j = \{\xi_t, \xi_s\}$ comprende un valor temporal $\xi_t \in \mathbb{R}$ y un vector espacial $\xi_s \in \mathbb{R}^{(D-1)}$. El conjunto de datos ξ se modela mediante un Modelo Mixto Gaussiano (GMM) de K componentes, definido por la función de densidad de probabilidad:

$$p(\xi_j) = \sum_{k=1}^K \pi_k \mathcal{N}(\xi_j; \mu_k, \Sigma_k),$$

donde π_k son las probabilidades iniciales y $\mathcal{N}(\xi_j; \mu_k, \Sigma_k)$ son las distribuciones gaussianas definidas por los vectores media μ_k y las matrices de covarianza Σ_k , cuyas componentes temporales y espaciales se pueden representar por separado como:

$$\mu_k = \{\mu_{t,k}, \mu_{s,k}\} \quad , \quad \Sigma_k = \begin{pmatrix} \Sigma_{tt,k} & \Sigma_{ts,k} \\ \Sigma_{st,k} & \Sigma_{ss,k} \end{pmatrix}.$$

Para cada componente k , la distribución estimada de ξ_s dado el valor temporal ξ_t se define por:

$$\begin{aligned} p(\xi_s | \xi_t, k) &= \mathcal{N}(\xi_s; \hat{\xi}_{s,k}, \hat{\Sigma}_{ss,k}), \\ \hat{\xi}_{s,k} &= \mu_{s,k} + \Sigma_{st,k} (\Sigma_{tt,k})^{-1} (\xi_t - \mu_{t,k}), \\ \hat{\Sigma}_{ss,k} &= \Sigma_{ss,k} - \Sigma_{st,k} (\Sigma_{tt,k})^{-1} \Sigma_{ts,k}. \end{aligned}$$

Considerando el modelo GMM completo, la distribución esperada se define como:

$$p(\xi_s | \xi_t) = \sum_{k=1}^K \beta_k \mathcal{N}(\xi_s; \hat{\xi}_{s,k}, \hat{\Sigma}_{ss,k}),$$

donde $\beta_k = p(k | \xi_t)$ es la probabilidad de que la componente k sea responsable de ξ_t , por ejemplo:

$$\beta_k = \frac{p(k) p(\xi_t | k)}{\sum_{i=1}^K p(i) p(\xi_t | i)} = \frac{\pi_k \mathcal{N}(\xi_t; \mu_{t,k}, \Sigma_{tt,k})}{\sum_{i=1}^K \pi_k \mathcal{N}(\xi_t; \mu_{t,i}, \Sigma_{tt,i})}.$$

Utilizando las propiedades de la transformación lineal de las distribuciones Gaussianas, una estimación de la esperanza condicional de ξ_s , dada ξ_t se define entonces por $p(\xi_s | \xi_t) \sim \mathcal{N}(\hat{\xi}_s, \hat{\Sigma}_{ss})$, donde los parámetros de la distribución Gaussiana son definidos mediante:

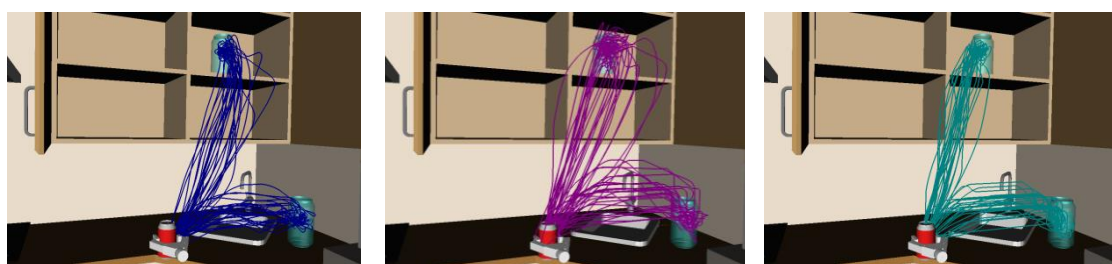
$$\hat{\xi}_s = \sum_{k=1}^K \beta_k \hat{\xi}_{s,k} \quad , \quad \hat{\Sigma}_{ss} = \sum_{k=1}^K \beta_k^2 \hat{\Sigma}_{ss,k}$$

Evaluando $\{\hat{\xi}_s, \hat{\Sigma}_{ss}\}$ en cada instante de tiempo ξ_t , se calcula la forma generalizada de los movimientos $\hat{\xi} = \{\xi_t, \hat{\xi}_s\}$ y las matrices de covarianza asociadas $\hat{\Sigma}_{ss}$, las cuales describen las restricciones.

Tabla 4.1: Codificación probabilística del conjunto de datos de las trayectorias empleando la Regresión Mixta Gaussiana (GMR).

La Figura 4.3 a) muestra la representación gráfica del conjunto de trayectorias generadas por los tres usuarios, durante la sesión de control, para las dos tareas especificadas mediante la utilización del módulo *trajectory visualizer*. Este conjunto de trayectorias se ha codificado en un modelo mixto gaussiano (GMM), utilizando cuatro componentes, y la regresión mixta gaussiana (GMR) se ha utilizado para obtener la posición esperada en cada instante. De este modo, se consigue una versión suavizada y generalizada de las trayectorias ejecutadas por los tres usuarios para cada una de las tareas acometidas, tal y como se puede observar en la Figura 4.3 b).

a)



b)

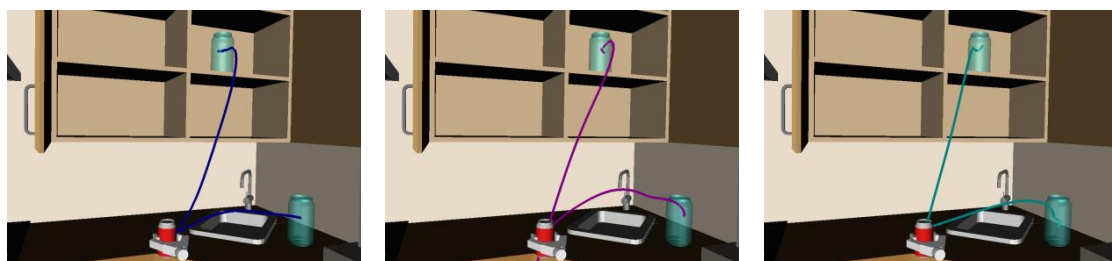


Figura 4.3: a) Trayectorias realizadas por los 3 sujetos durante la sesión de control, con 27 repeticiones sobre cada tarea. b) Regresión mixta gaussiana de las trayectorias generadas por los 3 sujetos para las correspondientes tareas.

Las diferencias existentes entre los movimientos intencionados realizados durante la sesión de control y la distorsión provocada por la adición de ruido en la sesión 1, se ponen de manifiesto en la Figura 4.4 a) y Figura 4.4 b), respectivamente. Estos movimientos son relativos a un participante pero constituyen una muestra representativa de los tres participantes para la sesión de control, sin ruido gaussiano o ajustado a nivel bajo de ruido, y la primera sesión del estudio piloto, con ruido gaussiano añadido o ajustado a nivel alto de ruido.

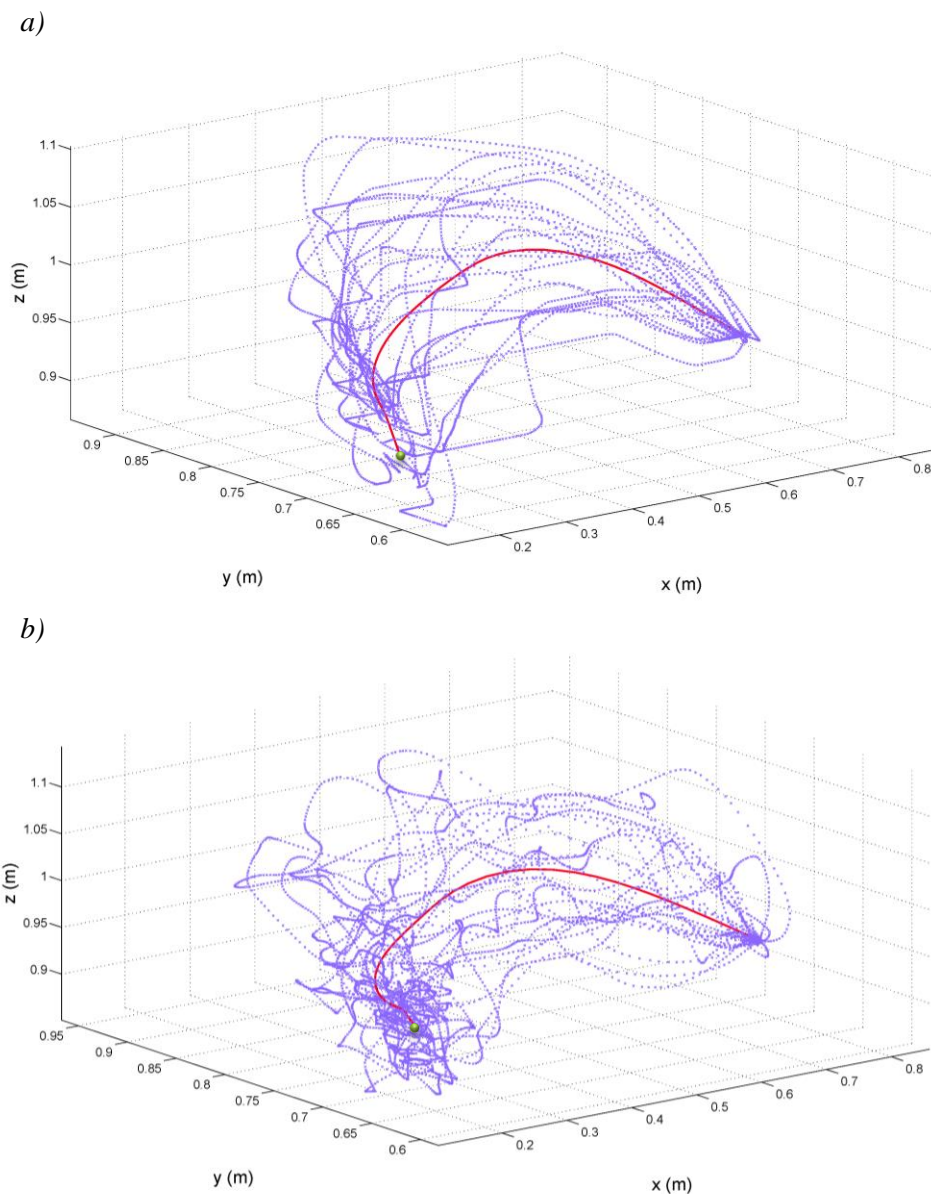


Figura 4.4: *a)* Trayectorias generadas por un usuario (27 repeticiones) y GMR obtenida para la sesión de control, sin ruido gaussiano (nivel bajo). *b)* Trayectorias generadas por un usuario (27 repeticiones) y GMR obtenida para la sesión 1, con ruido gaussiano ajustado a nivel alto. El objetivo se corresponde con la tarea situada en la encimera y se muestra mediante un punto verde.

A continuación, se realiza un breve análisis de los resultados obtenidos para los tres sujetos en dos sesiones distintas. Para la sesión 1, el control compartido está desactivado y el nivel de ruido es alto, mientras que para la sesión 3 tanto el ruido gaussiano como el grado de ajuste del control compartido se sitúan en su valor máximo. Todos los resultados se han obtenido usando 4 funciones gaussianas.

4.3.1.1 Usuario con clave de identificación id 184

La Figura 4.5 muestra la regresión mixta gaussiana (GMR) y modelo mixto gaussiano (GMM) de las trayectorias generadas por el usuario con id 184 para la sesiones 1 y 3.

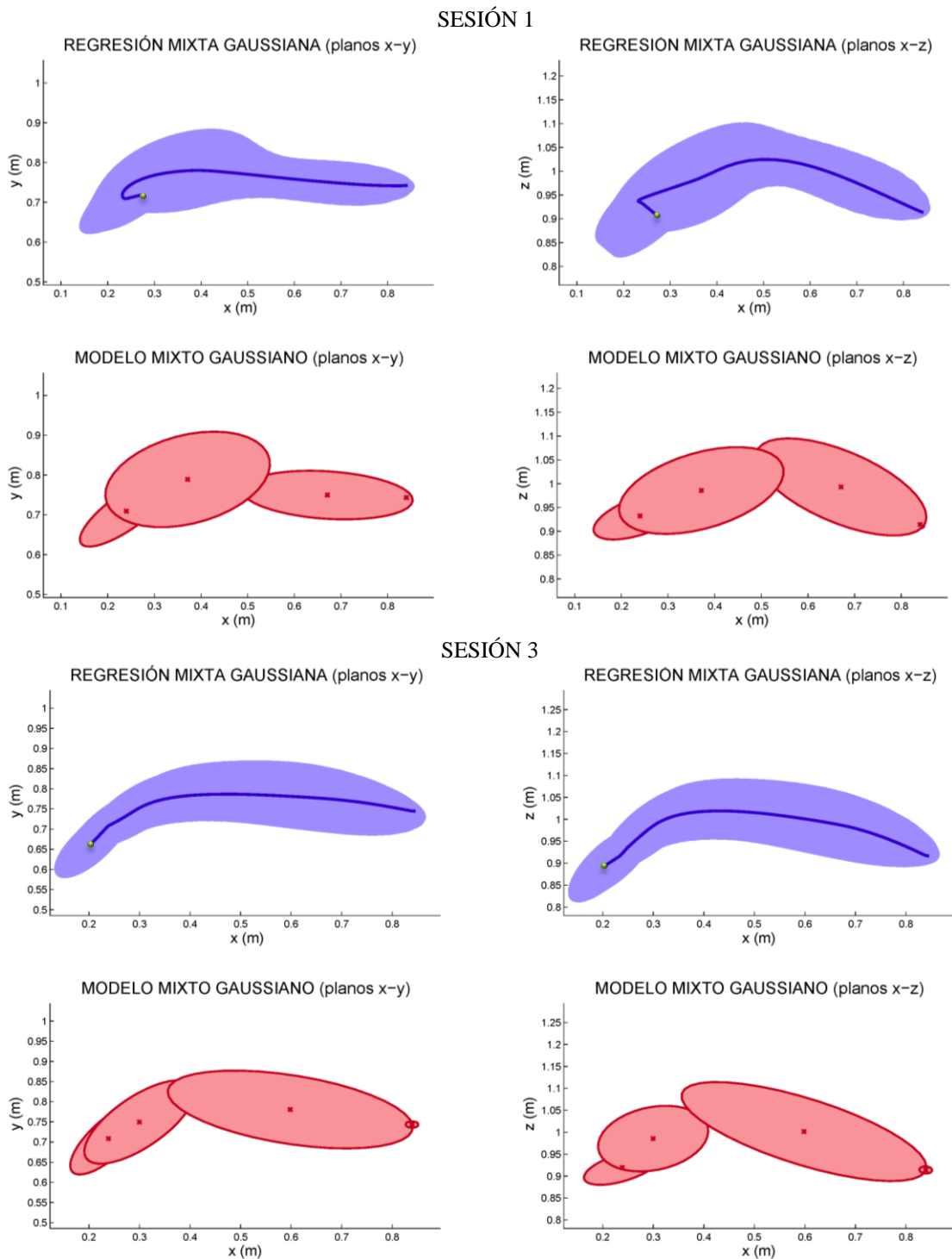


Figura 4.5: GMR y GMM de las trayectorias de la sesión 1 y 3 (usuario id 184).

En la figura anterior se observan las trayectorias generalizadas, en color azul, obtenidas a partir de las 27 repeticiones realizadas por el usuario con el identificador 184. La desviación estándar, asociada a estas trayectorias representativas, se muestra difuminada en color azul y simboliza la dispersión de los datos en la ejecución de la totalidad de las trayectorias.

Las representaciones gráficas de dichas trayectorias se corresponden con la vista obtenida en el plano $x-y$ (vista en planta) y en el plano $x-z$ (vista de perfil). Debajo de las trayectorias se sitúan las representaciones gráficas de los modelos de mezcla de funciones gaussianas, en color rojo, para el planos $x-y$ y el plano $x-z$. Estas funciones gaussianas se distribuyen de acuerdo con la codificación del conjunto de datos obtenido para cada una de las repeticiones.

Asimismo, la tarea seleccionada se corresponde con el emplazamiento del objetivo sobre la encimera de la cocina. La ubicación de esta tarea se representa en las gráficas por un punto verde situado en el extremo de las trayectorias representativas.

Para la sesión 1, con nivel de ruido alto y sin control compartido, se observa una dispersión baja en el plano $x-y$ durante el inicio de la trayectoria. Únicamente se aprecia un considerable aumento de la desviación estándar cuando se realiza la aproximación a la posición de la tarea. Dicho aumento también se manifiesta, de forma evidente, en el plano $x-z$ durante la segunda mitad del recorrido.

El incremento en la dispersión de los datos se debe a la introducción de la componente de ruido gaussiano, que genera un aumento de los movimientos no intencionados. Esta adición de ruido provoca una mayor dificultad cuando se pretende posicionar la lata en el interior del objetivo.

Durante la sesión 3, el nivel de ruido y el control se ajustan a su máximo valor. En este caso se debería observar, con respecto a la sesión 1, una disminución de la dispersión de los datos en torno al objetivo final. De esta manera, el efecto que genera la implementación del control compartido puede apreciarse con claridad en las gráficas correspondientes al plano $x-y$ y al plano $x-z$. A pesar de la dificultad que entraña la precisión en los movimientos, al aproximarse a la ubicación de la tarea (punto verde), se observa una reducción en la desviación estándar.

4.3.1.2 Usuario con clave de identificación id 399

La Figura 4.6 muestra la regresión mixta gaussiana (GMR) y modelo mixto gaussiano (GMM) de las trayectorias generadas por el usuario con id 399 para la sesiones 1 y 3.

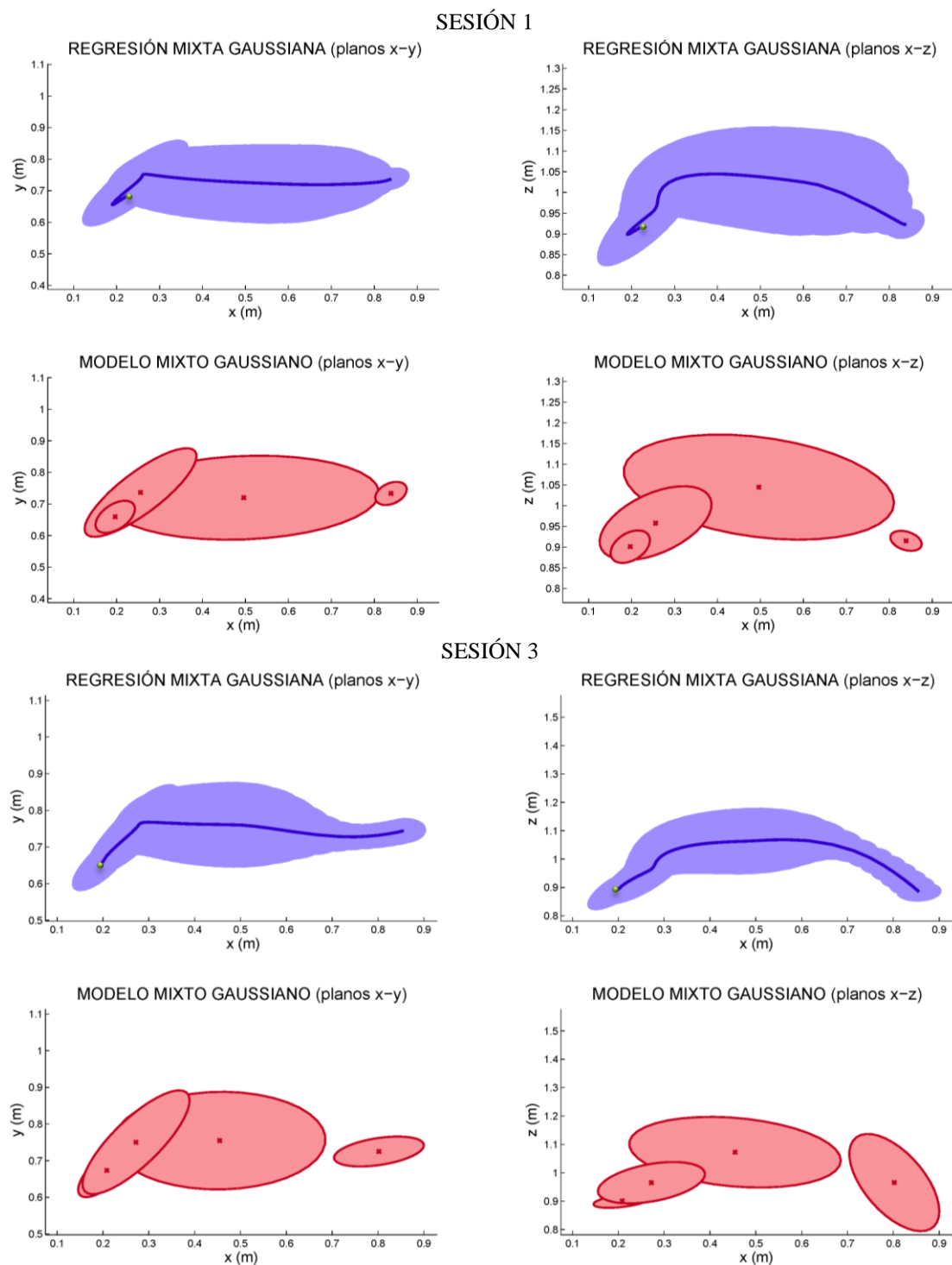


Figura 4.6: GMR y GMM de las trayectorias de la sesión 1 y 3 (usuario id 399).

La estrategia desarrollada por el usuario con identificador 399 se muestra en la figura anterior. De igual modo que en el apartado anterior, las gráficas representan la codificación de los datos obtenidos de las trayectorias del sujeto mediante la regresión mixta gaussiana (GMR) y el modelo mixto gaussiano (GMM). Asimismo, la tarea seleccionada se corresponde con la situada sobre la encimera de la cocina, denotada por un punto verde en las gráficas de las trayectorias representativas.

Como se puede observar en las gráficas de las trayectorias representativas (GMR), este usuario acomete la tarea de forma similar al sujeto anterior, realizando una curva más pronunciada para aproximarse al objetivo. La dispersión del conjunto de datos, representada por la desviación estándar, también resulta más acusada, especialmente, para el plano $x-z$. Esto implica una mayor variabilidad de los datos en el eje z (altura) durante la ejecución de las 27 repeticiones para la tarea propuesta.

Los modelos gaussianos mixtos (GMM) que se representan en el plano $x-y$ y en el plano $x-z$, codifican los datos de las trayectorias efectuadas para la sesión 1 y la sesión 3. Se puede apreciar que la generalización de la trayectoria, a través de la regresión mixta gaussiana (GMR), discurre cercana a los puntos centrales de las funciones gaussianas.

Para las representaciones gráficas en ambos planos, la adición de una componente de ruido gaussiano provoca una desviación estándar mayor en la parte central de la trayectoria representativa (GMR) durante las dos sesiones. En cambio, en la sesión 3 la desviación estándar se reduce considerablemente en la parte inicial de la trayectoria representativa.

De nuevo, para la sesión 3, la selección del control compartido a su nivel más alto tendría que provocar una disminución de la distorsión de las trayectorias representativas, particularmente, en las cercanías de objetivo. Esto se debe a que el ajuste del control compartido a su nivel máximo provoca una limitación de la velocidad cartesiana.

En consecuencia, en dicha sesión se puede observar una ligera disminución de la desviación estándar de los datos, tanto en el plano $x-y$ como en el plano $x-z$, tras efectuar el giro de aproximación para abordar el objetivo desde el punto de partida.

4.3.1.3 Usuario con clave de identificación id 624

La Figura 4.7 muestra la regresión mixta gaussiana (GMR) y modelo mixto gaussiano (GMM) de las trayectorias generadas por el usuario con id 624 para la sesiones 1 y 3.

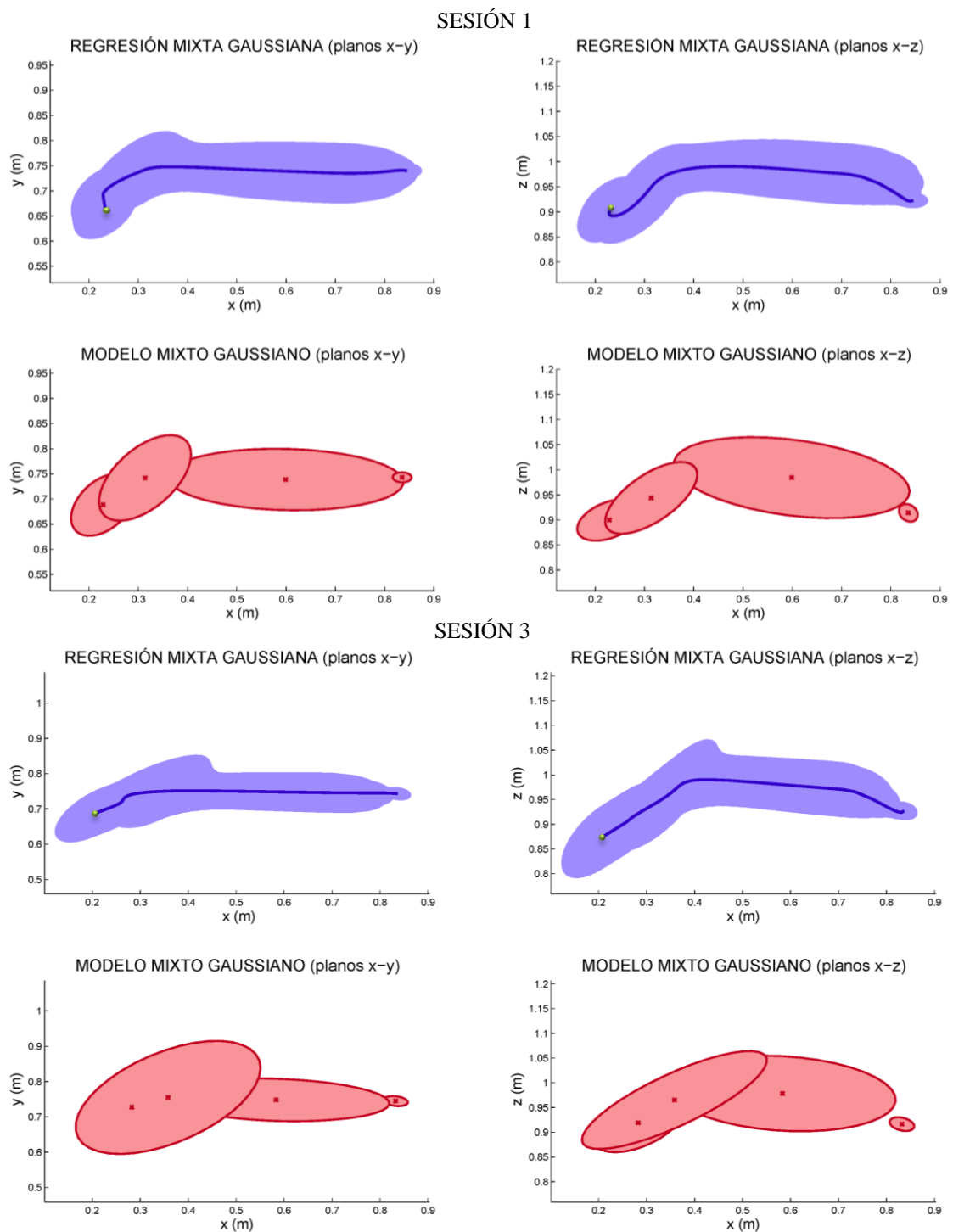


Figura 4.7: GMR y GMM de las trayectorias de la sesión 1 y 3 (usuario id 624).

La figura anterior muestra la estrategia adoptada por el último usuario, con código identificador 624. Las representaciones gráficas de las trayectorias características y la tarea seleccionada se corresponden con las analizadas anteriormente para los otros dos sujetos.

Mediante la codificación del conjunto de datos en el modelo de mezcla de funciones gaussianas (GMM) y la regresión mixta gaussiana (GMR), se obtiene la versión suavizada y generalizada del total de las trayectorias ejecutadas para la sesión 1 y 3. Se puede apreciar que esta generalización de las trayectorias discurre próxima al punto central de las funciones gaussianas, del mismo modo que ocurre para los dos sujetos anteriores.

Las trayectorias representativas, obtenidas de las 27 repeticiones realizadas, revelan las similitudes en la ejecución de la tarea con respecto a los otros usuarios. En consonancia con las habilidades desarrolladas por el resto de los usuarios, se puede observar que el sujeto acomete la tarea de manera prácticamente rectilínea hacia la posición del objetivo.

Este hecho posibilita una reducción sustancial en la dispersión de los datos que se generaba cuando se realizaba trayectoria más curvada, fundamentalmente en el eje z (altura). De hecho, se aprecia que de la desviación estándar es bastante uniforme a lo largo de toda la trayectoria, tanto en el plano x - y como en el plano x - z .

En la sesión 3, cuando el control compartido se encuentra activado, éste debería constituir una ayuda para lograr alcanzar el objetivo. Sin embargo, este efecto no se puede observar, en este sujeto, en ninguno de los dos planos representados. La desviación estándar, en relación con la sesión 1, no se reduce durante la aproximación al objetivo sino que se mantiene prácticamente constante.

Los resultados que se obtienen del análisis de las representaciones gráficas para este sujeto presentan un carácter ambiguo puesto que no se percibe una disminución de la velocidad, cuando se ajusta el control compartido a su máximo valor durante la sesión 3. Este hecho implica que, para ambas sesiones, la media de tiempos (MT) de este sujeto va a permanecer prácticamente constante y, en consecuencia, el control compartido no supone una ayuda para la realización de la tarea.

4.3.2 Evaluación de las Estrategias de Control Compartido

Los parámetros que se han aplicado al conjunto de datos, para evaluar el rendimiento durante el experimento, han sido la media de los tiempos (MT) y el número de errores cometidos. Siguiendo el procedimiento típico que se utiliza en los experimentos de la ley de Fitts, sólo la media de los tiempos para el primer intento con éxito se ha empleado. Los resultados obtenidos para los tres participantes en el experimento se pueden observar en la Figura 4.8 a).

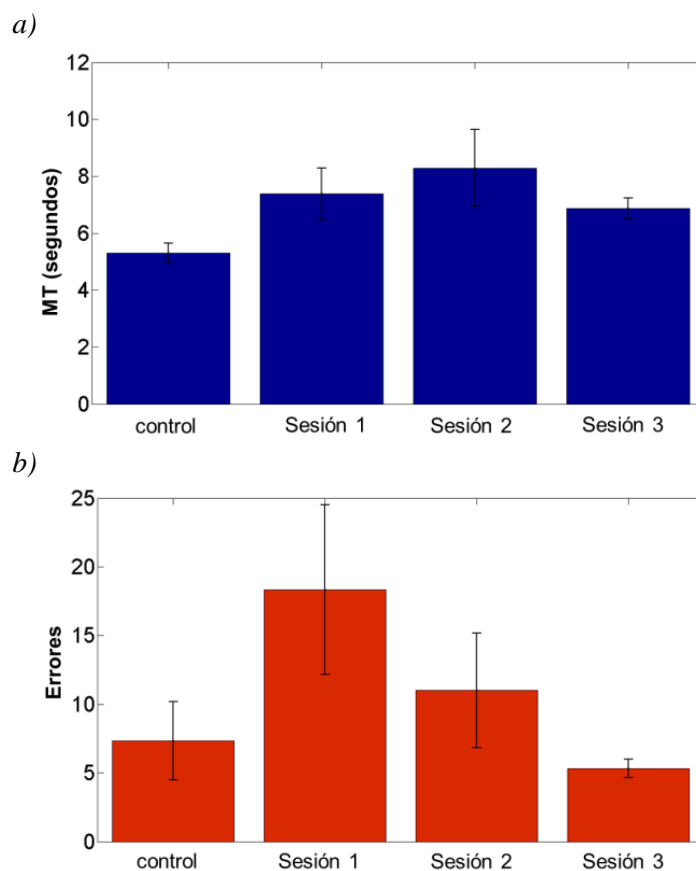


Figura 4.8: a) Media de los tiempos (MT) para el primer intento realizado con éxito. b) Número total de errores. Para cada sesión se muestra la media para los tres sujetos, mientras que las barras de error indican la desviación estándar.

Sin embargo para este tipo de experimentos se requiere una tasa de error muy baja, típicamente alrededor del 2-5%. Para el estudio piloto realizado, se hizo especial énfasis en esta instrucción, e incluso, se añadió una motivación especial mediante una gratificación económica al usuario con el mejor promedio de tiempo en todos los ensayos. No obstante, como se puede apreciar en la Figura 4.8 b), la tasa de error se

mantuvo relativamente alta, desde el 10% en la sesión 3 hasta el 34% de la sesión 1, a pesar de que las colisiones resultaban muy costosas en términos de tiempo.

La alta tasa de error provoca que los resultados de la media de tiempos (MT), mostrados en la Figura 4.8 a), sean un tanto ambiguos. Mientras que se observa un incremento en la media de los tiempos (MT) con respecto a la sesión de control, las tasas de error son también mucho más altas. Este hecho indica que los sujetos han sacrificado los errores a favor del promedio de sus tiempos (MT), con ruido añadido.

Debido a estos resultados, las alternativas fueron calcular la media de los tiempos (MT) para el primer intento de cada ensayo, se produzca o no una colisión, o calcular la media de los tiempos (MT) sobre todos los ensayos sin tener en cuenta los errores cometidos. Los resultados obtenidos para los tres sujetos pueden apreciarse en la Figura 4.9 a) y la Figura 4.9 b), respectivamente.

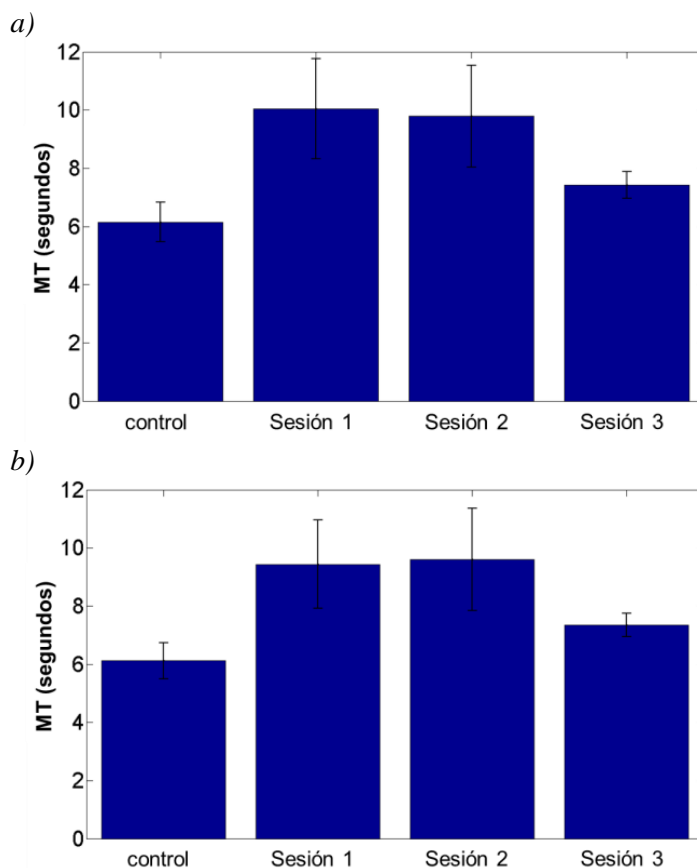


Figura 4.9: a) Media de los tiempos (MT) para el primer intento realizado, con o sin éxito. b) Media de los tiempos (MT) para todas las repeticiones. Para cada sesión se muestra la media para los tres sujetos, mientras que las barras de error indican la desviación estándar.

De esta manera, el efecto de los errores queda incluido, en cierta medida, en el tiempo medio (MT) obtenido para cada participante, puesto que las colisiones resultan muy costosas en términos de tiempo y, en consecuencia, el rendimiento sobre todos los ensayos disminuye.

En estas gráficas se puede observar una distinción más clara entre la sesión de control y las sesiones que incorporan ruido gaussiano, así como una mejora en el rendimiento cuando el control compartido se ajusta a un nivel alto en la sesión 3.

Capítulo 5

Conclusiones y Trabajos Futuros

En este capítulo se pretende realizar una descripción de las conclusiones más destacadas, obtenidas tanto de las herramientas software y hardware empleadas como de la plataforma software de experimentación desarrolladas en este Proyecto Fin de Carrera.

Por otro lado, también se describirán un conjunto de trabajos futuros necesarios para conseguir desarrollar una plataforma experimental más realista y adecuarla, de forma efectiva, a las características propias de los usuarios.

5.1 Conclusiones

La Interacción Humano-Robot (HRI), en el campo de la robótica asistencial, constituye un reto. En parte, debido a las propias necesidades de los usuarios reales que deben ser capaces de utilizar los robots en un entorno parcialmente no estructurado y complejo, como su propio hogar.

La heterogeneidad del grupo potencial de usuarios es otro desafío, puesto que requiere flexibilidad y capacidad de adaptación por parte del robot asistencial. A medida que se trabaja hacia la satisfacción de estos requisitos, existe una necesidad de metodologías robustas para la evaluación de experimentos en el campo de la interacción hombre-robot (HRI). De ese modo, se puede ayudar a impulsar el proceso de desarrollo y aumentar la reproducibilidad y la utilidad de experimentos aplicados a la robótica asistencial.

Las conclusiones más relevantes que se pueden extraer de este proyecto son las que se exponen a continuación.

5.1.1 Metodologías Experimentales Estudiadas

Los experimentos controlados pueden ayudar a proporcionar resultados reproducibles dentro del campo de la robótica asistencial y utilizarlos como base para extraer conclusiones sobre los trabajos realizados. Sin embargo, existen obstáculos para desarrollar las metodologías de evaluación existentes sobre el terreno.

Por ejemplo, teorías como la ley de Fitts, aunque resultan útiles para los dispositivos de entrada al ordenador, no son necesariamente aplicables para los movimientos realizados en espacios de alta dimensión, como los involucrados en el control de un robot asistencial.

Estas metodologías, por tanto, no son apropiadas cuando se pretende evaluar el rendimiento experimental de un sistema complejo que implique al robot asistencial y al usuario final.

5.1.2 Especificaciones de Diseño de la Plataforma Virtual

Para evaluar el sistema en una plataforma que se aproxime al entorno real y a las actividades de la vida diaria, se ha empleado el modelo virtual del entorno de pruebas de la cocina para robots asistenciales, que dispone la Universidad Carlos III de Madrid.

No obstante, el posicionamiento de una lata de refresco en el interior de un objetivo semitransparente, atravesándolo, resultó confuso e irreal para los participantes. Asimismo, es significativo destacar que la ejecución de la tarea, en un entorno de simulación 2D, provocó carencias en la percepción de la profundidad para los participantes. Este hecho imposibilitó la precisión de sus movimientos en la colocación de la lata dentro del objetivo seleccionado.

El control del robot asistencial, por parte del usuario final, se ha facilitado mediante el manejo de una mano robótica simplificada. De esta manera, los usuarios no requieren estar pendientes de los movimientos de las articulaciones del robot y de las posibles colisiones que se puedan producir con el entorno.

Aunque el estudio piloto realizado puede incluir usuarios reales, es difícil de encontrar un grupo homogéneo de usuarios con discapacidades, suficientemente grande, que pueda proporcionar datos significativos para un análisis estadístico. Por ello, se ha optado por emplear sujetos que no presentaban ningún tipo de discapacidad asociada.

5.1.3 Plataforma Virtual de Experimentación

La arquitectura software del entorno virtual de simulación presenta un carácter distribuido. De esta manera, todos los módulos pueden ser implementados y depurados de forma individual, reduciendo y acotando los posibles errores que se puedan cometer durante su desarrollo.

Para conseguir dicho propósito se utilizaron las librerías YARP. Las librerías YARP proporcionan el *middleware* necesario para gestionar una comunicación rápida y eficiente entre los módulos que integran la arquitectura, permitiendo mantenerlos desacoplados entre sí. También posibilita que dichos módulos puedan ser ejecutados en diferentes ordenadores con sistemas operativos distintos.

La implementación del módulo que controla el entorno virtual de simulación ha sido realizada mediante OpenRAVE. Esta herramienta de simulación y planificación para aplicaciones robóticas, admite la integración personalizada de funcionalidades que se cargan en tiempo de ejecución. Su arquitectura ha permitido la configuración de las

tareas de la vida diaria seleccionadas, el control de la mano robótica simplificada, la gestión de las colisiones con el entorno virtual, así como la implementación de un sistema de sensores de proximidad distribuidos por toda la mano robótica virtual.

De igual modo, OpenRAVE se ha utilizado para el desarrollo del módulo software que permite la visualización de las trayectorias generadas por los usuarios, para las distintas tareas planteadas. Este módulo presenta una gran versatilidad puesto que posibilita la representación gráfica de múltiples tipos de trayectorias, en el entorno virtual, seleccionando el número de usuarios y tareas que se deseen visualizar.

Un módulo software que genera ruido gaussiano, filtrado a baja frecuencia, ha sido implementado e integrado en la plataforma de experimentación. Mediante este módulo se distorsionan las trayectorias realizadas por los usuarios, provocando una componente no intencional en sus movimientos. En consecuencia, dicho módulo se emplea como modelo de una discapacidad genérica.

El módulo *experiment manager*, que gestiona la ejecución de los experimentos, permite la configuración de las tareas seleccionadas en el entorno virtual de experimentación sin necesidad de recompilar la aplicación. La secuencia de las tareas y su orden de ejecución se especifican individualmente para cada usuario en función del código de identificación que se les ha asignado previamente.

5.1.4 Metodología Experimental Desarrollada

Los procedimientos experimentales empleados para cuantificar el rendimiento en el control de un robot asistencial simplificado, sobre actividades de la vida cotidiana, han sido cuidadosamente definidos en un formulario de consentimiento.

Este formulario de consentimiento describe la metodología experimental que debían seguir los usuarios durante la realización del estudio piloto. De ese modo, los participantes fueron informados de los procedimientos de trabajo antes de comenzar la sesión de experimentos, la confidencialidad de sus datos, los riesgos y beneficios del estudio, así como la participación voluntaria en el estudio piloto.

Todos los participantes firmaron el formulario de consentimiento, las cuestiones planteadas les fueron respondidas satisfactoriamente y el estudio piloto se desarrolló sin ningún tipo de incidencias.

5.1.5 Evaluación y Validación de Resultados

La evaluación de la plataforma software de experimentación, mediante la realización de un estudio piloto, ha ofrecido resultados prometedores.

La plataforma software desarrollada muestra un funcionamiento correcto que responde a los requisitos impuestos. Las pruebas realizadas durante la implementación y la verificación de su funcionamiento confirman la fiabilidad y robusta integración de todos los módulos que componen la arquitectura, posibilitando la ejecución en tiempo real de las distintas tareas programadas.

Los resultados preliminares obtenidos, durante la realización de los experimentos, indican una mejora en el rendimiento de los usuarios durante las sesiones que incorporan el control compartido. No obstante, el análisis estadístico de los datos también muestra la necesidad de un ajuste más óptimo de los parámetros relativos al control compartido.

En consecuencia, se ha desarrollado una arquitectura software que facilita la evaluación experimental de metodologías de trabajo aplicadas al campo de la robótica asistencial. Esta plataforma permite la simulación de tareas específicamente diseñadas en un entorno virtual de experimentación y posibilita la validación experimental de los resultados obtenidos mediante la realización de estudios pilotos.

5.2 Trabajos Futuros

La amplia variedad de aplicaciones que se pueden desarrollar mediante la plataforma virtual de experimentación posibilitan la viabilidad de numerosas líneas de

investigación. Una selección de las líneas de trabajo futuras, que actualmente se encuentran en progreso, se enumeran a continuación.

5.2.1 Manipulador Robótico Asistencial

La elaboración de experimentos controlados más realistas implica inexorablemente la utilización del modelo completo del robot asistencial ASIBOT.

Para que el manipulador robótico asistencial ASIBOT pueda ser teleoperado por los usuarios, se requiere la integración de dos módulos específicos en la arquitectura software desarrollada. Por consiguiente, resultaría necesaria la implementación conjunta de un módulo dedicado a resolver la cinemática inversa diferencial del robot y un módulo que permita controlar el movimiento de sus articulaciones.

Además, para extender las funcionalidades propias del robot, sería conveniente la distribución de un conjunto de sensores infrarrojos de proximidad, combinados con sensores táctiles para la detección de colisiones, por todos los eslabones del modelo virtual del robot ASIBOT.

5.2.2 Interacción con el Entorno Virtual de Experimentación

El módulo que gestiona los experimentos, *experiment manager*, debería permitir a los desarrolladores de la arquitectura seleccionar la configuración de las variables que caracterizan los experimentos mediante archivos externos de configuración. De este modo, no sería necesario recompilar el módulo cada vez que se modificasen las especificaciones de los experimentos. Se podría seleccionar, por ejemplo, el número y la ubicación de las tareas en el entorno virtual así como definir la penalización más adecuada en caso de colisión, para dotar de más realismo a la simulación sin perder por ello el control sobre el experimento.

La mayor limitación que presenta el entorno virtual de experimentación es la falta de percepción de la profundidad. Puesto que OpenRAVE contempla la posibilidad de operar en modo estéreo *Quad buffer*, este problema puede ser resuelto mediante la

instalación de un sistema de visión 3D. La implementación de este sistema permitiría a los usuarios adquirir cierta percepción sobre la profundidad en el entorno virtual. Igualmente, la colocación de una cámara en el extremo del robot facilitaría la visión para la localización y manipulación del objetivo seleccionado.

La incorporación en el diseño de la simulación de un dispositivo háptico modelo PHANTOM Omni^{®14}, mostrado en la Figura 5.1, proporcionaría a los usuarios una realimentación de fuerza en el entorno virtual.



Figura 5.1: Dispositivo háptico modelo PHANTOM Omni[®].

Mediante este tipo de realimentación, se podría percibir la activación del control compartido o la detección de una colisión con el entorno virtual, mediante una vibración energética, repercutiendo en una mejora en el rendimiento de la tarea sin implicar que el manipulador robótico deba retornar a su posición inicial de partida durante el ensayo.

Además, la utilización de un sistema de audio, en forma de pulsos de tonos, podría ayudar a los usuarios a evitar posibles colisiones del robot con el entorno virtual. La frecuencia de los tonos se podría utilizar para identificar un determinado eslabón y la frecuencia de los pulsos sería proporcional a la relación de máxima proximidad, determinada por el conjunto total de sensores en cada momento.

¹⁴ <http://www.sensable.com/haptic-phantom-omni.htm> (Última revisión: 02-07-2013)

5.2.3 Control Compartido Adaptativo y Planificación Supervisada

Los manipuladores robóticos asistenciales, como ASIBOT, tienen como finalidad permitir a los usuarios con discapacidades físicas realizar actividades cotidianas, de manera segura y eficaz, en entornos dinámicos.

Para cumplir este propósito, se debería implementar un control compartido adaptativo, donde los sensores de proximidad y colisión se utilizaran, en tiempo de ejecución, para limitar las colisiones con el entorno durante el proceso de teleoperación.

Asimismo, futuros desarrollos en estrategias de planificación automática, bajo la supervisión del usuario, podrían permitir que el usuario y el robot se adaptasen en tiempo real, con el objetivo de converger en un uso adecuado de los sensores implementados en el robot y mejorar el rendimiento sobre las tareas de la vida cotidiana.

5.2.4 Usuarios Finales

Cuando se evalúa experimentalmente un sistema robótico completo, la inclusión de usuarios reales es obligatoria, aunque la realización de experimentos controlados en sistemas tan complejos pueda constituir un desafío.

Por otro lado, el número total de usuarios para futuros estudios pilotos debería establecerse en un mínimo de doce participantes, cifra que los médicos y terapeutas consideran suficiente para extraer conclusiones significativas.

Es de esperar que un mayor énfasis en las metodologías experimentales pueda ayudar a trasladar estos sistemas robóticos fuera de los laboratorios de investigación hasta las casas de los usuarios reales, que hoy en día todavía están aguardándolo.

5.3 Publicaciones Relacionadas

Los trabajos publicados, relativos al Proyecto Fin de Carrera que se ha realizado, se enumeran a continuación:

1. M.F. Stoelen, A. Jardón, V. Fernández, J. G. Victores, S. Martínez, F. Bonsignorio and C. Balaguer. (2011, May). Methodologies for Experimental Evaluation of Assistive Robotics HRI. *Robocity2030 9th Workshop. Robots colaborativos e interacción humano-robot*. ISBN: 978-84-614-5558.UPM-CSIC, 111-127.
2. A. Jardón, M. F. Stoelen, V. Fernández, J. G. Victores, S. Martínez de la Casa, C. Balaguer & F. Bonsignorio. (2011). Aplicación de teoría de la información para el modelado y cuantificación de la interacción persona-robot. *DRT4ALL. IV Congreso Internacional de Diseño, Redes de Investigación y Tecnologías para todos* (págs. 36-38). ISBN: 978-84-88934-50-5.
3. M. F. Stoelen, A. Jardón, V. Fernández, C. Balaguer & F. Bonsignorio. (2011, Mar). An information-theoretic approach to modeling and quantifying assistive robotics HRI. *Late Breaking Report, Proceedings of the 6th international conference on Human-robot interaction (HRI)*. Lausanne. Switzerland.

Además, la siguiente publicación parte de la base del trabajo desarrollado en este Proyecto Fin de Carrera.

1. M. F. Stoelen, V. F. Tejada, A. Jardón, F. Bonsignorio & C. Balaguer. (2012, Oct) Benchmarking Shared Control for Assistive Manipulators: From Controllability to the Speed-Accuracy Trade-Off. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura. Portugal.

6.- Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	32.333
Amortización	15
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	6.470
Total	38.817

Tabla 6.1: Presupuesto global del Proyecto.

Referencias

A. Jardón, M. F. Stoelen, V. Fernández, J. G. Victores, S. Martínez de la Casa, C. Balaguer & F. Bonsignorio. (2011). Aplicación de teoría de la información para el modelado y cuantificación de la interacción persona-robot. *DRT4ALL. IV Congreso Internacional de Diseño, Redes de Investigación y Tecnologías para todos* (págs. 36-38). ISBN:978-84-88934-50-5.

Accot, J. & Zhai, S. (1997). Beyond Fitts' law: models for trajectory-based HCI tasks. *Proceedings of the SIGCHI Conference on Human, 97*, pp. 295-302.

Amigoni, F., Reggiani, M. and Schiaffonati, V. (2009). An insightful comparison between experiments in mobile robotics and in science. *Autonomous Robots, 27*(4), 313-325.

Balaguer, C., Gimenez, A., Jardon, A., Cabas, R. & Correal, R. (2005). Live experimentation of the service robot applications for elderly people care in home environments. *In Proceedings of the International Conference on Intelligent Robots and Systems*, (pp. 2345-2350).

Balaguer, C., Gimenez, A., Jardon, A., Correal, R., Martinez, S., Sabatini, A. M. (2007). Proprio and Teleoperation of a Robotic System for Disabled. *4*, 415-427.

Balaguer, C., Gimenez, A., Jardon, A., Sabatini, A., Topping, M. & Bolmsjo, G. (2006). The mats robot: service climbing robot for personal assistance. *Robotics Automation Magazine, IEEE, 13*(1), 51-58.

Bien, Z., Chung, M.J., Chang, P.H., Kwon, D.S., Kim, D.J., Han, J.S., Kim, J.H, Kim, D.H., Park, H.S., Kang, S.H., Lee, K, & Lim, S.C. (2004). Integration of a rehabilitation robotic system (kares ii) with human friendly man-machine interaction units. *Autonomous Robots, 16*(2), 165-191.

Calinon, S. (2009). *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press.

Chan, R. & Childress, D. (1990a). On a unifying noise-velocity relationship and information transmission in human-machine systems. *IEEE Transactions on System, Man and Cybernetics*, 20(5), 1125-1135.

Diankov, R. (2010, Aug). *Automated Construction of Robotics Manipulation Programs*. PhD thesis, Robotics Institute: Carnegie Mellon University.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6), 381-391.

Fitzpatrick, P., Metta, G. and Natale, L. (2007). Towards Long-Lived Robot Genes. *Robotics and Autonomous Systems*, 56(1), 29-45.

K. Tsui, H. Yanco, D. J. Feil-Seifer, and M. J. Mataric. (Aug 2008). Survey of domain-specific performance measures in assistive robotic technology. *in Proceedings of the Performance Metrics for Intelligent*, (págs. 116–123). Washington, D.C.

K.M. Tsui, and Yanco H.A. (2009). Towards establishing clinical credibility for rehabilitation and assistive robots through experimental design. *Workshop on Good experimental methodology in robotics, Robotics Science and Systems*. Seattle, WA, USA.

M.F. Stoelen, A. Jardón, V. Fernández, J. G. Vítores, S. Martínez, F. Bonsignorio and C. Balaguer. (2011, May). Methodologies for Experimental Evaluation of Assistive Robotics HRI. *Robocity2030 9th Workshop. Robots colaborativos e interacción humano-robot*. ISBN: 978-84-614-5558. UPM-CSIC, 111-127.

Mackenzie, I. S. (1989). A note on the information-theoretic basis for Fitts' law. *J. Mot. Behav.*, 21, pp. 323-330.

Mataric, M.J., Eriksson, J., Feil-Seifer, D.J. & Winstein, C.J. (2007). Socially assistive robotics for post-stroke rehabilitation. *Journal of Neuroengineering and Rehabilitation*, 4(5).

Ramos, T. (Septiembre 2009). Simulación de la plataforma robótica Hoap-3 en el simulador OpenHRP3. Leganés, Madrid: UC3M.

Shook, L. & Akin, D. (2001). Evaluation of Various Hand Controllers for Use by a Space Suited Subject. *In 31st International Conference On Environmental System, Vol. 10*. Orlando, FL, USA.

Stoelen, M. F. (Septiembre 2010). Physical and cognitive user-adaptation of an assistive robot. Leganés, Madrid: UC3M.

Topping, M. (1993). Early experiences encountered in the placement of 60 handy 1 robotic aids to eating. *In Proceedings of the IEEE 93 Conference on Cybernetics*, (pp. 543-546).

Tsui, K., Yanco, H., Kontak, D. & Beliveau, L. (2008). Development and evaluation of a flexible interface for a wheelchair mounted robotic arm. *Proceedings of the 3rd international conference on human robot interaction - HRI'08*, (pp. 105-112).

Victores, J. G. (Septiembre 2010). Software engineering techniques applied to assistive robotics: guidelines & tools. Leganés, Madrid: UC3M.

Ware, J. (1972). An input adaptive, pursuit tracking model of the human operator. *In Seventh Annual Conference on Manual Control, Vol. 281*, p. 33.

Anexo A

Manual de Puesta en Funcionamiento

La secuencia de instrucciones para ejecutar los módulos que integran la arquitectura software y la puesta en funcionamiento del entorno virtual de experimentación, mediante las conexiones de puertos YARP, se enumeran a continuación:

[1]. Ejecutar el servidor YARP:

```
$> yarp server
```

[2]. Para inicializar el driver del SpaceNavigator:

```
$> ASIBOTcoderepo/branches/hmi/hmi_spacnavigator/extern/bin/linux-x86/  
$> sudo ./spacnavd
```

Nota: Cuando se ejecuta debe encenderse la luz azul del SpaceNavigator.

[3]. Para ejecutar el módulo del SpaceNavigator:

```
$> ASIBOTcoderepo/branches/hmi/hmi_spacnavigator/out/linux-x86/  
$> ./spacnavigator_snav
```

[4]. Para ejecutar el módulo del administrador de experimentos se necesita introducir la clave id del sujeto y el número de la sesión:

```
$> ASIBOTcoderepo/tags/IROS2011/exp_manager/out/linux-x86/  
$> ./main_exp_manager_steering --id_subject --session
```

Nota: Para el parámetro `--id_subject` se utilizan las claves id asignadas a los sujetos: 184, 399 y 624. Para el parámetro `--session` se utilizan los números de las sesiones que conforman el experimento:

- Sesión de entrenamiento, sin ruido gaussiano (`--session: 0`).
- Sesión de control, sin ruido gaussiano (`--session: 1`).
- Sesión 1, con ruido gaussiano (`--session: 2`).
- Sesión 2, con ruido gaussiano (`--session: 3`).
- Sesión 3, con ruido gaussiano (`--session: 4`).

[5]. Para ejecutar el módulo del entorno de simulación (OpenRAVE):

```
$> ASIBOTcoderepo/tags/IROS2011/exp_openrave_driver/out/linux-x86/  
$> ./exp_openrave_driver
```

[6]. Para ejecutar el módulo que almacena las trayectorias realizadas durante la simulación:

```
$> ASIBOTcoderepo/tags/IROS2011/exp_manager/out/linux-x86/
$> ./main_trajectory_recorder
```

Nota: los archivos con los datos obtenidos se almacenan en el mismo directorio. El formato de los nombres de los ficheros es el siguiente:

- 'su': id_subject (clave de identificación del usuario).
- 'se': session (número de la sesión).
- 'e': executed (máximo de 27 ejecuciones para el experimento).
- 't': trial (número del ensayo).
- 'a': attempt (número de intentos para un determinado ensayo).

[7]. Para ejecutar el módulo del control compartido se necesita ajustar el tiempo mínimo para que se produzca una colisión (k) y dos parámetros asociados a la depuración del módulo:

```
$> ASIBOTcoderepo/tags/IROS2011/uci_proximity/out/linux-x86
$> ./main_uci_proximity --k --pdebug1--pdebug2
```

Nota: Los parámetros de depuración, --pdebug1 y --pdebug2, se ajustan a 0 durante el desarrollo del experimento. El parámetro --k toma los siguientes valores para las respectivas sesiones:

- Sesión de entrenamiento, sin control compartido (--k: 0).
- Sesión de control, sin control compartido (--k: 0).
- Sesión 1, sin control compartido (--k: 0).
- Sesión 2, con control compartido ajustado a nivel bajo (--k: 0.8).
- Sesión 3, con control compartido ajustado a nivel alto (--k: 1.6).

[8]. Para ejecutar el módulo que distorsiona las velocidades que comandan los usuarios, introduciendo ruido gaussiano:

```
$> ASIBOTcoderepo/tags/IROS2011/exp_manager/out/linux-x86/
$> ./main_trajectory_player
```

[9]. Para ejecutar el módulo que muestra la ventana con el color y tiempo de la simulación se necesita introducir la fuente de escritura deseada:

```
$> ASIBOTcoderepo/tags/IROS2011/exp_manager/out/linux-x86/
$> ./main_window_feedback --font
```

Nota: El tipo de letra se introduce mediante el parámetro --font. Para el experimento se ha utilizado: -sony-fixed-medium-r-normal--24-230-75-75-c-120-jisx0201.1976-0.

[10]. Para ejecutar el script de conexión de todos los puertos YARP que comunican entre si los módulos de la arquitectura software:

```
$> ASIBOTcoderepo/tags/IROS2011/share/  
$> ./connect_IROS2011_experiment.sh
```

Nota: Verificar que todos los puertos YARP se han conectado con éxito.

[11]. En el termina donde se ejecuta el módulo que graba las trayectorias durante la simulación, se introduce 'y' para confirmar que se desea grabar las trayectorias de los usuarios.

[12]. En el terminal donde se ejecuta el módulo del administrador de experimentos, se introduce 'y' para inicializar la arquitectura software del experimento.

Nota: Cada sesión consta de 27 ejecuciones con 2 diferentes posiciones del objetivo, denominadas trial 2 y trial 3, en tandas de 2 repeticiones. El orden de las posiciones del objetivo se distribuye de forma aleatoria para cada clave id del usuario. En este módulo se muestra información del número del ensayo y la ejecución en la que se encuentra cada usuario, si se ha tenido un éxito o fracaso en el intento, la clave id del usuario, la sesión del experimento y el ruido gaussiano asignado a dicha sesión.

Realizado todos los pasos de forma satisfactoria, por parte de los supervisores, el experimento comienza cuando el usuario presiona el botón izquierdo del dispositivo de entrada, SpaceNavigator.

La Tabla A.1 muestra el script de conexión de todos los puertos YARP que comunican los módulos de la plataforma de experimentación.

```
#!/bin/bash  
  
yarp connect /spacenavigator_rate_o /exp_manager_rateX_i  
yarp connect /spacenavigator_button_o /exp_manager_buttons_i  
yarp connect /exp_driver_collisions_o /exp_manager_collisions_i  
yarp connect /exp_driver_pose_o /exp_manager_X_i  
yarp connect /exp_driver_pose_o /uci_trans_i  
yarp connect /exp_manager_setup_o /exp_driver_setup_i  
yarp connect /uci_robo_o /exp_driver_rate_i  
yarp connect /trajectory_player_o /exp_manager_noise_i  
yarp connect /exp_driver_sensors_o /uci_prox_i  
yarp connect /exp_manager_rateX_o /uci_user_i  
yarp connect /uci_robo_o /exp_manager_externall_i  
yarp connect /exp_manager_visual_o /window_feedback_i
```

Tabla A.1: Script para ejecutar las conexiones de los puertos YARP.

En la Tabla A.2 se muestran las conexiones de los puertos YARP, identificando los puertos de entrada y de salida, así como la etiqueta que los asocia al diagrama de bloques representado en la Figura 3.7 (apartado 3.4).

PUERTOS YARP		ETIQUETAS
SALIDA	ENTRADA	
/spacnavigator_rate_o	/exp_manager_rateX_i	\dot{X}_{IN} (RATES)
/spacnavigator_button_o	/exp_manager_buttons_i	BUTTONS
/exp_driver_collisions_o	/exp_manager_collisions_i	COLLISIONS
/exp_driver_pose_o	/exp_manager_X_i	X (POSE)
/exp_driver_pose_o	/uci_trans_i	
/exp_manager_setup_o	/exp_driver_setup_i	SETUP
/uci_robo_o	/exp_driver_rate_i	$\dot{X}_{COMMAND}$ (RATES)
/uci_robo_o	/exp_manager_external1_i	
/exp_driver_sensors_o	/uci_prox_i	SENSORS
/exp_manager_rateX_o	/uci_user_i	$\dot{X}_{MANAGER}$ (RATES)
/trajectory_player_o	/exp_manager_noise_i	\dot{X}_{NOISE} (RATES)
/exp_manager_visual_o	/window_feedback_i	TIME/COLOR

Tabla A.2: Asociación entre las conexiones de los puertos YARP de entrada y salida y la etiqueta que los identifica con el flujo de datos.

Anexo B

Formulario de Consentimiento

CONSENT FORM

Project Title	<i>Pilot Study on Adaptive Shared Control for a Simplified Assistive Robot</i>
Why is this research being done?	<p><i>This is a research study being conducted by Martin F. Stoelen, Alberto Jardón Huete and Fabio Bonsignorio of the RoboticsLab at Universidad Carlos III de Madrid. We are inviting you to participate in this research project because you are between 18 and 60 years old, are right-handed, and have no reduced mobility of either arm (for example a cast or an inhibiting injury).</i></p> <p><i>The purpose of this research is to assess an experimental procedure for quantifying performance in controlling an assistive robot on typical household tasks (daily life activities).</i></p>
What will I be asked to do?	<p><i>You will be asked to perform tasks using a computer input device to move a virtual object on a computer workstation display. The total duration of participation is about 50 minutes. The input device used is a joystick of type SpaceNavigator.</i></p> <p><i>You will be given a brief of the task procedures before beginning. The task involves moving a robotic hand holding a soda-can using the input device mentioned earlier in a kitchen environment. The hand will start in the same position for each trial. A transparent target in the form of a larger can will be shown for each trial.</i></p> <p><i>Your goal is to place the can the hand is holding completely inside the transparent target (the target is imaginary, it does not impede your movements). A timer will be displayed in the top left part of the display. It's color will change depending on the state of the current trial. Before starting a trial the timer will be blue. To start moving the hand you must press the left hand side button on the input device (using the left hand). The timer will change color to green and start running. You should then attempt to move the can being held by the hand as fast as you can to within the target and stop. When this has been achieved the timer will stop automatically and you will proceed to the next trial.</i></p> <p><i>However, if the hand itself (not the soda-can) collides with the environment, the timer will change to red for a moment, and the hand will go back to its initial position. The timer will keep running and you should try again to finish the task, even though you will have lost the time spent until the collision. Please try to perform the trials as fast as possible, taking into account that collisions can be very costly in terms of time.</i></p> <p><i>The subject with the best average time over the 6 non-training sessions will receive EURO 10 in addition to the EURO 5 paid for participation. No</i></p>

	<p><i>other information about individual performance will be provided.</i></p> <p><i>First you will be given 2 training session with 25 trials each to get familiar with the experiment setup. If you have any questions during the training session or in a break, please feel free to ask the experimenter. You will then perform 6 main sessions, each of 25 trials, and with a short break in-between. Remember that the timer does not run after you have successfully completed a trial (when the timer is blue), and you are allowed to take breaks in-between trials.</i></p> <p><i>The position and size of the target will vary randomly between trials. The performance of the hand will vary between sessions. The first 2 main sessions the hand performance will be the same as the 2 training sessions. For the remaining 4 a simple shared control system will attempt to adjust the use of 20 proximity sensors placed on the hand, to help you avoid collisions when close to obstacles. For all trials you should simply attempt to achieve the lowest times possible, while keeping in mind that collisions are costly in terms of time.</i></p>
What about confidentiality?	<p><i>We will do our best to keep your personal information confidential. To help protect your confidentiality:</i></p> <ol style="list-style-type: none"> <i>1. Your name will not be included on the surveys and other collected data.</i> <i>2. A code will be placed on the survey and other collected data.</i> <i>3. Through the use of an identification key, the researcher will be able to link your survey to your identity.</i> <i>4. Only the researcher will have access to the identification key.</i> <p><i>If we write a report or article about this research project, your identity will be protected to the maximum extent possible.</i></p>
What are the risks of this research?	<p><i>There may be some risks from participating in this research study related to fatigue of hands, wrists and arms as a result of repeated motions. You will be given breaks between trials to alleviate this. Another potential risk is the identification of your results by persons other than the experimenters. This will be prevented to the maximum extent possible and all performance data will be kept secure on password protected computers. The data can only be linked to a given subject through a identification key, which will be available exclusively to the experimenters.</i></p>
What are the benefits of this research?	<p><i>This research is not designed to help you personally, but the results may help the investigator learn more about whether the adaptive shared control used can be useful for assistive robots.</i></p>
Do I have to be in this research? May I stop	<p><i>Your participation in this research is completely voluntary. You may choose not to take part at all. If you decide to participate in this research, you may stop participating at any time. If you decide not to participate in this study or if you stop participating at any time, you will not be penalized or lose any</i></p>

participating at any time?	<i>benefits to which you otherwise qualify.</i>	
What if I have questions?	<p><i>This research is being conducted by Martin F. Stoelen, Alberto Jardón Huete and Fabio Bonsignorio of the RoboticsLab at UC3M. If you have any questions about the research study itself, please contact:</i></p> <p style="text-align: center;">Martin F. Stoelen Dpto. Ingeniería de Sistemas y Automática Universidad Carlos III de Madrid c/Butarque 15. 28911 Leganés (Madrid) (e-mail) mstoelen@ing.uc3m.es (telephone) +34 916249970</p> <p style="text-align: center;">Alberto Jardón Huete Dpto. Ingeniería de Sistemas y Automática Universidad Carlos III de Madrid c/Butarque 15. 28911 Leganés (Madrid) (e-mail) ajardon@ing.uc3m.es (telephone) +34 916246242</p> <p style="text-align: center;">Fabio Bonsignorio Dpto. Ingeniería de Sistemas y Automática Universidad Carlos III de Madrid c/Butarque 15. 28911 Leganés (Madrid) (e-mail) fbonsign@ing.uc3m.es</p>	
Statement of Age of Subject and Consent	<p><i>Your signature indicates that:</i></p> <p><i>you are at least 18 years of age;</i></p> <p><i>the research has been explained to you;</i></p> <p><i>your questions have been fully answered; and</i></p> <p><i>you freely and voluntarily choose to participate in this research project.</i></p>	
Signature and Date	NAME OF SUBJECT	
	SIGNATURE OF SUBJECT	
	DATE	