UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SISTEMA DE TELEOPERACIÓN BASADO EN INTERFÁZ HÁPTICA PARA BRAZO ROBÓTICO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

DAVID RODRIGO VALENZUELA URRUTIA

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
MARTIN ADAMS
OMAR DAUD ALBASINI

SANTIAGO DE CHILE
MAYO 2016

RESUMEN DE TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA MENCIÓN ELÉCTRICA Y
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO.
POR: DAVID RODRIGO VALENZUELA URRUTIA
FECHA: MAYO 2016
PROFESOR GUÍA: JAVIER RUIZ DEL SOLAR SAN MARTÍN

Sistema de Teleoperación Basado en Interfaz Háptica para Brazo Robótico

Esta tesis se enfoca en la interfaz hombre máquina (*Human-Machine Interface*) de un sistema de Teleoperación Bilaterial, con el objetivo de mejorar la experiencia del operador, incrementando la eficiencia y calidad del trabajo realizado por el robot controlado por el usuario. Se propone un sistema de teleoperación basado en la tecnología háptica, la cual entrega al usuario información del entorno de trabajo del robot a través del sentido del tacto además de la información tradicional entregada a través del sentido de la vista. Con un dispositivo de control háptico el usuario es capaz de percibir los relieves y formas de los objetos que rodean al robot. Para concretar el desarrollo de la teleoperación háptica, se implementa un método de cálculo de fuerza de retroalimentación utilizando la *Point Cloud* obtenida desde el entorno de operación del robot. En la teleoperación tradicional, el usuario sólo es asistido por las imágenes de las cámaras ubicadas en el sitio de operación. Con la metodología propuesta, el operador puede tomar mejores decisiones ya que además de la información visual, posee la información táctil.

La hipótesis de este trabajo es que la teleoperación háptica permite al operador controlar un brazo robótico con mayor precisión, comparando su desempeño con una teleoperación no háptica. La retroalimentación háptica en el dispositivo de control permite evitar movimientos indeseados con el robot, y por lo tanto previene colisiones. En el sistema de teleoperación propuesto, el brazo robótico tiene sujeto una cámara RGB-D cerca del efector, con lo cual se obtienen imágenes 3D del ambiente desde diferentes ángulos. El movimiento del efector del robot es controlado por el usuario a través del dispositivo háptico *Phantom Omni*, el cual está sujeto a las restricciones del algoritmo de *proxy*. Usando el algoritmo háptico, la información de posición del *proxy* y el movimiento requerido por el operador, se realiza el desplazamiento del efector del robot. El efector se mueve libremente cuando no hay obstáculos, pero al detectar una posible colisión emite una señal que activa en el dispositivo *Phantom Omni* una fuerza de retroalimentación para el usuario, quien percibe la repulsión del objeto presente en el ambiente real del robot. La validación experimental de la metodología de teleoperación propuesta, consiste en la realización de pruebas de seguimiento de trayectoria por 20 operadores humanos voluntarios, utilizando el efector del robot industrial *KUKA* y un panel de pruebas. Los operadores deben completar las pruebas de seguimiento mediante el uso de retroalimentación háptica y también en la ausencia de esta. Además, se les pide completar estas tareas usando dos modos de cinemática del robot. Para cada usuario voluntario se obtuvo el error de posición del efector del robot en cada sección de la trayectoria, se contabilizó la cantidad de correcciones de posición, cantidad de colisiones (fuertes y débiles), y además se obtuvo el tiempo requerido para completar la tarea.

Los resultados obtenidos a partir del análisis estadístico de la validación experimental, confirman que con el método de teleoperación háptico (comparado con la teleoperación no háptica) se aumenta la precisión del operador en tareas de seguimiento de trayectoria. Este aumento en la precisión se refleja en la reducción del error promedio entre el desplazamiento ideal y el desplazamiento realizado por los usuarios, además de reducir el número de colisiones totales entre el efector del robot y el panel de pruebas. De esta manera, se confirma la hipótesis planteada, demostrando que al agregar el sentido háptico a sistemas de control bilaterales aumenta la precisión de teleoperación y se reducen los tiempos de entrenamiento que requieren los operadores para mover el robot real con éxito.

ABSTRACT OF THE THESIS TO OBTAIN THE GRADE OF
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
AND ELECTRICAL ENGINEER.
BY: DAVID RODRIGO VALENZUELA URRUTIA
DATE: MAY 2016
THESIS ADVISOR: JAVIER RUIZ DEL SOLAR SAN MARTÍN

Teleoperation System Based on Haptic Interface for Robotic Arm

This thesis is focused on the Human-Machine Interface of a Bilateral Teleoperation System, the objective of which is to improve the operator experience via increasing the efficiency and quality of the teleoperated robot tasks. The teleoperation system proposed in this work uses haptic technology that provides the user with environmental information from the robot workspace via the senses of touch and sight, using the haptic device and a traditional screen. With the haptic control device, the user senses the reliefs and forms of the objects that surround the robot. To make this possible, it is necessary calculate the force feedback from the Point Cloud data of the robot workspace. Using traditional teleoperation systems, the user is only assisted by images from cameras located in the operation zone. On the other hand, with the proposed approach the operator can make better decisions with more information (haptic and visual) from the robot environment.

The hypothesis of this work is that the haptic teleoperation gives the user a more precise control of the robotic arm compared to traditional non-haptic teleoperation. The haptic feedback in the control device avoids undesirable movements with the robotic arm and, therefore, prevents collisions. In the proposed teleoperation system, the robotic arm has a RGB-D camera near its end-effector, enabling 3D images to be taken of the environment using different angles. The end-effector movement is controlled by the user through the haptic device Phantom Omni and the proxy algorithm. Using the force feedback algorithm, the proxy position, and the required movement from the operator, the robot end-effector can be moved. The end-effector moves freely when no obstacles are detected, but when a possible collision is close, a signal is activated in the Phantom Omni, and the force is fed back to the user, who perceives the object repulsion from the real robot workspace. The experimental validation of the proposed methodology consists of follow path tasks in a test panel with 20 volunteers, using the end-effector of an industrial robot KUKA. The operators must complete the follow path tasks with and without the force feedback. Furthermore, the users have to complete the tasks with two modes of robot cinematic control. For each user, the following are obtained: the effector error position in every step of the trajectory, the total position corrections, the number of collisions (strong and weak), and finally the total time in completing the tasks.

The results obtained from the statistical analysis of experimental validation confirm that using the method of haptic teleoperation instead of the non-haptic method increases the operator precision in a follow path task. This precision increment is observed in the mean error reduction between the ideal trajectory and the trajectory generated by the users, and decrease of the total number of collisions between the robot end-effector and the test panel. The hypothesis is confirmed, proving that incorporating the touch sense to a bilateral teleoperation system increases the precision and decreases the training time required by the users for moving the robot in the real environment.

A mi madre Gloria Ivonne y
a mi padre Rodrigo Antonio

Agradecimientos

En primer lugar, agradezco a mi madre Gloria Ivonne y mi padre Rodrigo Antonio, por su apoyo incondicional y toda la fuerza que me entregaron durante estos veinticinco años, siete de los cuales viví en la Universidad. Ellos son el pilar fundamental en mi vida, y entregaron la energía que necesité para superar los numerosos desafíos de la vida.

Agradezco a Dios por la oportunidad de estudiar en esta Universidad y la voluntad que me entrega día a día para superar las metas con mucha valentía, dedicación y esfuerzo.

Agradezco a cada una de las personas que me acompañaron en este camino mediante discusiones, ideas, propuestas y apoyo. De manera muy especial agradezco a Rodrigo Muñoz por sus consejos, su asistencia técnica y profesional en todas las etapas del proyecto. A Sebastian Morris, y a Aquiles Martínez por su importante colaboración en las etapas iniciales de este desafío. Y a todos aquellos que participaron voluntariamente en la validación experimental del proyecto.

A mi profesor guía Javier Ruiz del Solar, quien siempre me entregó apoyo y me alentó a seguir adelante con el proyecto, aportando con ideas y planteando desafíos nuevos.

A los miembros de la comisión Martin Adams y Omar Daud, quienes revisaron la presente tesis y realizaron los comentarios necesarios para aclarar y pulir los últimos detalles del trabajo.

Finalmente, agradezco a CONICYT por el financiamiento entregado a través de la Beca Nacional de Magister.

Tabla de contenido

Índice de tablas

Índice de figuras

1 Introducción

1.1. Motivación

1.1.1. Fundamentación general

La teleoperación de sistemas electromecánicos es ampliamente usada en la industria, siendo uno de los nichos más importantes la minería, y también en operaciones donde se requiera manipulación de sustancias peligrosas. La teleoperación de un robot es necesaria cuando su autonomía es restrictiva, debido a los diversos escenarios de peligros e imprevistos a los cuales se puede enfrentar, o bien, cuando las labores que debe desempeñar son muy variadas y en condiciones de entornos muy disímiles. Es en estos casos, donde el juicio humano prevalece por sobre algún algoritmo de trabajo que controle los movimientos del robot. Por otra parte, la teleoperación se vuelve necesaria para reemplazar a los operarios de maquinaria pesada que trabajan dentro de una faena, enfrentándose a condiciones adversas y peligrosas para su salud. Estos operarios pueden ser trasladados a oficinas o sitios de operación a kilómetros de distancia del lugar de faena, en un ambiente seguro y libre de contaminantes.

El desafío que plantea la teleoperación (de cualquier tipo de sistema electromecánico) es mejorar la interacción hombre-máquina, para poder en primer lugar, obtener sensorialmente para el usuario una experiencia que aporte la mayor cantidad de información posible del entorno en el cual se desempeña el efector final, y por lo tanto, en segundo lugar, tomar una mejor decisión por parte del operador, que permita mejorar el desempeño del sistema teleoperado, lo cual se traduce en disminuir la probabilidad de accidentes, mejorar la precisión en las tareas propuestas y reducir los tiempos de toma de decisión y ejecución de movimientos.

La forma más común de teleoperación que se encuentra hoy en día en funcionamiento en las industrias, es a través de cámaras de video que muestran en tiempo real el efector final del robot y el entorno de trabajo. En estos casos se utilizan varias cámaras para enfocar desde diferentes ángulos el lugar de operación, y por lo tanto, en la pantalla de control el operador posee varias ventanas con diferentes puntos de vista del sistema. Este método requiere que el operador posea una buena percepción de profundidad del robot con respecto al entorno, usando la información de todas las cámaras disponibles. Por otra parte, el operador requiere conocer las dimensiones reales del robot con respecto a las dimensiones de los objetos que hay en el entorno, para que de esta forma pueda evitar colisiones indeseadas. Todos estos detalles hacen que la teleoperación a través de video pueda resultar confusa, o bien, requiere que los

operadores se encuentren bien entrenados en la utilización de ese sistema en particular.

Una de las primeras publicaciones sobre teleoperación [1] muestra las limitaciones en las habilidades humanas con respecto a la orientación espacial en pantallas de monitoreo 2D o 3D. Para estos casos, se propone un sistema integrado con teleoperación global y sistema autónomo para áreas donde la teleoperación se vuelve difícil. Otra manera de mejorar el desempeño del usuario [2] es cambiando el espacio de operación, desde un espacio de trabajo (*work-space*) 3D a un espacio 2D de configuración de parámetros (*configuration-space*) donde las posiciones prohibidas del efector final son más fáciles de definir. Cambiar el espacio de operación cambia completamente las restricciones que tiene el sistema, y por lo tanto, el desafío de mejorar la experiencia del usuario en la teleoperación se mantiene sin una mejora directa. Por otra parte, se han desarrollado evaluaciones a entornos virtuales de teleoperación [3] usando cámaras estereoscópicas que permiten obtener una sensación de profundidad 3D, con lo cual se muestran los primeros pasos para recrear entornos virtuales 3D que se basan en el entorno de operación del robot.

La tecnología háptica se comenzó a utilizar en la teleoperación de robots [4] [5] usando la arquitectura maestro-esclavo. Los dispositivos hápticos permiten al operador del robot tener retroalimentación táctil (utilizando el sentido del tacto) del entorno real en el cual el robot está trabajando. Esta retroalimentación puede ser vibratoria como señal de alerta o bien ser proporcionada como una fuerza en el mismo controlador de mando maestro, que obliga al operador a cambiar la dirección de movimiento del dispositivo controlado esclavo. Por ejemplo, el dispositivo esclavo puede ser un brazo robótico que sigue los movimientos del brazo maestro cuando se está en un entorno libre de obstáculos. Si el brazo esclavo hace contacto con un obstáculo que le impide el movimiento, provoca una diferencia entre la posición real y la posición deseada. La fuerza de retroalimentación que se envía al dispositivo maestro y que es percibida por el usuario es proporcional a la diferencia de posición mencionada anteriormente.

La investigación en telerobótica es importante para mejorar las habilidades de los operadores en tareas de manipulación peligrosas, haciendo que los operadores puedan trabajar en un ambiente protegido y alejado de la faena. Por ejemplo en la industria minera [6] [7] se utiliza la teleoperación para controlar los brazos picadores de rocas *rock breaker teleoperation*. Se ha demostrado que la teleoperación de brazos picadores de rocas disminuye el riesgo de colisión, reduce el tiempo de entrenamiento de los operarios, aumenta el tiempo de trabajo de estas máquinas sin necesidad de realizar paradas, mejoras las habilidades de precisión de los usuarios que controlan estos dispositivos, y entrega beneficies en la mantención y costos de producción. En estos casos, se puede incluir un sistema de retroalimentación háptico, para mejorar aún más el desempeño de los operadores y aumentar los beneficios demostrados en las investigaciones [6] [7].

1.1.2. Definición del problema a abordar

En la presente tesis se incorpora un sistema de retroalimentación háptico a la teleoperación de un brazo robótico. La interfaz háptica implementada en particular está inspirada en la investigación de Rydén et al. [8] [9] en la cual se calcula la fuerza de retroalimentación a

partir de una *Point Cloud* obtenida con una cámara RGB-D. La nube de puntos entrega una representación de las superficies del entorno, o en otras palabras es una "fotografía" en 3D.

Con el algoritmo háptico incorporado al sistema de teleoperación, se genera una fuerza de repulsión que evita colisiones del efector del brazo robótico con algún obstáculo en el entorno de trabajo. Cuando se desea acercar el efector robótico a una superficie en particular, se puede asegurar estar lo más cerca posible sin colisionar, permitiendo el desplazamiento del efector por sobre una superficie que puede ser regular (lisa) o con relieves significativos. Este enfoque de teleoperación háptica no ha sido puesto a prueba en las investigaciones publicadas hasta la fecha. Para poder cuantificar el desempeño que tiene un operario al utilizar el sistema de teleoperación con la interfaz háptica, se diseñan una serie de pruebas para ser completadas por diferentes usuarios.

1.2. Objetivos

1.2.1. Objetivos generales

El objetivo general de este trabajo es mejorar la interacción entre los humanos y las máquinas la cual se lleva a cabo en diversas interfaces de usuario, haciendo más natural la transmisión de información desde el robot hacia el operador y así mejorar la toma de decisiones.

El objetivo final de la telepresencia es utilizar la mayor cantidad de sentidos humanos para recrear una realidad virtual tan similar al entorno real como sea posible. Con más retroalimentación sensorial a partir del mundo real, el operador que está inmerso en la realidad virtual puede reaccionar a los estímulos del entorno real más fácilmente [10], haciendo tareas en forma más precisa. Además, la persona puede reaccionar de forma más rápida frente a estímulos reflejos que en un entorno real surgen de manera espontánea.

1.2.2. Objetivos específicos

- Desarrollar interfaz de teleoperación de arquitectura Maestro (*Phantom Omni*) - Esclavo (brazo robótico).
- Escanear el entorno real de operación del robot con una cámara RGB-D *Microsoft XBOX Kinect*.
- Creación de entorno virtual de operación a partir de los datos de la *Point Cloud*.
- Calcular la fuerza de retroalimentación mediante el algoritmo de *proxy* [8] [9], a partir de los datos de la *Point Cloud*.
- Evaluar el sistema de teleoperación en forma cuantitativa.

1.3. Hipótesis

La hipótesis de este trabajo es que integrando la retroalimentación háptica a la retroalimentación visual en una teleoperación, se mejora la experiencia del usuario, el cual puede alcanzar mayor precisión, velocidad y confianza al utilizar el brazo robótico. Además, se facilita el entrenamiento de nuevos usuarios del sistema.

Creando una experiencia de operación más intuitiva para cualquier usuario, sin importar su conocimiento previo del sistema de teleoperación, permite mejorar el rendimiento y reducir los tiempos de entrenamiento.

1.4. Metodología

La metodología que se propone para desarrollar un sistema de teleoperación con retroalimentación háptica consiste en:

1. Analizar e implementar algoritmo de renderizado háptico a partir de un *proxy* de Rydén et al. [8] [9].

2. Modificar el algoritmo anterior, para ser usado en una aplicación de teleoperación.

3. Diseñar y construir interfaz de comunicación entre el brazo robótico y un computador externo "cliente" en un entorno de *ROS*.

4. Diseñar y construir sistema de adquisición de *Point Clouds* con respecto a distintos puntos de referencia dentro del entorno de trabajo del robot.

5. Diseñar y construir el sistema de control del robot incluyendo el algoritmo de renderizado háptico.

6. Evaluar el sistema de teleoperación háptico realizando la validación experimental con usuarios, obteniendo datos cuantitativos estadísticos.

1.5. Aportes del trabajo de tesis

Se propone un método de teleoperación con retroalimentación háptica que no se encuentra en la literatura hasta la fecha. Los métodos de teleoperación háptica encontrados basan su funcionamiento en sensores de fuerza, presión o proximidad, ubicados en el efector del robot. A diferencia de estos, el enfoque propuesto basa su funcionamiento en el cálculo de fuerzas de retroalimentación a partir de una nube de puntos, la cual es una representación virtual tridimensional del entorno real de trabajo del robot.

1.6. Estructura de la tesis

Esta tesis presenta las siguientes secciones: El Capítulo 2 muestra una revisión de las Definiciones y Conceptos básicos que se deben manejar para comprender la tesis, junto al Estado del Arte actual del problema planteado. El Capítulo 3 muestra el Desarrollo del Sistema de Teleoperación propuesto, el Capítulo 4 presenta la Validación Experimental de la metodología, y el Capítulo 5 entrega los Resultados y Análisis. El Capítulo 6 presenta las Conclusiones del proyecto y plantea el trabajo futuro. Finalmente se presentan las Referencias de donde se obtuvo la información, y los Anexos donde se encuentra una guía útil paso a paso para realizar futuros desarrollos utilizando el robot industrial.

2 Definiciones y Estado del arte

2.1. Definiciones

2.1.1. Háptica

La háptica es la utilización del sentido del tacto para la manipulación o percepción de objetos [11]. Para esto se utiliza la propiocepción que corresponde a la información que entrega el organismo sobre la posición de los músculos y por lo tanto, se obtiene en forma directa la posición relativa de las partes del cuerpo en relación al entorno con el cual se está interactuando [12].

En definitiva la utilización de dispositivos hápticos permite ocupar el sentido del tacto (incluida la propiocepción corporal) para interactuar con un entorno real o virtual. Estos dispositivos hápticos se pueden clasificar en dos tipos básicos: Los de retroalimentación vibrotáctil y los de retroalimentación de fuerza-torque [13].

a. Retroalimentación vibrotáctil

Basada en la respuesta de receptores sensoriales presentes en la piel [14], como por ejemplo los corpúsculos de *Meissner* (tacto ligero, vibraciones de menos de $50[Hz]$), corpúsculos de *Pacini* (vibraciones y presión mecánica), corpúsculos de *Ruffini* (cambios de temperatura), receptores de *Merkel* (presión y textura), y corpúsculo de *Krause* (sensación de frío).

En esta categoría se encuentran por ejemplo los dispositivos electrónicos de lectura *Braille* (Figura 2.1) donde la yema de los dedos se encarga de percibir la forma correspondiente a cada letra del alfabeto en forma independiente. Otro representante de este tipo de retroalimentación, son los controles (*joystick*) de videojuegos que poseen sistema de vibración (Figura 2.2), donde el usuario es capaz de percibir una alerta vibratoria en las manos cuando ocurre un evento de colisión en el entorno virtual.

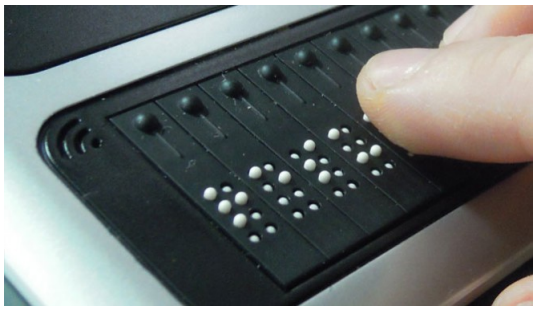

Figura 2.1: Dispositivo de lectura *Braille*.

Figura 2.2: *Joystick* vibrotáctil.

b. Retroalimentación de fuerza-torque

Los dispositivos de retroalimentación fuerza-torque se basan en la respuesta kinéstésica o propioceptiva del ser humano, la cual utiliza los receptores neuronales ubicados en los tendones y músculos. En este caso, se utiliza una fuerza que se opone al movimiento de alguna articulación del cuerpo humano y que nuestro sistema propioceptivo es capaz de identificar e incluso diferenciar la intensidad y dirección.

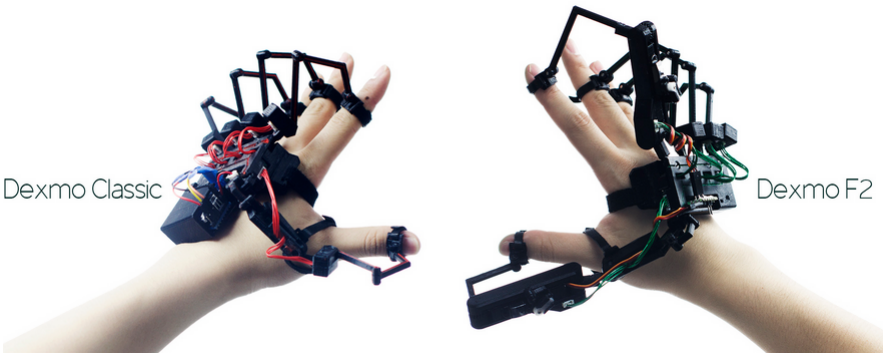

Figura 2.3: *Dexmo* exoesqueleto para las manos.

Entre los dispositivos que utilizan fuerza de retroalimentación, se encuentran diversos tipos de exoesqueletos (Figura 2.3), que facilitan el movimiento de articulaciones para personas con alguna discapacidad o bien, permiten la interacción del usuario con un entorno virtual. Por otra parte, existe una amplia gama de dispositivos hápticos utilizados principalmente

en modelación de objetos 3D que permite interactuar con el entorno virtual utilizando el sentido del tacto (Figura 2.4). Con estos dispositivos se logra obtener una sensación de estar "tocando" el objeto virtual con la punta de un lápiz o *stylus* que sostiene la mano.

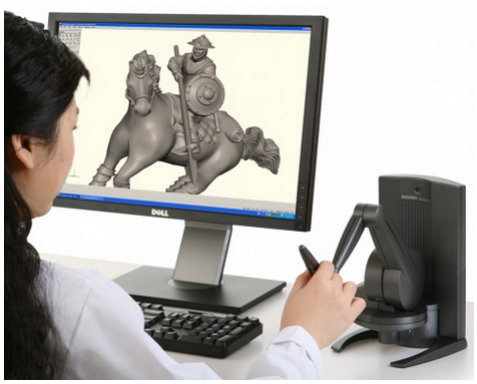

Figura 2.4: Dispositivo háptico *SensAble Phantom Desktop*.

A continuación se detallará el funcionamiento del dispositivo háptico utilizado en la propuesta de teleoperación de la presente tesis.

2.1.2. Dispositivo de interfaz háptica *Phantom Omni*

El dispositivo háptico *SensAble Phantom Omni* [15] posee 6 grados de libertad en lectura de posición (*encoders*), de los cuales 3 tienen efectores de fuerza. El usuario toma el "*stylus*" del *Phantom* como si fuera un lápiz utilizando una sola mano.

Este dispositivo se conecta directamente al computador a través del puerto *FireWire (IEEE 1394)*. Posee una frecuencia de actualización de posición de $1[KHz]$. En la Figura 2.5 se muestran los *joints* o articulaciones del dispositivo enumerados desde J_1 a J_6.

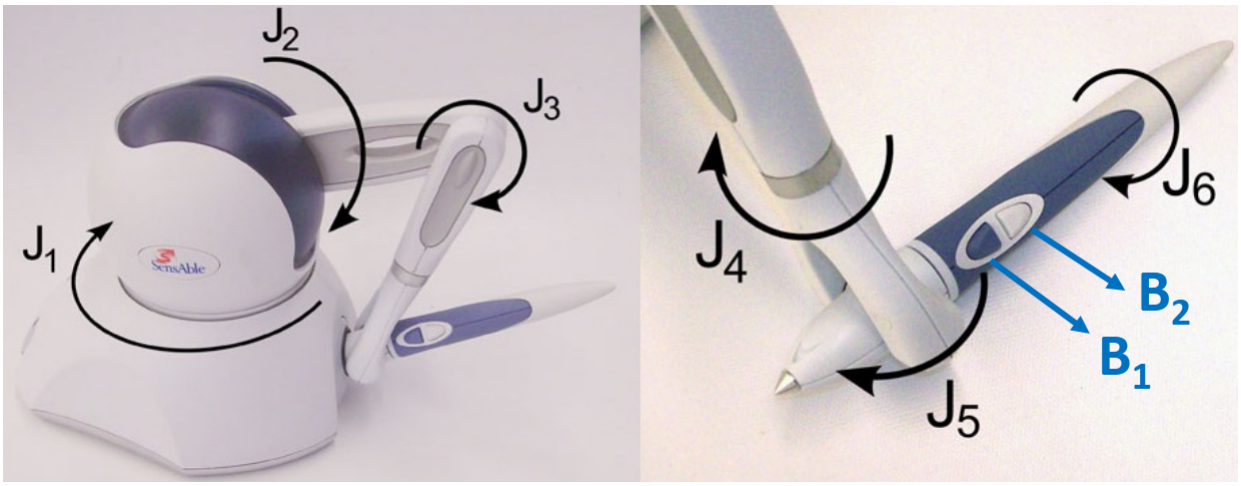

Figura 2.5: *Phantom Omni* joints y botones. Imagen obtenida de [15].

En los *joints* desde J_1 a J_3 se genera el torque necesario para representar un vector de fuerza cartesiano (x, y, z) en el *workspace* propio del dispositivo. Los botones B_1 y B_2 sirven

como entrada digital (tal como funcionan los botones del *mouse* del computador) y pueden cumplir alguna función específica programable.

2.1.3. Teleoperación Maestro-Esclavo

Los sistemas de teleoperación Maestro-Esclavo se caracterizan por tener dos dispositivos iguales o con una semejanza funcional completa, conectados con un sistema de comunicación bilateral (comunicación en ambos sentidos) [16]. Uno de los dispositivos se denomina Maestro, quien da las instrucciones de operación al otro dispositivo que se denomina Esclavo. Estas instrucciones de operación pueden ser llevadas a cabo a través de la Cinemática Directa o Cinemática Inversa (dependiendo de la funcionalidad que se requiera) del dispositivo esclavo.

La semejanza funcional completa quiere decir que el dispositivo Esclavo tiene sus características de movimiento completamente homologadas por el dispositivo Maestro, que puede ser controlado directamente por el usuario. En otras palabras, la funcionalidad del dispositivo Esclavo es una representación directa de la funcionalidad del dispositivo Maestro, con respecto al movimiento y su relación con el *workspace* propio. Un caso particular de dispositivo Maestro es conformado por una "Interfaz Humano-Máquina" o *HMI* por sus siglas en inglés, ya que que el elemento que está en contacto directo con el usuario.

La comunicación entre el dispositivo Maestro y el Esclavo puede ser llevada a cabo por diversos protocolos y procedimientos. Una de las características que se destacan de este método, es que permite al usuario interactuar con el dispositivo Maestro a kilómetros de distancia del dispositivo Esclavo, en un ambiente de trabajo seguro, sin importar las duras condiciones en donde se desempeña el dispositivo Esclavo.

2.1.4. Brazo robótico

Los brazos robóticos cumplen diversas funciones, en las cuales se requieren movimientos traslacionales o lineales y movimientos rotacionales [17]. Se pueden encontrar ejemplos de la utilización de brazos robóticos en diversos campos de la ciencia e ingeniería, desde líneas de ensamble en industria automotriz donde se requiere manipulación de piezas y soldaduras, trabajos "*pick and place*" en la industria de alimentos (para ordenar productos en sus empaques),desactivación de explosivos, operaciones en minería para manipular rocas, operaciones espaciales en satélites, etc.

Un brazo robótico está compuesto de cuatro elementos principales: La base, los *joints*, los *links* y el efector o *end-effector*. En la Figura 2.6 se observa un esquema general de los elementos básicos de un brazo robótico.

1. La base: Consiste en la estructura de apoyo del brazo, ubicada usualmente en la parte inferior. En general concentra el centro de masa de todo el brazo robótico, asegurando la estabilidad de los movimientos de las articulaciones, y evitando algún desequilibrio de la estructura.

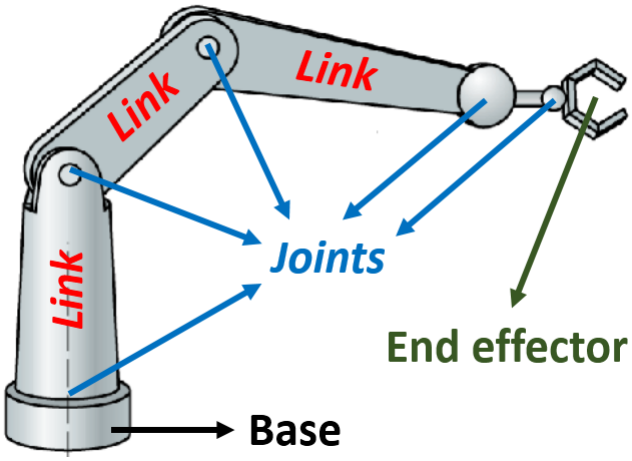

Figura 2.6: Elementos básicos de un brazo robótico.

2. *Joints*: Corresponden a las articulaciones del brazo, que son las partes móviles accionadas por distintos tipos de motores. Usualmente, el movimiento angular de cada *joint* está limitado dependiendo del tipo de brazo y de la funcionalidad deseada.

3. *Links*: Es la estructura rígida que une a los *joints*. Los *links* son construidos típicamente de metal y su forma, dureza y peso varían dependiendo de las aplicaciones del brazo robótico.

4. *End-effector*: El efector del brazo robótico es el elemento principal encargado de interactuar con el *workspace* del brazo. Se puede tratar de pinzas, puntas para soldar, sopletes, elementos para cortar, manos robóticas, etc. Se encargan de realizar la acción final en el entorno de trabajo, por lo tanto son tan diversos como el usuario necesite.

2.1.5. Cámaras RGB-D y *Point Clouds*

Una cámara RGB-D es un dispositivo que produce una *range image* en donde cada pixel de la imagen RGB tradicional es asociada a una distancia [18]. De esta manera, se genera una imagen con "profundidad" en donde se pueden detectar fácilmente cuáles puntos de la imagen están más cerca y cuales están alejados.

En la actualidad existen muchas alternativas de cámaras RGB-D al alcance de usuarios y desarrolladores. El dispositivo que más se ha expandido en el mercado es el *Microsoft XBOX Kinect* que es utilizado principalmente como interfaz para juegos de video. En la Figura 2.7 se observa un esquema del *Kinect* en donde se identifica de izquierda a derecha, el proyector de la grilla de puntos infrarrojos, la cámara RGB, y el sensor de infrarrojos.

En una cámara RGB-D se genera una grilla de puntos infrarrojos que se proyecta sobre las superficies [19]. Luego a través de un sensor infrarrojo, se detecta la posición e intensidad de cada uno de los puntos proyectados, y por lo tanto, indirectamente se obtiene la distancia de cada punto al objeto más cercano en el cual se proyectó. Como se observa en la Figura 2.8 la grilla de puntos de luz infrarroja proyectada sobre los objetos no es visible por el ojo humano, pero si se puede detectar con una cámara de video tradicional (y por supuesto, con

Figura 2.7: Cámara RGB-D *Microsoft XBOX Kinect*.

el sensor especializado en dicha frecuencia). Por otra parte, se observa que los objetos más cercanos a la cámara producen "sombras" en donde los puntos de la grilla no llegan. Estas zonas oscurecidas son lugares donde no se puede determinar si existen objetos o es espacio libre. Para solucionar esta indeterminación, se debe buscar una nueva posición de la cámara RGB-D para abarcar un nuevo ángulo de visión y completar la imagen 3D generada a partir de la nube de puntos o *Point Cloud*.

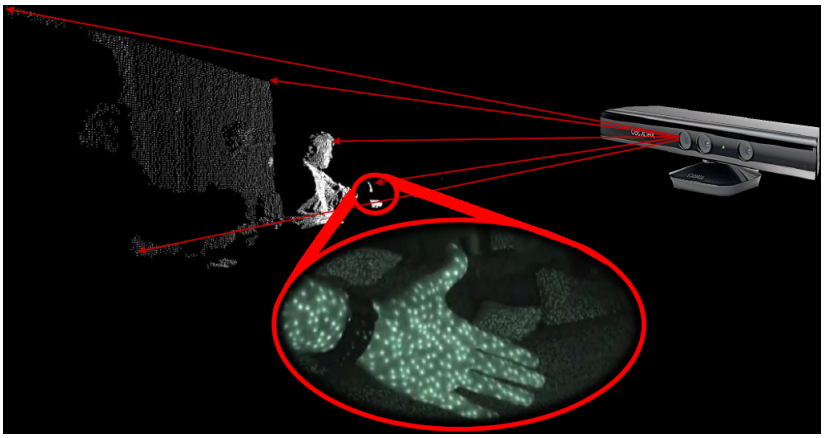

Figura 2.8: Captura de *Point Cloud* usando cámara RGB-D *Microsoft XBOX Kinect*.

2.1.6. *Voxel* y *Octree*

Un *voxel* corresponde a la representación espacial tridimensional de la mínima unidad (el cubo más pequeño) que conforma una imagen de profundidad [20]. De forma análoga al *pixel*, que es la mínima unidad que conforma una imagen tradicional en 2D, el *voxel* es la mínima unidad de una imagen 3D. La resolución de las imágenes determina el tamaño del *voxel*, ya que a mayor resolución es menor el tamaño de cada cubo elemental.

Los *voxeles* se utilizan para estandarizar la resolución de una *Point Cloud*, en donde cada punto se ubica en una posición espacial arbitraria. Cuando se aplica un *voxel grid* o grilla de *voxeles* sobre una *Point Cloud*, es posible reducir la cantidad de puntos que la conforman, minimizando las redundancias que se producen al definir un objeto en el espacio cartesiano. En la Figura 2.9 se observa una curva azul obtenida a partir de una nube de puntos en el espacio tridimensional, que puede ser representada en la grilla de *voxeles* (cubos rojos) con

una cantidad menor de información. En este caso se pierde resolución de la curva azul, pero se aumenta la velocidad para poder procesar la nueva *Point Cloud* conformada por los puntos que representan los centros de cada uno de los *voxeles*.

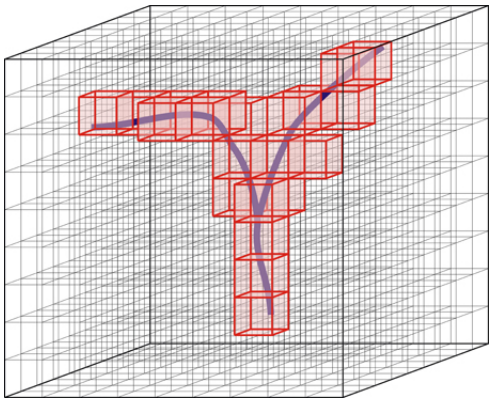

Figura 2.9: Voxel grid. Imagen obtenida de [21].

Un *octree* es una estructura de datos de tipo "árbol" en donde cada nodo interno posee ocho hijos (Ver Figura 2.10) [22]. Esta estructura de datos es usada para particionar el espacio tridimensional en ocho octantes. Cada uno de estos octantes se puede representar por un *octree*, en el cual se definen nuevamente ocho octantes y así recursivamente.

La *Point Cloud*, luego de ser reducida utilizando un *voxel grid*, se puede organizar en la estructura *octree* para reducir el tiempo de búsqueda de puntos en el espacio y de esta forma aumentar el rendimiento del algoritmo de renderización háptica a partir de nube de puntos.

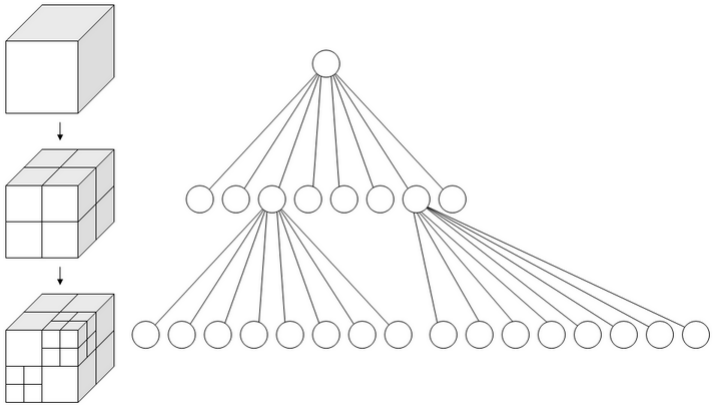

Figura 2.10: Esquema de Octree. Imagen obtenida de [23].

2.1.7. Cinemática directa

La cinemática directa o *Forward Kinematics (FK)* es el proceso en el cual se obtiene la posición del efector (*end-effector*) del brazo robótico a partir de los valores de cada uno de los *joints* [24]. La posición del efector se calcula a partir de los movimientos en cadena de cada uno de los *joints* desde la base hasta el extremo del efector. La solución de la cinemática

directa es única, lo cual significa que para una combinación de ángulos de *joints* determinada, existe una sola posición de efector final posible.

A continuación se muestra un ejemplo de la utilización de la cinemática directa para un brazo robótico de dos grados de libertad [25]:

- En el esquema de la Figura 2.11 se busca encontrar la posición $X = (x, y)$ correspondiente a la ubicación del efector final del brazo.
- Se conocen las posiciones deseadas (ángulos) de los *joints* 1 y 2, denominados Θ_1 y Θ_2 respectivamente.
- Se conocen los largos de los *links* 1 y 2 denominados l_1 y l_2 respectivamente, con $(0, 0)$ el origen de coordenadas para el sistema.

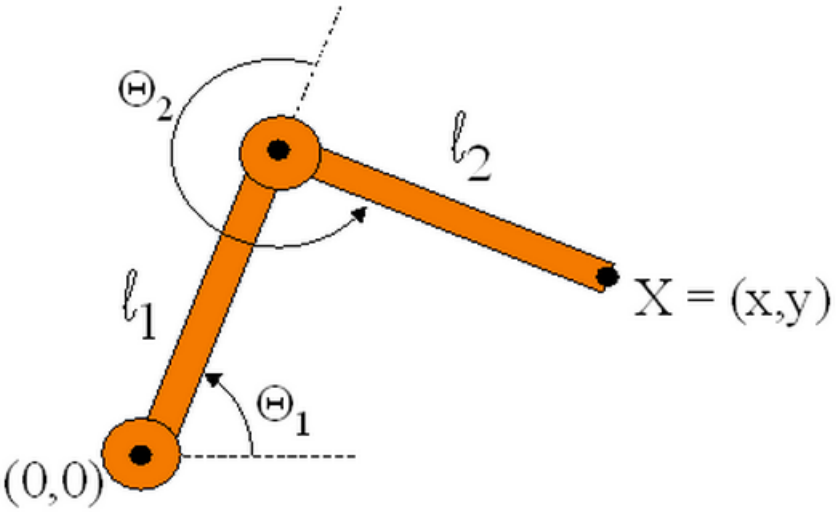

Figura 2.11: Ejemplo cinemáticas - Brazo con dos grados de libertad.

Entonces, la posición del efector $X = (x, y)$, dados los ángulos de los *joints* es determinada por las siguientes ecuaciones:

$$x = l_1 cos(\Theta_1) + l_2 cos(\Theta_1 + \Theta_2) \tag{2.1}$$

$$y = l_1 sin(\Theta_1) + l_2 sin(\Theta_1 + \Theta_2) \tag{2.2}$$

2.1.8. Cinemática inversa

La cinemática inversa o *Inverse Kinematics (IK)* es el proceso en el cual se obtienen los ángulos de los *joints* para alcanzar una posición de efector final determinado [24]. Este cálculo es mucho más difícil que el de cinemática directa, ya que es más trabajo de cómputo y la complejidad de las ecuaciones se eleva considerablemente con cada grado de libertad nuevo agregado al brazo. Cuando se resuelve este problema, se pueden encontrar tres casos: El primer caso corresponde a una solución única (es el más difícil de encontrar), el segundo caso corresponde a múltiples soluciones válidas (muchas combinaciones de ángulos de *joints* para obtener la misma posición del efector) y finalmente un tercer caso en donde la solución

no existe (con ninguna combinación de ángulos de las articulaciones se puede alcanzar la posición del efector deseada).

En general, la cinemática inversa es la más utilizada, ya que en la mayoría de los problemas en los cuales opera un robot debe movilizar su efector o *end-effector* desde un punto de partida A a un punto de llegada B en el espacio, sin importar cómo realice este movimiento. El problema principal es que para lograr llegar a B, el robot primero requiere la información de la cantidad angular que debe mover cada *joint*. Es por esto que el primer paso antes de realizar el movimiento, es la obtención de la cinemática inversa.

A continuación se muestra un ejemplo de la utilización de la cinemática inversa para un brazo robótico de dos grados de libertad [26]:

- En el esquema de la Figura 2.11 se busca encontrar los ángulos Θ_1 y Θ_2 correspondientes a los *joints* del brazo.
- Se conoce la posición $X = (x, y)$ correspondiente a la ubicación deseada del *end-effector* del brazo.
- Se conocen los largos de los *links* 1 y 2 denominados l_1 y l_2 respectivamente, con $(0, 0)$ el origen de coordenadas para el sistema.

Entonces, los ángulos de los *joints* Θ_1 y Θ_2 son determinados a partir de la posición del efector $X = (x, y)$ utilizando las siguientes ecuaciones:

$$\Theta_1 = \frac{-(l_2 sin(\Theta_2)x + (l_1 + l_2 cos(\Theta_2)))y}{(l_2 sin(\Theta_2))y + (l_1 + l_2 cos(\Theta_2))x} \tag{2.3}$$

$$\Theta_2 = \arccos\left(\frac{x^2 + x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right) \tag{2.4}$$

2.1.9. Prueba-T para muestras apareadas

La "Prueba-T para muestras apareadas" o "*Paired-sample T-test*" es una técnica estadística que se utiliza para comparar dos grupos de datos o muestras estadísticas que están correlacionadas [27]. Este tipo de análisis estadístico es ideal para comprobar la hipótesis presentada en esta tesis, y es utilizada en en Capítulo donde se muestran y se analizan los resultados.

La aplicación del "*Paired-sample T-test*" es ideal para los casos en los cuales se quiere conocer si el uso de una tecnología o herramienta tiene un impacto real comparado con la no utilización de esta. En el caso de la presente tesis, el primer grupo de muestras corresponde a los resultados obtenidos con retroalimentación háptica y el segundo grupo corresponde a los resultados obtenidos sin esta tecnología.

A continuación se describen los pasos para realizar este *T-test* pareado:

1. Se definen dos hipótesis del *test*. La primera corresponde a la hipótesis nula, la cual asume

que la media de los dos sets de muestras es la misma. La segunda hipótesis será la hipótesis alternativa, la cual asume que la media de los dos sets de muestras no son iguales.

2. Se selecciona el nivel de significancia, el cual corresponde al umbral a partir del cual se considera válida la primera o la segunda hipótesis. Si el resultado obtenido al realizar este *test* es menor al nivel de significancia, entonces se dice que los resultados son estadísticamente significativos, lo cual implica que utilizar la tecnología estudiada si posee un impacto estadístico en los resultados obtenidos.

3. Se obtiene el resultado de la Prueba-T para muestras apareadas, el cual está determinado por la Ecuación 2.5.

$$P = \frac{\mathrm{d}}{\sqrt{\frac{s^2}{n}}} \tag{2.5}$$

donde P es el resultado de la Prueba-T para muestras apareadas, d es la diferencia de la media de los dos grupos de muestras, s^2 es la varianza de las muestras y n es el número de muestras.

Realizando una analogía de este *test* estadístico con las probabilidades, se dice que el resultado de comparar dos grupos de muestras es estadísticamente significativo con un nivel de significancia P, si tiene una probabilidad menor a P de que esa diferencia mostrada al comparar los dos grupos de muestras se deba a una acción aleatoria o azar. Es por esto que un estudio presenta resultados más concluyentes si es capaz de mostrar un nivel de P de significancia más pequeño (lo más cercano a cero posible). Para el caso presentado en esta tesis, el nivel de significancia corresponde al 10 % , el cual es un criterio utilizado para validar resultados en pruebas de teleoperación bajo distintas condiciones de control. En la sección de resultados se mostrará y analizará el valor de significancia obtenido en cada prueba.

2.1.10. *ROS: Robot Operating System*

ROS (Robot Operating System) corresponde a un conjunto de programas y librerías usados en diversos campos de la robótica y que se relacionan mutuamente a través de mensajes (en diferentes tópicos) y servicios [28]. Esta forma de organizar el desarrollo en robótica, permite facilitar la utilización de herramientas típicas y agregar otras nuevas sin tener que volver a desarrollar la plataforma de comunicación. *ROS* presenta un marco de trabajo (*framework*) en donde desarrollar y testear software para el robot (sensores, actuadores, procesamiento de datos, etc.) es simple y colaborativo, ya que múltiples desarrolladores pueden trabajar en distintas partes del robot sin interferir en el trabajo de los demás.

2.1.11. *Hardware-in-the-loop simulation*

El sistema de simulación "*Hardware-in-the-loop*" o "HIL" es una herramienta utilizada durante el desarrollo de sistemas embebidos que requieren proceso de datos en tiempo real [29]. Muchos sistemas que se requieren desarrollar son de alta complejidad técnica y logística, por lo tanto, las pruebas que se pueden llevar a cabo a través de un prototipo son muy costosas. El costo de realizar pruebas con un prototipo puede implicar un gasto monetario (insumos, infraestructura, etc.), o bien un gasto de tiempo al tener que detener la operación de la planta (o sistema) para agregar el nuevo dispositivo y realizar pruebas de funcionamiento. Para realizar estas pruebas y comprobar el correcto desempeño del nuevo dispositivo, se evita utilizar directamente los componentes de la planta y se reemplazan por un sistema "HIL" que permite proporcionar las mismas señales de entrada y salida del sistema, con lo cual se pueden realizar pruebas tal y como resultaría en la planta real.

El sistema "HIL" es en términos simples una simulación de la planta real, por lo tanto se puede interactuar con ella a través de las señales de entrada (sensores, interfaces de usuario, etc.), obteniendo señales de salida (actuadores, indicadores, etc.), las cuales son idénticas a las que se pueden obtener de la planta real, como si realmente estuviera conectada al dispositivo que se quiere probar. La principal ventaja de utilizar "HIL" es que no se requiere detener el funcionamiento de la planta real para poner a prueba un dispositivo nuevo, además de poder detectar algún error crítico en el prototipo antes de ser conectado al resto de la planta y ocasionar un problema mayor.

Para el desarrollo del sistema de control del brazo robótico, se utiliza un simulador "HIL" a través del software *Gazebo*, el cual es un simulador del robot y de su entorno de trabajo físico. A través de *Gazebo* se pueden agregar todo tipo de sensores al robot, recibiendo las señales medidas tal y como si fueran mensajes de los sensores reales, lo cual permite una evaluación rápida del sistema completo. Cuando se tiene la versión definitiva del sistema de control, simplemente se desactiva el "HIL" y se realizan las conexiones a los actuadores y sensores reales, para finalmente ejecutar el programa de control sin necesidad de realizar mayores cambios.

2.1.12. *Gazebo*

El conjunto de paquetes o librerías *Gazebo* conforman un simulador físico gratuito para desarrollos en robótica, con una interfaz gráfica de fácil uso [30]. Estos paquetes están completamente integrados a *ROS*, utilizando el mismo tipo de mensajes para la comunicación entre nodos. Existe una gran comunidad de desarrolladores independientes para *Gazebo*, lo cual permite una colaboración activa entre distintos programadores, resolviendo dudas y proponiendo nuevos desafíos.

Este tipo de simuladores permite poner a prueba de forma rápida algoritmos nuevos sin necesidad de utilizarlos en el robot real, ya que permite una interacción bastante realista entre el dispositivo que se pone a prueba y el entorno de trabajo. Al poseer un motor físico robusto y gráficos de alta calidad, es capaz de emular el comportamiento del robot cuando

está expuesto a gravedad, colisiones con el entorno y otros dispositivos móviles, activación de distintos tipos de sensores también simulados, instrucciones de un operador en tiempo real, entre otros. Por otra parte, *Gazebo* ofrece simular eficientemente poblaciones de robots en entornos complejos exteriores (al aire libre) o interiores (por ejemplo dentro de una casa), lo cual permite una gran flexibilidad a la hora de resolver desafíos en el campo de la robótica móvil y la visión computacional.

2.2. Estado del arte

2.2.1. Teleoperación Maestro-Esclavo Háptica de brazo robótico

La teleoperación maestro-esclavo con retroalimentación háptica presenta muchos enfoques de control distintos en la literatura. J. Park et al. [31] muestra una conexión virtual entre el efector del robot esclavo y el dispositivo maestro, a través de un sistema elástico o "resorte" el cual es ampliamente usado en otros desarrollos. El efector del robot esclavo sigue la posición objetivo dada por el efector del controlador maestro mapeado en el espacio de trabajo del esclavo. El control se lleva a cabo a través de la cinemática inversa del brazo, con lo cual sólo se necesita saber la posición cartesiana de ambas puntas efectoras (esclavo (x_S, y_S, z_S) y maestro (x_M, y_M, z_M))y posteriormente se realiza una transformación lineal proporcional para llevar la posición del efector maestro desde su propio espacio de trabajo ((x_M, y_M, z_M)), hacia el *workspace* del brazo esclavo (*slave goal* (x_{SG}, y_{SG}, z_{SG})). El brazo esclavo entonces se moverá desde su posición actual (x_S, y_S, z_S) hasta el punto objetivo (x_{SG}, y_{SG}, z_{SG}) determinado por la posición del efector maestro.

En el caso de que no existan obstáculos entre la posición del brazo robótico (x_S, y_S, z_S) y el objetivo (x_{SG}, y_{SG}, z_{SG}), entonces se llegará hasta esa posición sin problemas. Si existe algún objeto que interrumpa el paso del brazo robótico en forma mecánica (a través de una colisión por ejemplo), el brazo no podrá continuar con el movimiento y se generará una diferencia entre la posición actual y la posición deseada. Entonces la fuerza háptica F_H que retorna al controlador maestro, será proporcional (K constante de proporcionalidad) a esa diferencia de posición:

$$F_H = K \begin{bmatrix} x_S - x_{SG} \\ y_S - y_{SG} \\ z_S - z_{SG} \end{bmatrix} \tag{2.6}$$

La ecuación anterior corresponde a la *Ley de Hook* o Ley de elasticidad que indica que "la fuerza F necesaria para extender o comprimir un resorte una cierta distancia X es proporcional a esa distancia [32], con K un factor de rigidez (*stiffness*) característico del resorte":

$$F = -KX \tag{2.7}$$

La constante de proporcionalidad K debe ser calibrada dependiendo del tipo de dispositivo de control háptico, para que el usuario alcance a tener un tiempo de reacción adecuado y así poder tomar una nueva decisión con respecto al movimiento del robot. Es por esto que se dice que la fuerza de retroalimientación entregado al usuario hace que este ultimo pueda cerrar el lazo o *loop* de control.

El enfoque propuesto por J. Park et al. [31] necesita un contacto real entre el robot esclavo y el entorno de trabajo, para obtener la fuerza de retroalimentación. Cuando se requieren aplicaciones en las cuales no se puede permitir contacto entre el robot esclavo y el ambiente se deben utilizar otras metodologías. En algunos casos no se recomienda generar contacto

físico entre el robot y los objetos, ya que estos últimos pueden resultar con daños debido a que muchas veces los brazos robóticos son de tipo industrial (pesados y con mucha fuerza en sus motores, lo cual entrega un *momentum* de gran magnitud).

Una manera de mejorar la fuerza de retroalimentación háptica propuesto por P. Chotiprayanakul et al. [33] es agregar un campo de fuerza repulsivo tridimensional a cada objeto u obstáculo en el entorno de trabajo de robot. Este campo de repulsión se agrega al resorte virtual que conecta la posición del efector esclavo con la posición deseada del robot, explicada anteriormente. El campo de fuerza de cada objeto, se puede calcular antes del momento de la teleoperación, con los datos obtenidos de algún sensor de distancias como *scanners* lásers, cámaras RGB-D, etc. Otra forma de obtener el campo de fuerzas, es a través de un sensor de proximidad ubicado en el efector del robot, con lo cual se obtiene el dato de distancia al obstáculo más cercano. El *force field* es en realidad una suma de fuerzas calculadas de manera proporcional a la menor distancia desde el efector del robot hasta el objeto más próximo. Una manera de mejorar este algoritmo es obtener la distancia menor entre los obstáculos del entorno y cada *joint* (articulación) o *link* (segmento rígido) del brazo robótico, para lo cual se requieren más sensores de proximidad a lo largo del brazo. Esto último mejora considerablemente la prevención de colisiones, pero al mismo tiempo agrega más restricciones al espacio de trabajo libre permitido.

El desarrollo presentado en la investigación de M. Mamdouh et al. [34] muestra a la teleoperación maestro-esclavo usando el resorte virtual de J. Park et al. [31], agregando un sensor de fuerza y torque ubicado en el efector del brazo robótico esclavo. El sensor de fuerza cumplirá su función cuando haga contacto real con el objeto en el ambiente de trabajo, enviando una señal que será proporcional en magnitud a la fuerza de retroalimentación. Uno de los inconvenientes de la utilización de estos sensores para teleoperación, ocurre cuando el robot está en movimiento, ya que las vibraciones producen ruido en el sensor de fuerza y por lo tanto la fuerza retroalimentada posee un ruido intrínseco que debe ser quitado con un filtro pasa bajos. Otro aspecto a considerar es la distribución de sensores de fuerza en torno a los lugares donde es probable tener contacto con el robot, ya que posiblemente se requieran mucho más de un sensor para tener una información más completa. Los sensores entregan un valor escalar de fuerza, pero idealmente se quisiera medir un valor vectorial que indique la dirección de la fuerza en las tres dimensiones, lo cual se puede obtener sumando los resultados de fuerza obtenidos en diferentes sensores ubicados en lugares definidos de los *joints* o *links* del robot.

2.2.2. Teleoperación con retardo temporal

El desarrollo de la teleoperación con retardo temporal (*time-delayed teleoperation*) es motivada por diversas aplicaciones en las cuales el largo tiempo de comunicación entre los dispositivos Maestro y Esclavo es inevitable. En esta categoría se encuentran las teleoperaciones en misiones espaciales, que son de mucha utilidad para la mantención de satélites (alargando la vida útil de estos), pero la comunicación entre la Tierra y los dispositivos controlados en órbita (*on-orbit*) poseen un tiempo de retardo inherente lo cual reduce considerablemente el desempeño en la telemanipulación [35].

El equipo de *Satellite Servicing Capabilities Office (SSCO)* [36], que trabaja en el *Goddard Space Flight Center* de la *NASA*, investiga y desarrolla nuevas tecnologías para realizar reparaciones y asistencia técnica remota en naves espaciales o satélites en órbita [37].

Figura 2.12: Teleoperación en la Estación Espacial Internacional (*ISS*). Imagen obtenida del sitio de la *SSCO-NASA* [36].

Otra aplicación en la cual se destaca la teleoperación con retraso temporal, es en las misiones de servicio (mantención y reparación) del Telescopio Espacial Hubble, en los cuales las manipulaciones en órbita *on-orbit* fueron fundamentales para extender la vida útil del telescopio [38] [39] [40].

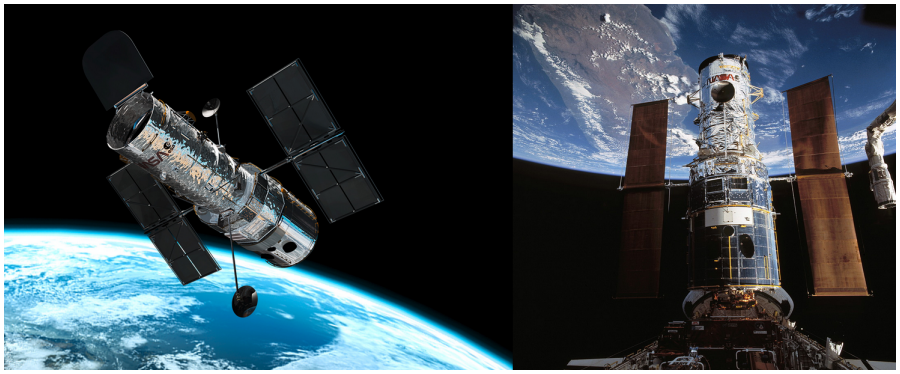

Figura 2.13: Izq: Telescopio Espacial Hubble. Der: Mantención del Hubble a través de teleoperación *on-orbit*. Imagenes [39].

Una de las investigaciones más recientes muestra el proceso de teleoperación para recargar combustible en órbita [41]. Se desarrolló la teleoperación asistida a través de *virtual-fixture* basado en el conocimiento previo que se tiene del entorno de trabajo real del brazo robótico. Los trabajos de T. Xia et al. [42] [43] muestran la metodología de telemanipulación usando este sistema. La contribución de esta tesis permite la interacción del brazo robótico con un entorno de trabajo desconocido (a diferencia de las investigaciones anteriores), controlando la posición del efector del robot a través de fuerza de retroalimentación háptica, sin importar el tiempo de retraso en la comunicación entre el lado Maestro y el Esclavo.

Un enfoque distinto es propuesto por J. Zainan et al. [44], donde la teleoperación se basa en la realidad virtual para trabajar con mensajes que tienen *delay* entre un robot espacial en

órbita y la estación de control en Tierra. La etapa fundamental en este tipo de enfoque, es el ajuste del modelo virtual con el entorno real de trabajo, ya que debe tener el menor error posible para evitar cualquier diferencia entre la percepción del operador y el movimiento real del robot con respecto al *workspace*. Con el enfoque propuesto por J. Zainan et al. [44] es posible reducir considerablemente el error de posición del *workspace* virtual con respecto al mundo real, aumentando la precisión en la teleoperación con tiempo de retardo.

Una aplicación de la investigación mencionada anteriormente, es la utilización de un *fixture* virtual para entornos de trabajo no estructurados presentado por Z. Jiang et al. [45], en cuyo caso la fuerza de retroalimentación es utilizada para guiar al operador (*teleoperación compartida*) resolviendo una tarea de seguimiento de trayectoria, aumentando su precisión.

Otra aplicación del enfoque mostrado anteriormente, es una teleoperación con tiempo de retardo variable para robot de cirugía presentado por Z. Boroujeni et al. [46], donde es necesario utilizar un esquema de retroalimentación adaptivo para trabajar con un modelo basado en incertezas (modelo estocástico).

2.2.3. Clasificación de Sistemas de Teleoperación

Se pueden clasificar los sistemas de teleoperación en sistemas bilaterales, compartidos, y semi-autónomos [47]. A continuación se detallan estas tres clases:

a. Sistema bilateral

La teleoperación bilateral utiliza las instrucciones directas desde el robot o controlador maestro hacia el robot esclavo, y en el sentido opuesto la retroalimentación desde el robot esclavo hacia el maestro. Como su nombre lo indica existe una comunicación simétrica, en el cual el emisor y receptor de los mensajes son ambos extremos de la teleoperación. El operador se encarga de cerrar el circuito de comunicación (*closed-loop scheme*) y es quien toma la decisión de un nuevo movimiento en base a la información que entrega el dispositivo esclavo.

En este trabajo de tesis en particular, se presenta un sistema bilateral maestro-esclavo de teleoperación. En el siguiente capítulo se mostrarán los detalles de desarrollo de este tipo de sistemas bilaterales.

b. Sistema compartido

La teleoperación compartida (*shared control*) se basa también en un enfoque maestro-esclavo, pero en este caso se tiene conocimiento previo de la trayectoria ideal o el punto objetivo que se debe alcanzar, los cuales no cambian con el desarrollo de la tarea de teleoperación. La retroalimentación asiste los movimientos del usuario, facilitando el alcance de la trayectoria o posición ideales. En este caso el operador no debe oponerse a la retroali-

mentación, sino que debe seguirla y tratar de copiar los movimientos ideales preestablecidos propuestos por el sistema. Ejemplos de la utilización de un sistema de control compartido se encuentran en los trabajos de P. Griffiths et al. [48] y J. van Oosterhout et al. [49], en donde el usuario es guiado por la fuerza aplicada en el dispositivo maestro con el objetivo de alcanzar un punto específico en el extremo del dispositivo esclavo.

Este sistema de teleoperación es ideal para el entrenamiento de usuarios, en donde se quiere mejorar su desempeño de motricidad y coordinación. También se puede utilizar para reducir el ruido provocado por los temblores en las manos o brazos de los operarios causados por el estrés y cansancio después de una larga jornada de manipulación. Este ruido en el controlador se reduce generando una fuerza de retroalimentación que se oponga a los movimientos de corto alcance y que ocurran con una frecuencia determinada. Por otra parte, este sistema es utilizado como herramienta de rehabilitación para pacientes con problemas neurológicos o musculo-esqueléticos, en donde se puede facilitar y promover la repetición de movimientos de las extremidades.

c. Sistema semi-autónomo

Los sistemas semi-autónomos de teleoperación utilizan tanto el enfoque bilateral como bloques programados de comportamientos autónomos entre el robot esclavo y el área de operación. Por ejemplo, se pueden tener rutinas programadas para que el efector del robot esclavo se posicione en cierto lugar seleccionado y luego manipule un objeto preestablecido como un cilindro, esfera o cubo y luego llevarlo hacia la zona objetivo. Este tipo de sistemas semi-autónomos son útiles para ciertas operaciones en las cuales se conoce el entorno de teleoperación y los objetos con los cuales el robot esclavo se relaciona.

2.2.4. Sistemas virtuales en la teleoperación

a. Campo Potencial Artificial

Un campo potencial artificial es un método clásico de planificación de trayectorias en robótica [50]. Con esta metodología, se busca simular un campo de fuerzas similar al que experimenta una partícula cargada cuando pasa a través de un campo eléctrico.

En el entorno de operación del robot, se definen los obstáculos como elementos que poseen la misma "carga eléctrica" que el robot, y por lo tanto se produce una fuerza de repulsión. Por el contrario, el punto objetivo o meta, posee una "carga" de signo contrario al robot y por lo tanto se produce una fuerza de atracción. La combinación de estas fuerzas atractivas y repulsivas, produce líneas de campo de fuerza por el cual se desplaza el robot, sugiriendo una trayectoria cuyo trabajo total sea mínimo. En la Figura 2.14 se observa la representación 3D de los obstáculos y el punto objetivo, además de las líneas de campo que estos producen y la curva de desplazamiento ideal del robot.

Esta metodología es recomendada cuando se tiene claridad del punto objetivo al cual

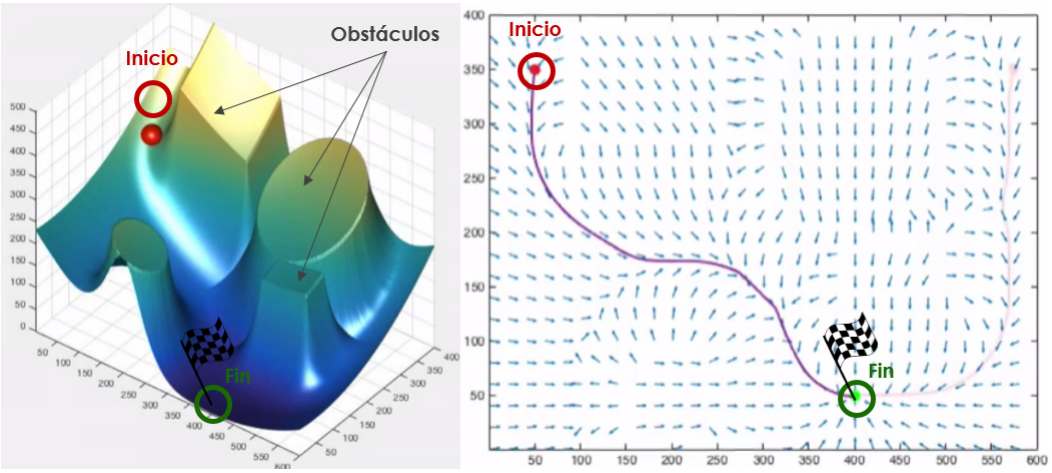

Figura 2.14: Esquema 3D de un Campo Potencial Artificial y su representación como líneas de campo eléctrico. Imagenes [51].

debe llegar el robot, para facilitar su desplazamiento. Dentro de las ventajas que se destacan de este método es que se pueden generar trayectorias en tiempo real, a medida que se van actualizando las lineas de campo de fuerza. Además, las trayectorias son suaves, evitando posibles colisiones con los obstáculos del entorno. La principal desventaja de este método es que el robot puede caer en un mínimo local del campo de fuerza, y por lo tanto queda atrapado en una zona donde no debería estar.

Una de las principales diferencias del método de Campo Potencial Artifical con la metodología propuesta en esta tesis, es que la fuerza del Campo Potencial actúa a cualquier distancia a partir del obstáculo, es decir, la fuerza de repulsión disminuye con la distancia a medida que el efector del robot se aleja de este, pero no desaparece. En cambio, la metodología de teleoperación basada en *Point Clouds* propone una fuerza repulsiva sólo si el efector está dentro de un margen de colisión.

b. Espacio de Configuración

El Espacio de Configuración es una forma de representar todas las posiciones disponibles del efector de un robot, considerando sus limitaciones intrínsecas, como los grados de libertad y sus ángulos de movimiento, y además las limitaciones del entorno de operación determinadas por los obstáculos [52]. Con esta metodología se puede determinar las posiciones del efector del robot en las cuales se asegura que no tendrá colisiones con el entorno, independientemente del número de grados de libertad que posee.

A diferencia de la metodología propuesta en esta tesis, el espacio de configuración restringe el movimiento del efector del robot, y por lo tanto, este no llega a interactuar con los objetos virtuales del entorno. Se evitan colisiones al limitar previamente el alcance de movimiento de las articulaciones del robot. En la Figura 2.15 se observa el espacio de posiciones de los obstáculos y el efector del robot, y además la representación del espacio de configuraciones para el caso con dos grados de libertad.

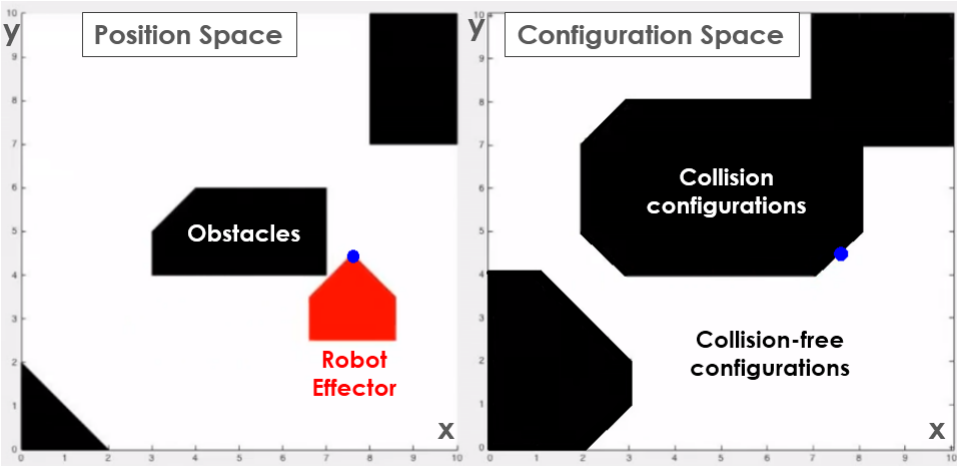

Figura 2.15: Esquema de posición de obstáculos junto al efector de un robot y su representación en el Espacio de Configuración. Imagenes [53].

c. Elemento virtual *Proxy*

Se define el elemento virtual *proxy* como una representación de la posición del efector del robot real en el ambiente de trabajo simulado [54]. Usualmente esta representación es una esfera, cuyo centro se define por la posición del efector del robot y su radio está determinado por la distancia máxima a la cual se permite acercar dicho punto a los objetos del entorno virtual.

El *proxy* interactúa "físicamente" con el entorno simulado siguiendo todos los movimientos del robot real, pero con la restricciones impuestas dentro de entorno virtualizado. Si el *proxy* hace contacto con algún elemento del entorno virtual, entonces este limita su desplazamiento y por lo tanto no puede penetrar o atravesar dicho objeto. El entorno virtual puede construirse como una representación del entorno real de trabajo para realizar teleoperaciones. Para el caso de una teleoperación háptica, el *proxy* permite calcular la fuerza de retroalimentación, cuando su posición deseada sea distinta a su posición real actual. Esta diferencia de posición es proporcional a la fuerza de repulsión que genera el objeto virtual que se está intentando atravesar con el *proxy*. Con este enfoque se permite "tocar" objetos virtuales tan cerca como el radio del *proxy* lo permita, y con esto se evita tener un contacto real del robot con el objeto físico en el entorno de operación.

d. Sistema de evasión de colisiones semi-autónomo

El algoritmo de evasión de colisiones de A. Leeper et al. [55] (*Constraint-Aware Teleoperation*) para teleoperación, utiliza las restricciones espaciales del entorno de trabajo real del brazo robótico, para generar zonas de movimiento libre y zonas prohibidas. Utilizando una cámara RGB-D se escanea el *workspace* real del robot, generando un *workspace* virtual que representa la posición y dimensiones de todos los objetos en el entorno. El desarrollo de A. Leeper et al. [55] sólo permite una teleoperación semi-autónoma, en donde el usuario determina el punto objetivo al cual se desea mover el efector del robot, y el algoritmo calcula

la trayectoria en forma autónoma para llegar hasta ese punto, considerando las restricciones espaciales y así evitar colisiones. Este algoritmo busca reducir la distancia entre el punto efector actual y el punto efector objetivo, moviendo cada *joint* del robot paso a paso, considerando las restricciones dadas por la cámara RGB-D. Es un algoritmo iterativo en donde la solución de la trayectoria obtenida no necesariamente es la óptima, sino que es la que permite el movimiento más directo hacia la posición objetivo, lo cual podría hacer que el efector del robot se acerque demasiado a algún objeto y pueda provocar una colisión indeseada.

2.2.5. Algoritmos hápticos basados en *Point Clouds*

Uno de los primeros algoritmos hápticos basados en nubes de puntos (*Point Clouds*) se presentó en el trabajo de N. El-Far et al. [56]. Con este desarrollo ya no es necesario generar superficies poligonales (*polygonal mesh*) a partir de los datos del mundo real, sino que se utiliza directamente la información de posición tridimensional de cada uno de los puntos obtenidos.

N. El-Far et al. [56] introduce el concepto de HIP (*Haptic Interaction Point*) también denominado GO (*God Object*), que corresponde a un punto virtual tridimensional que representa la posición del efector del dispositivo háptico dentro del mundo virtual. La fuerza háptica retroalimentada es un vector calculado a partir de la interacción del HIP con los objetos virtuales del entorno. A través de este elemento HIP se realiza la interacción entre el mundo real del teleoperador, con el mundo virtual que puede ser creado artificialmente o en base al mundo real del robot esclavo que es manejado por el usuario.

El desarrollo propuesto por Rydén et al. [8] [9] utiliza los datos de la *Point Cloud* directamente para realizar el renderizado háptico (o cálculo de fuerza de retroalimentación), utilizando las ideas presentadas en los trabajos de A. Leeper et al. [57] y N. El-Far et al. [56]. El algoritmo háptico presentado por F. Rydén et al. [8] [9] es fundamental para el desarrollo del presente trabajo de tesis, ya que permite calcular la fuerza háptica con la cual se realiza la interacción entre el brazo robótico y el ambiente virtualizado. La teleoperación propuesta en este trabajo de tesis utiliza este algoritmo de renderizado háptico, desarrollando una aplicación con un enfoque novedoso y que puede emplearse en la industria. En el siguiente capítulo se detallará el funcionamiento de este algoritmo, y cómo se relaciona con el sistema de teleoperación propuesto.

2.2.6. Evaluación de interfaces hápticas

Para poder evaluar el funcionamiento de una interfaz humano-máquina de tipo háptica, se deben considerar diversos factores. En primer lugar, las habilidades humanas motrices son consideradas tanto o más importantes que las capacidades técnicas de los sistemas de teleoperación, como muestra S. Hirche et al. [47]. Es por este motivo que los sistemas tecnológicos se deben adaptar a las condiciones de los usuarios de manera que estos se sientan lo más cómodos posibles con los instrumentos de control. El operador es quien cierra el lazo de control e integra una parte fundamental del sistema de teleoperación. En otras palabras, la

percepción humana y su reacción frente a los estímulos están incluidos en el lazo de control (S. Hirche et al. [47]), tomando como desafío hacer lo más natural posible el entorno de operación, para que el usuario sea capaz de reaccionar de manera fluida a los estímulos del medio real en donde se desea teleoperar con el robot.

La investigación de J. Wildenbeest et al. [58] muestra una evaluación de la calidad de la retroalimentación háptica y su importancia para tareas de teleoperación. Se encontró que se aumenta el desempeño en tareas de teleoperación cuando se agrega retroalimentación háptica. Con el uso de una interfaz háptica de baja frecuencia se aumenta el desempeño considerablemente, luego se aumenta progresivamente la frecuencia de retroalimentación, produciendo mejoras contínuas pero marginales. En términos generales, el estudio de J. Wildenbeest et al. [58] demostró que la utilización de retroalimentación háptica mejora el desempeño comparativamente con un sistema sin interfaz háptica, sin importar la frecuencia de manera considerable.

La publicación de V. Nitsch et al. [59] evalúa la efectividad de diferentes formas de interfaces hápticas para tareas de teleoperación. Se realiza una comparación de datos recolectados de diferentes desarrollos, en donde se mide la tasa de error, tiempos de operación, precisión de usuarios, entre otros indicadores, para determinar las mejoras presentadas en sistemas hápticos (retroalimentación vibrotáctil y retroalimentación de fuerza) en comparación con sistemas no hápticos. En general se demuestra nuevamente que las interfaces hápticas ayudan a mejorar los diversos indicadores de desempeño en teleoperación.

2.2.7. Aplicaciones de teleoperación

Z. Ju et al. [60] desarrolló un sistema de teleoperación háptica en robots humanoides, utilizando una arquitectura maestro (*Phantom Omni*) esclavo (*7-DOF Baxter robot arm*). Para lograr el objetivo, se utilizan dos enfoques de control de movimiento: la primera es cinemática directa, y la segunda es cinemática inversa de lazo cerrado o "CLIK" (*closed-loop inverse kinematics*). La fuerza de retroalimentación calculada es proporcional a la diferencia entre la posición real y la posición deseada del brazo robótico esclavo, lo cual corresponde a la ya mencionada *Ley de Hook* o Ley de elasticidad [32].

Diversas aplicaciones de teleoperación háptica se han desarrollado en el área de la medicina. Se usan robots teleoperados para realizar cirugías poco invasivas (A. Talasaz et al. [61]). Se utiliza el enfoque maestro-esclavo, donde la fuerza retroalimentada se obtiene a partir de las mediciones un un sensor tipo galga extensiométrica (*strain gauges*) con la cual se obtienen datos de presión de manera indirecta a través de su deformación física (cuando hay una tensión asociada). Este sensor se ubica en una zona cercana al efector de este tipo de robots, que usualmente cuenta con pequeñas pinzas para realizar con precisión tareas en lugares de pequeño tamaño y difícil acceso. Los resultados de estos desarrollos muestran que usando retroalimentación de fuerza, el cirujano puede controlar el efector final del robot con más precisión y puede realizar esta tarea reduciendo los riesgos asociados a aplicar una presión desmedida.

Una teleoperación háptica de un brazo robótico *LWR KUKA* ha sido desarrollada por I.

Sarakoglou et al. [62], utilizando la combinación de un dispositivo de fuerza háptica *Omega7*, y un novedoso dispositivo de retroalimentación táctil. El dispositivo táctil creado se ubica en la punta del dedo índice del usuario de manera de permitirle sentir de manera remota la rugosidad de la superficie correspondiente a la zona de trabajo del robot. El experimento realizado consiste en explorar y recorrer un camino con relieve (trazado dentro de la zona de trabajo del brazo robótico), usando la punta del efector del robot. Los resultados indican que se incrementa el desempeño en la teleoperación, reduciendo la fuerza de contacto y además minimizando el error en la posición del efector del robot cuando la retroalimentación táctil y de fuerza están activadas.

Una telemanipulación háptica basada en un sistema de control compartido, presentada por H. Boessenkool et al. [63], es utilizada para apretar una tuerca mecánicamente con una llave, analizando la precisión del usuario dado tres diferentes niveles de transparencia. La transparencia, en esta investigación, es el grado de seguimiento de la posición y fuerza desde el dispositivo maestro hacia el esclavo y viceversa, lo cual tiene que ver con la velocidad de respuesta y la proporcionalidad con la cual se obtiene la retroalimentación que será entregada al usuario. H. Boessenkool et al. [63] muestra que cuando el control compartido es utilizado, se registra una mejora en la precisión de la posición, sin importar el nivel de transparencia. La investigación de J. van Oosterhout et al. [49] indica que el control háptico compartido (el cual entrega una fuerza de asistencia al usuario), sólo es beneficioso cuando el sistema no tiene inexactitudes en la posición del efector al momento de entregar una nueva posición objetivo. Es un enfoque interesante para realizar una teleoperación, pero necesita conocer el camino deseado que debe seguir el efector del robot esclavo y determinar sus movimientos ideales de manera previa al momento de realizar la teleoperación.

2.3. Discusión

Se ha revisado el marco teórico sobre el cual se han propuesto diferentes enfoques de teleoperación, mostrando sus fortalezas y al mismo tiempo, verificando su gran potencial para realizar aportes en distintas áreas. El objetivo general es siempre mejorar el desempeño de las tareas de manipulación remota.

Se han implementado diversos métodos para realizar teleoperaciones con retroalimentación háptica, pero ninguno plantea el enfoque que se entrega en esta tesis. Se incorporan elementos de varias investigaciones realizadas hasta el momento, con el fin de potenciar un sistema nuevo de teleoperación que en el futuro puede ser utilizado en diversas aplicaciones industriales. En el siguiente capítulo se detallarán los pasos para la construcción del prototipo de teleoperación con interfaz háptica basado en *Point Clouds* propuesto.

3 Desarrollo de Sistema de Teleoperación Háptico Propuesto

3.1. Descripción general

Se implementó un sistema de teleoperación háptico bilateral basado en la arquitectura maestro-esclavo, utilizando un brazo robótico (esclavo) y un dispositivo háptico *Phantom Omni* (maestro). Este sistema de teleoperación háptico considera al usuario parte fundamental para cerrar el lazo de control entre el mundo real y el mundo virtual generado. En el siguiente esquema se muestran los principales bloques que conforman el proyecto.

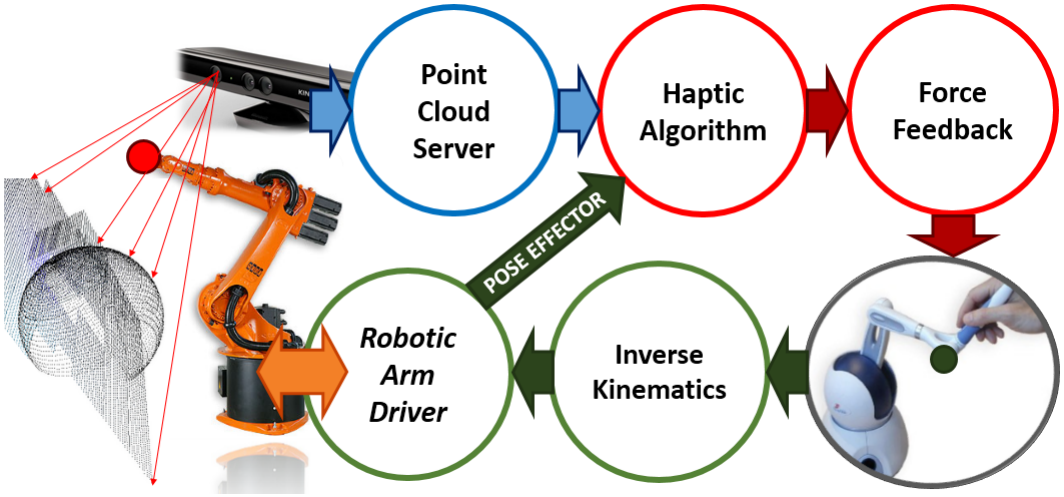

Figura 3.1: Esquema global de sistema de teleoperación háptica desarrollado

En la Figura 3.1 se observa al lado izquierdo el *workspace* del brazo robótico el cual tiene conectado cerca del efector (*link* número 5) una cámara RGB-D *Microsoft XBOX Kinect*. Los datos adquiridos por la cámara, son procesados por el nodo *Point Cloud Server* a partir del cual se calcula la fuerza de retroalimentación en el nodo *Haptic Algorithm*, que además requiere conocer la posición actual del efector del robot. Las instrucciones que se entregan al robot son manejadas por el nodo *Robotic Arm Driver* que recibe el ángulo deseado de cada uno de los *joints* del robot obtenidos a partir del nodo de *Inverse Kinematics*. El usuario, quien controla el dispositivo *Phantom Omni*, se encarga de cerrar el ciclo de control, recibiendo por un lado la fuerza de retroalimentación que proviene del nodo *Force Feedback*

y por el otro lado, envía la posición deseada del efector del robot hacia el nodo que calcula la cinemática inversa.

El hardware de interfaz háptica utilizado es el dispositivo *Phantom Omni* (ver Figura 3.12) de la empresa *Geomagic* [64], el cual tiene 6 grados de libertad o DOF (*Degrees-Of-Freedom*) de movimiento en su efector final (que es un puntero tipo lápiz o *stylus*) y la retroalimentación de fuerza háptica es aplicada en 3 de estos grados.

La posición del efector del dispositivo *Phantom Omni* dada por $P_{eff}(x, y, z)$ en las coordenadas del *workspace* del *Phantom* ($W_{Phantom}$) entrega la posición deseada para el efector final del brazo robótico dado por $K_{eff}(x, y, z)$ dentro del *workspace* del robot (W_{ROBOT}) a través de un escalamiento proporcional lineal.

Sea $P_{eff} \in W_{Phantom}$ entonces:

$$K_{eff} = C \cdot P_{eff} \tag{3.1}$$

con la constante $C \in \mathbb{R}$, y $K_{eff} \in W_{ROBOT}$.

La cinemática inversa del brazo robótico se implementa a través del software *MoveIt* [65] que fue desarrollado para el entorno *ROS*. La posición deseada del efector del robot y las restricciones angulares de cada uno de sus *joints* (que son dadas por el fabricante del brazo), determinan finalmente el ángulo que debe ser alcanzado por cada *joint* para lograr el objetivo de posicionar el efector en la ubicación pedida. Luego, se envía al *ROBOT-PC* la posición deseada de los seis *joints* del robot, y como consecuencia se obtiene la posición del efector requerida. El área de teleoperación válida corresponde a la zona en la cual el algoritmo de cinemática inversa (*inverse kinematics o IK*) tiene solución, que es un subconjunto del área de trabajo disponible, ya que existen restricciones de movimiento propias de la forma en la cual está construido el robot.

La cámara RGB-D *Microsoft XBOX Kinect* se encuentra anclada, con una estructura impresa en 3D, a 30 centímetros del efector del robot, con el objetivo de que la posición del efector permita también cambiar el punto de vista de la cámara, lo cual permite explorar distintos ángulos del ambiente de trabajo en el cual está inserto el brazo robótico.

El esquema general de la teleoperación bilateral propuesta se muestra en la Figura 3.2. A través de la pantalla, el usuario puede observar el ambiente virtual creado con la *Point Cloud*, el brazo robótico simulado llamado *robot-shadow* que representa la posición deseada del robot, y la posición del *robot-real* (ver Figura 3.3). El usuario controla la posición del efector del *robot-shadow*, utilizando el dispositivo *Phantom Omni*. A través del dispositivo háptico, el usuario puede sentir la fuerza de retroalimentación dada por la interacción entre el *end-effector* del *robot-shadow* y la nube de puntos. La fuerza de retroalimentación permite evitar alcanzar posiciones peligrosas para el robot, en donde es probable que se ocasione alguna colisión del robot con el ambiente que lo rodea. Cuando el operador está completamente seguro sobre la posición deseada del robot, entonces puede presionar el primer botón ubicado en el *stylus* del *Phantom Omni*, y el *robot-real* se mueve desde su posición actual hasta la posición donde se encuentra el *robot-shadow*. Esa trayectoria será permitida (y por lo tanto ejecutada) sólo si no existen *voxeles* ocupados en la trayectoria en linea recta entre el *robot-real* (posición origen) y *robot-shadow* (posición destino).

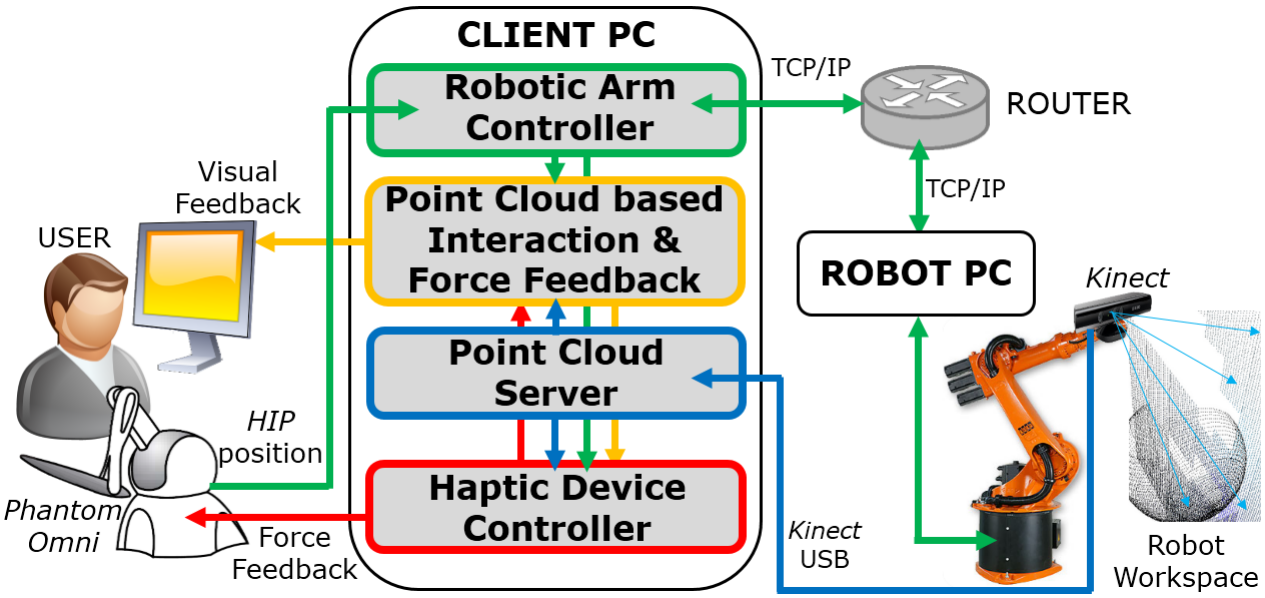

Figura 3.2: Esquema de arquitectura de comunicación. El operador interactúa con el sistema a través del dispositivo *Phantom Omni*. Los nodos de *ROS* corren dentro del *CLIENT-PC*. El nodo *ROBOTIC ARM DRIVER* realiza la conexión con el *ROBOT-PC* a través del protocolo TCP/IP.

3.2. Etapas del sistema de teleoperación

Se puede dividir el sistema de teleoperación háptico implementado en tres etapas o estados:

Primera etapa

La primera etapa es el escaneo del *workspace* del brazo robótico, que consiste en al menos cuatro fotografías tomadas con la cámara RGB-D en diferentes posiciones, lo cual entrega múltiples ángulos de visión de los objetos en el entorno. La posición del efector del robot determina la posición en la cual se toman estas imágenes. Se tienen predefinidas las posiciones de los *joints* con los cuales se pueden obtener ángulos de visión variados y que presentan una gran utilidad al momento de realizar la teleoperación.

Cada una de las imágenes obtenidas con la cámara RGB-D es una *Point Cloud*, las cuales son sumadas, obteniendo una sola gran *Point Cloud* general. En esta etapa, algunos grupos de puntos son redundantes (ya que corresponden a la representación espacial del mismo cuadrante) por lo que se debe reducir su redundancia utilizando *voxel grid* o grilla de voxeles (representación 3D de un pixel) en el nodo de *ROS* denominado *Point Cloud Server*. Por otra parte, se utiliza la estructura *Octree* incluida en *Point Cloud Library (PCL) [66]* para facilitar la búsqueda de puntos específicos dentro de la *Point Cloud* sumada, lo cual permitirá generar la fuerza de retroalimentación háptica.

Segunda etapa

La segunda etapa corresponde al movimiento del robot simulado *robot-shadow* que se representa gráficamente como brazo robótico transparente en el *workspace* virtual a través del software *Rviz* (ver Figura 3.3). La esfera de *proxy* sigue la posición del efector final del *robot-shadow* e interactúa con los puntos de la *Point Cloud* registrada, utilizando el algoritmo que se describirá en la siguiente sección. El robot simulado *robot-shadow* responde a las instrucciones dadas por el dispositivo *Phantom Omni* a través del nodo de *ROS* que calcula la cinemática inversa del brazo robótico. Si el usuario intenta pasar a través de un objeto en el entorno virtual, podrá sentir la fuerza de retroalimentación en el dispositivo de control *Phantom Omni* como resultado del método de renderizado háptico basado en *Point Clouds* (*Proxy Method of Haptic Rendering*).

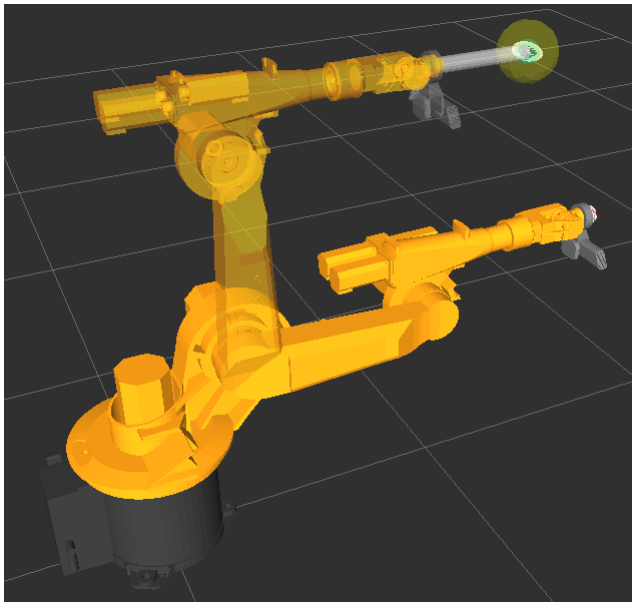

Figura 3.3: El *robot-shadow* es el robot transparente que tiene la esfera amarilla *proxy* en su efector final. El *robot-real* se representa a través del robot de color sólido, y su posición coincide exactamente con la posición del brazo robótico en el mundo real.

Tercera etapa

En la tercera etapa, se define la posición objetivo del brazo robótico real, presionando el botón gris del *stylus* del dispositivo *Phantom Omni*. El movimiento de cada *joint* del robot real es efectuado desde la posición de *robot-real* hacia la posición del *robot-shadow* en el momento que el usuario presiona el botón. Este enfoque de teleoperación *offline* basado en un entorno de realidad virtual, hace más fácil practicar diferentes movimientos antes de efectuarlos en la realidad. Sólo cuando el operador está completamente seguro de la posición final que debe alcanzar el robot, entonces el robot ejecuta la acción. Dado este enfoque de teleoperación, el robot real no necesita seguir la posición del *robot-shadow* en tiempo real (lo que sería un enfoque *online*).

3.3. Método de renderizado háptico

El algoritmo de interacción háptica con la nube de puntos fue desarrollado por Rydén et al. [8] [9]. En esta tesis se implementa y utiliza este método para una teleoperación con retardo temporal. El *proxy* es un objeto virtual creado para interactuar con la *Point Cloud*, el cual está conformado por tres esferas concéntricas de radios R_1, R_2 y R_3. Los radios de las esferas que constituyen al *proxy* cumplen con la siguiente relación:

$$R_1 < R_2 < R_3 \tag{3.2}$$

El *HIP* (*Haptic Interface Point*) es un objeto virtual puntual, que representa la posición del efector final del dispositivo háptico *Phantom Omni* dentro del ambiente simulado. En este caso, el *HIP* se ubica en el efector final del robot simulado *robot-shadow* (ver Figura 3.4). El *workspace* del *robot-shadow* es un poliedro (paralelepípedo) que se construye proporcionalmente a partir del poliedro que determina el *workspace* del dispositivo *Phantom Omni*, considerando una constante de proporcionalidad independiente para cada dimensión espacial (x, y, z).

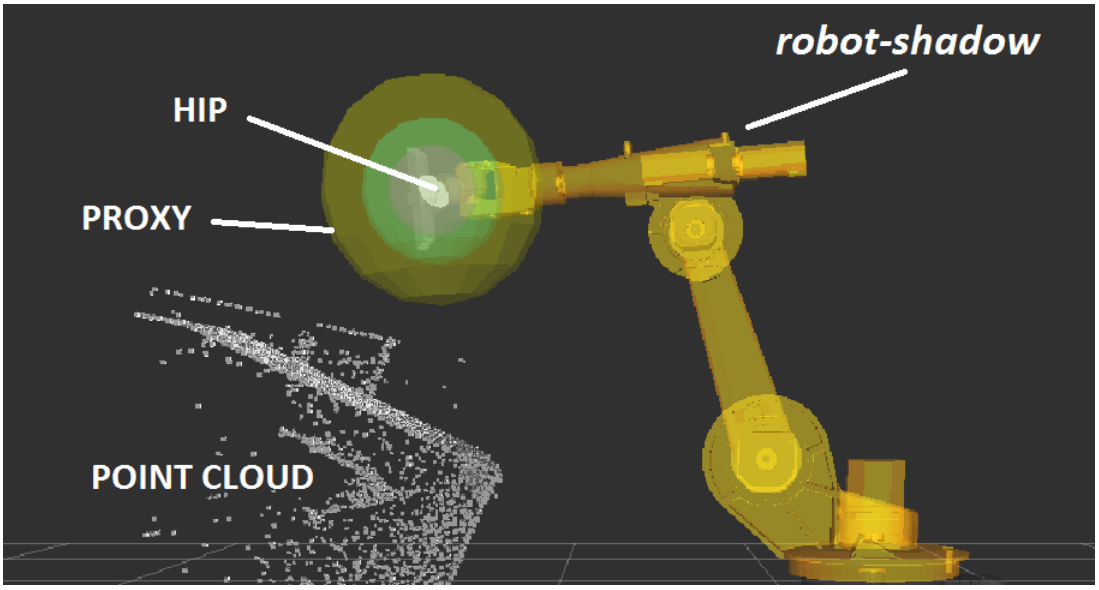

Figura 3.4: Elementos del Método de renderizado háptico.

Los puntos (de la *Point Cloud* grabada en el paso anterior) que están ubicados dentro de la esfera de radio R_3, son usados para calcular el vector normal $\hat{n}$ a la superficie de los objetos que están en el entorno. Sea $P_{PROXY}(k)$ el vector posición del *proxy* en el paso (tiempo discreto) k, $P_{HIP}(k)$ el vector posición del *HIP* en el paso k, ambos en el *workspace* 3D simulado. Sea M el número de puntos de la *Point Cloud* dentro de la esfera de radio R_3. Sea P_{i,R_3} el vector posición del i-ésimo punto de la *Point Cloud* que está dentro de la esfera de radio R_3. Entonces, el vector normal en el paso k está dado por la siguiente ecuación:

$$\hat{n}(k) = \hat{n_k} = \frac{1}{M} \sum_{\mathrm{i}=1}^{M} \frac{P_{PROXY}(k) - P_{\mathrm{i},R_3}}{\|P_{PROXY}(k) - P_{\mathrm{i},R_3}\|} \tag{3.3}$$

Si la esfera de radio R_3 no posee puntos de la *Point Cloud* en su interior, entonces la posición del *proxy* seguirá continuamente a la posición del *HIP*. Si la esfera de radio R_3 tiene puntos dentro, entonces se calcula el vector normal a la superficie con la ecuación 3.3 y se definen los siguientes cuatro casos de "Estados del *proxy*" [8] [9] (ver Figura 3.5):

1. ***Movimiento libre*:** Sea $\hat{u_k}$ el vector unitario desde el centro del *proxy* hasta la posición del *HIP*, el cual representa el movimiento deseado del *proxy*. Si las esferas de radios R_1 y R_2 no tienen puntos de la *Point Cloud* en su interior, entonces el *proxy* sigue la posición del *HIP*, moviendose un paso en la dirección del vector $\hat{u_k}$ (Figura 3.5(a)).

2. ***Contacto y con el HIP fuera de la superficie*:** Este estado ocurre si la esfera de radio R_1 no tiene puntos en su interior, pero la esfera de radio R_2 si los tiene, y el *HIP* está fuera de la superficie. La dirección de movimiento del *proxy* será dada por el vector unitario $\hat{u_k}$ (Figura 3.5(b)).

3. ***Contacto y con el HIP dentro de la superficie*:** Este estado ocurre si la esfera de radio R_1 no tiene puntos en su interior, pero la esfera de radio R_2 si los tiene, y el *HIP* está dentro de la superficie. La dirección de movimiento del *proxy* está dada por el vector $u_{k,p}$ (Figura 3.5(c)), que se define como la proyección de u_k en el plano definido por $\hat{n_k}$.

4. ***Traspaso*:** Si la esfera de radio R_1 tiene puntos de la *Point Cloud* en su interior, el siguiente movimiento del *proxy* tomará la dirección dada por el vector normal a la superficie $\hat{n_k}$ (Figura 3.5(d)).

En base a lo definido para cada "Estado del *proxy*", la posición del *proxy* se calcula iterativamente siguiendo las reglas que se muestran a continuación.

Se define d_k como el parámetro de delta de posición del *proxy*, lo cual representa el paso o cambio de posición mínimo del *proxy* desde su posición actual a la posición objetivo. Este valor debe tener un equilibro en su magnitud, ya que si es demasiado grande el *proxy* no podrá alcanzar con precisión algunas posiciones, y por otra parte si es demasiado pequeño el valor de este parámetro, la posición del *proxy* nunca convergerá.

Las reglas de actualización de la posición del *proxy* son:

- Si el *proxy* está en los estados *Movimiento libre* o en *Contacto y con el HIP fuera de la superficie* entonces:

$$P_{PROXY}(k+1) = P_{PROXY}(k) - \mathrm{d}_k \hat{u_k} \tag{3.4}$$

- Si el *proxy* está en el estado *Contacto y con el HIP dentro de la superficie* entonces:

$$P_{PROXY}(k+1) = P_{PROXY}(k) - \mathrm{d}_k \hat{u_{k,p}} \tag{3.5}$$

- Si el *proxy* está en el estado *Traspaso* entonces:

$$P_{PROXY}(k+1) = P_{PROXY}(k) - \mathrm{d}_k \hat{n_k} \tag{3.6}$$

La aplicación de las reglas de actualización de posición del *proxy* deben ser iteradas hasta que la posición del *proxy* converja. Los criterios de convergencia son dos, pero es necesario asegurar el cumplimiento de al menos uno de estos criterios:

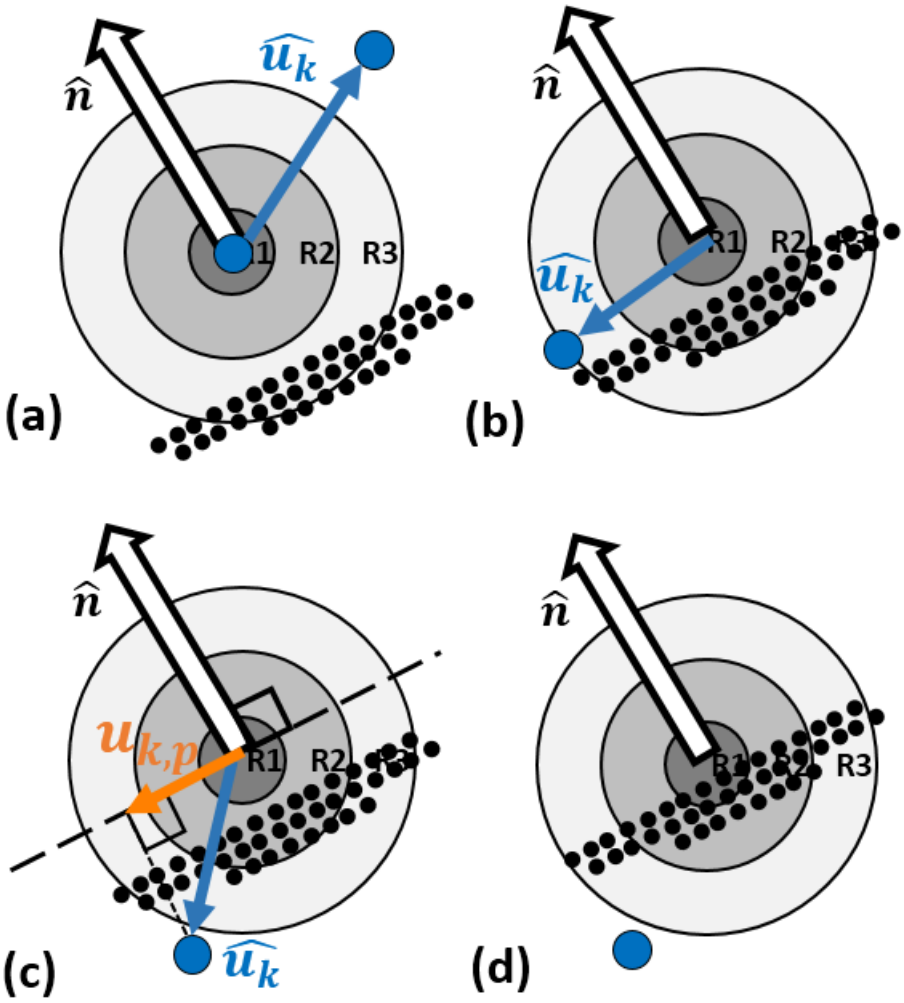

Figura 3.5: Estados del *proxy*: (a) Movimiento libre, (b) Contacto con el *HIP* fuera de la superficie, (c) Contacto con el *HIP* dentro de la superficie, y (d) Traspaso.

- Criterio de convergencia de posición:

$$\|P_{PROXY} - P_{HIP}\| < \varepsilon \tag{3.7}$$

  donde ε es el umbral de convergencia de posición.
- Criterio de convergencia angular (este criterio es el que prevalece cuando el *proxy* está en *Contacto*:

$$\arccos \frac{-u_k^{\intercal} \hat{n}_k}{\|u_k\|} < \theta \tag{3.8}$$

  donde θ es el umbral de convergencia angular.

3.4. Fuerza de retroalimentación háptica

3.4.1. Cálculo de Fuerza

Cuando la posición del *proxy* converge, entonces se puede calcular la fuerza de retroalimentación (ver Figura 3.6) usando la *Ley de Hook* o Ley de elasticidad [32] entre la posición del *proxy* y la posición del *HIP* (F. Rydén et al. [8] [9]).

Sea K_{spring} la constante elástica del resorte virtual que une al *proxy* y al *HIP*. La fuerza retroalimentada (F_{HIP}) al dispositivo háptico *Phantom Omni* es directamente proporcional a la distancia que separa el *HIP* y el punto medio entre la esfera de radio R_1 y la esfera de radio R_2 del *proxy*, tal como se muestra Ecuación 3.9.

$$F_{HIP} = -\frac{u_k}{\|u_k\|} K_{spring} \left(\|u_k\| - \frac{R_1 + R_2}{2} \right) \tag{3.9}$$

donde:

$$u_k = u(k) = P_{PROXY}(k) - P_{HIP}(k) \tag{3.10}$$

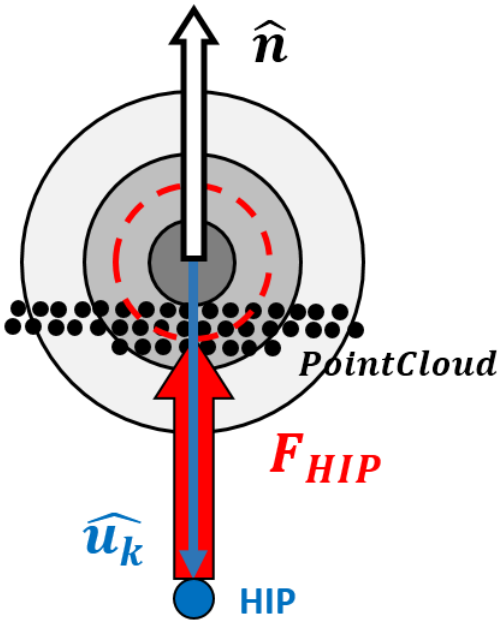

Figura 3.6: Diagrama de fuerza de retroalimentación háptica. Se muestra el *HIP*, las 3 esferas que conforman el *proxy*, la nube de puntos, el vector normal a la superficie y la fuerza de retroalimentación háptica F_{HIP} calculada.

El *HIP* no interactúa con la *Point Cloud*, pero el *proxy* si puede cambiar su posición en función de los puntos dados por los obstáculos del entorno. Esta arquitectura, presentada

por F. Rydén et al. [8] [9] entrega la retroalimentación necesaria al usuario, para que pueda sentir la fuerza elástica sobre el *stylus* del dispositivo *Phantom Omni*, simulando una fuerza de contacto real con el objeto virtualizado que es una representación del mismo objeto en el mundo real.

3.5. Ciclo general de algoritmo háptico

A continuación se muestra una visión general del algoritmo para calcular la fuerza de retroalimentación háptica (ver Figura 3.7). Primero se obtiene el "Estado del *proxy*", luego se calcula vector normal a la superficie, después se actualiza la posición del *proxy* en el *workspace* virtualizado y finalmente se calcula la fuerza de retroalimentación. El ciclo es cerrado por el usuario del sistema, quien recibe la fuerza de retroalimentación en su mano a través del dispositivo *Phantom Omni*, y toma una acción de acuerdo a la fuerza que puede sentir, moviendo el *HIP* de posición e iniciando un nuevo ciclo háptico.

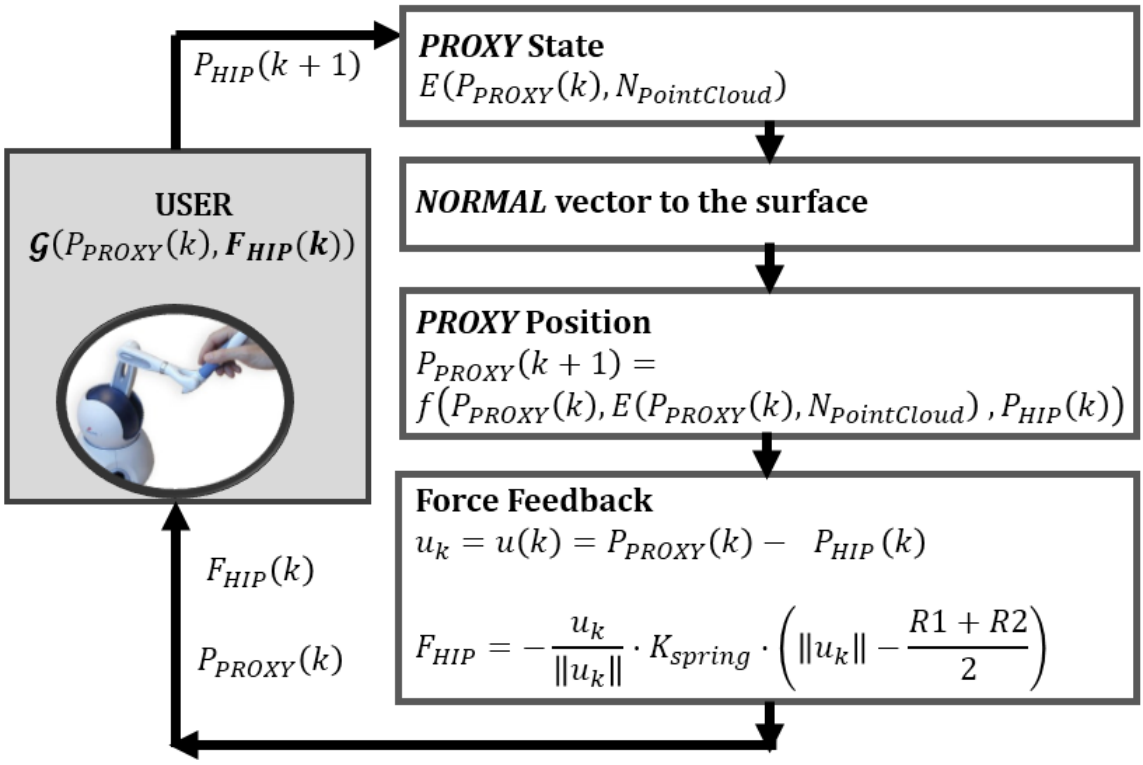

Figura 3.7: Ciclo general del algoritmo háptico.

La tasa mínima de frecuencia a la cual se debe ejecutar el algoritmo de renderizado háptico debe ser de $1[KHz]$, lo cual implica que la fuerza háptica calculada debe ser actualizada con esa frecuencia. La tasa de $1[KHz]$ es recomendada por Basdogan et al. [67] porque así es posible obtener una fuerza suave de renderizado háptico, considerando que la tasa de actualización visual es usualmente de $30[Hz]$. Esta diferencia de frecuencias en la actualización sensorial, está dada por la forma en la cual nuestros sentidos obtienen información del entorno, en donde claramente es posible distinguir diferencias hápticas a frecuencias más altas

(cercanas a $1[KHz]$), que hacer lo mismo con nuestro sentido visual, el cual sobre $30[Hz]$ muchas veces no es capaz de distinguir cambios o mejoras.

3.6. Brazo robótico

Se desarrolla el nodo de *ROS* denominado *ROBOTIC ARM DRIVER* el cual utiliza servicios para realizar las peticiones al robot, y es ejecutado en el *CLIENT-PC* (ver Figura 3.2). Se implementa una cola de prioridad (*priority queue*) que funciona como *buffer* de las instrucciones que se envían al robot, asegurando que se respete un orden adecuado según la importancia de estos comandos antes de ser enviados al robot.

A continuación se muestra el listado de peticiones disponibles para el brazo robótico, usando el nodo *ROBOTIC ARM DRIVER*:

1. La petición *Stop* tiene la prioridad más alta, la cual se encarga de bloquear todos los movimientos del robot.

2. La petición *PTP* o *Point to Point* permite definir el ángulo deseado de cada uno de los *joints* del robot. El movimiento por defecto implementado en el brazo robótico, hace que al utilizar la petición *PTP* los *joints* posean una velocidad variable (que se ajusta automáticamente) de tal forma de asegurar todos los *joints* lleguen al mismo tiempo a su posición objetivo establecida.

3. La petición *Set velocity* permite cambiar la velocidad de movimiento máxima de los *joints* del robot desde 0 % (no hay movimiento) hasta 100 % (que representa la velocidad límite que puede entregar el brazo robótico).

La limitación agregada al *ROBOTIC ARM DRIVER* sólo permite una comunicación de una vía, por lo tanto se tiene que esperar hasta que la cola de prioridad este vacía, antes de realizar una petición de estado actual de robot. Esta restricción en la comunicación hace que este enfoque sea similar a las tareas de manipulación en órbita, donde el tiempo de retraso en la comunicación es variable, y usualmente va desde 1 a 4 segundos (T. B. Sheridan et al. [35]).

3.6.1. Simulación del robot en *Gazebo*

A medida que el software de control del brazo robótico se desarrolla, es necesario probar su funcionamiento, para lo cual se utiliza el motor físico *Gazebo*. Una vez que el programa funciona correctamente en el simulador físico, se prueba sobre el robot real, detectando tempranamente cualquier error de programación que pueda ocasionar un accidente al momento de operar el robot real.

Una de las ventajas de utilizar la plataforma *Gazebo*, es que proporciona comunicación a través de mensajes de *ROS*, simulando no sólo la física del robot y del entorno de operación,

sino que también los sensores que se le agregan al robot y sus respectivos mensajes. En el caso del presente proyecto, se agrega una cámara RGB-D *Kinect* virtual en la punta del efector del robot simulado en *Gazebo*.

La Figura 3.8 (lado izquierdo) muestra el brazo robótico en un entorno de operación virtual compuesto por un bloque de ladrillo ahuecado sobre un cilindro de aproximadamente 1,30$[m]$ de altura. La Figura 3.8 (lado derecho) muestra la visualización de los datos de posición del robot entregados por el mensaje de *Joint States* generado por el robot virtual, y los datos de *Point Cloud* entregados por el *Kinect* virtual, luego de ejecutar una acción de *Scan* del *workspace* virtual. Es así como el programa visualizador *Rviz* no distingue el origen de los mensajes (tópicos) *Joint States* o *Point Cloud* a los cuales está suscrito, mostrando la posición del brazo robótico y el entorno detectado como si la información proviniera del brazo robótico real. Este enfoque utilizado para desarrollar el software de control de movimiento del robot se denomina simulación *Hardware-in-the-loop* o HIL, el cual es ampliamente usado en sistemas complejos de ingeniería, donde se necesita testear nuevos bloques desarrollados del proyecto, sin tener que probarlos directamente sobre la planta o dispositivo real, ya que implica un riesgo mayor o un coste monetario alto. El simulador HIL responde a los estímulos físicos (entradas y salidas) de la misma manera que lo hace el dispositivo o planta real, como es el caso que muestra el brazo robótico en *Gazebo*.

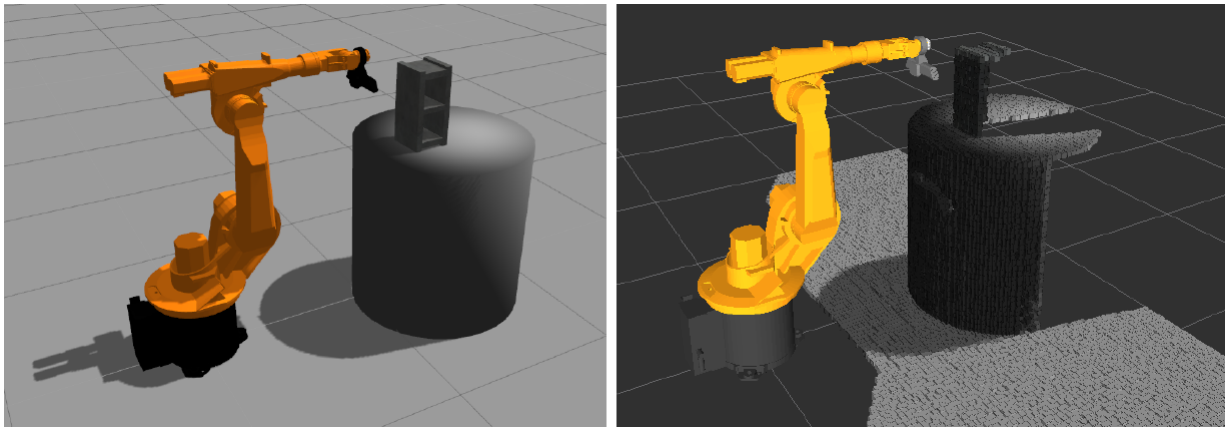

Figura 3.8: Izq: Brazo robótico simulado en motor físico *Gazebo*. Der: Brazo robótico simulado en visualizador *Rviz*.

Otra ventaja que se destaca al utilizar el programa *Gazebo* es su versatilidad y facilidad a la hora de crear un *workspace* simulado nuevo y exponer al algoritmo de detección de obstáculos a nuevos desafíos sin la necesidad de tener que construir físicamente estos objetos. *Gazebo* tiene una interfaz de usuario muy amigable, en la cual sólo se necesitan un par de clics para agregar al entorno objetos comunes como una lata de bebida, un bowl, ladrillos, cubos, cilindros, esferas, etc. Además, se pueden cargar en el entorno modelos 3D más complejos como automóviles, muebles, casas, edificios, etc.

3.7. Servidor de *Point Clouds*

El nodo denominado *Point Cloud Server* se desarrolla en base a la *Point Cloud Library* (PCL) [66]. Este nodo tiene tres funciones principales: Generar un ambiente virtual a través de una *Point Cloud*, editar o borrar el ambiente generado con la *Point Cloud*, y revisar si hay puntos de la *Point Cloud* entre la posición deseada del efector del brazo robótico y la posición del efector del robot real, para poder de esta manera advertir que el brazo está en ruta de colisión.

3.7.1. Entorno virtual

El entorno virtual generado a partir del ambiente real de operación del brazo robótico, se crea con los siguientes pasos:

1. *Captura de imagen RGB-D (Point Cloud)*: Se calcula la posición relativa de la cámara *Kinect* con respecto al *frame* global del entorno, a partir de la posición de los *joints* del robot. De esta manera la imagen o nube de puntos obtenida se puede posicionar en el entorno virtual, en la ubicación que corresponde, fijándola con respecto al marco de referencia total del *workspace* sin importar la posición del brazo robótico.

2. *Agregar una nueva Point Cloud a la Point Cloud global acumulada*: La *Point Cloud* acumulada contiene la fusión de varias *Point Clouds* tomadas desde diferentes puntos de vista. Esto se consigue ya que la cámara está cercana al efector del robot y por lo tanto, puede alcanzar distintos ángulos de visión. Cada nueva imagen 3D se referencia con respecto al *frame* global y luego se suma a la nube de puntos general. El numero de puntos totales aumenta rápidamente con cada fotografía 3D agregada, por lo tanto, es necesario que en el siguiente paso se pueda hacer que los datos de esta nube de puntos sean más manejables. Si se considera que una nube de puntos puede tener al menos un millón de puntos, esto se puede transformar en un problema a la hora de sumar *Point Clouds* ya que la memoria RAM del computador debe tener esta información de manera accesible en cada momento. Trabajar con una *Point Cloud* gigantesca se vuelve inmanejable incluso para un computador de gama media-alta, por lo tanto, se debe tener presente esta restricción de recursos a la hora del diseño del algoritmo.

3. *Se reduce la Point Cloud acumulada a través de "Voxel Grid"*: El *voxel* es el elemento cúbico mínimo en una matriz 3D que representa a una imagen RGB-D. El tamaño del *voxel* se puede definir arbitrariamente, dependiendo de la resolución que se quiere alcanzar. Si se quieren obtener detalles mejor definidos, el tamaño del *voxel* debe ser pequeño, pero al mismo tiempo se dificulta el manejo de datos de la *Point Cloud* que podría contener demasiados puntos. Si dos o más puntos de la *Point Cloud* están dentro del mismo *voxel*, entonces esos datos se reducen a un sólo punto que representa el centro del *voxel*. Este método permite que la reducir la redundancia de puntos en la nube de puntos global, y hacer que el algoritmo de renderizado háptico pueda funcionar más rápidamente. Esta redundancia de puntos de genera cuando se suman dos o más *Point Clouds* y algunas zonas de las imágenes RGB-D se solapan o sobre-posicionan, lo que genera que múltiples

puntos representen al mismo lugar del entorno real.

Por otra parte, se definió una *Point Cloud* fina (de mayor densidad de puntos, o *voxel* más pequeño) que es utilizada por el algoritmo de retroalimentación háptico, y una *Point Cloud* gruesa (de menor densidad de puntos, o *voxel* más grande) que se utiliza en *Rviz* sólo para ser visualizada y para que el programa no se torne lento al cargar una cantidad de datos innecesariamente grande. En la Figura 3.9 se observa el efecto de la reducción de densidad de puntos de la nube, utilizando un *Voxel Grid* y además, se reduce la extensión de la *Point Cloud*, limitando su rango de alcance al área de *workspace* del robot ($2 \times 2[m]$).

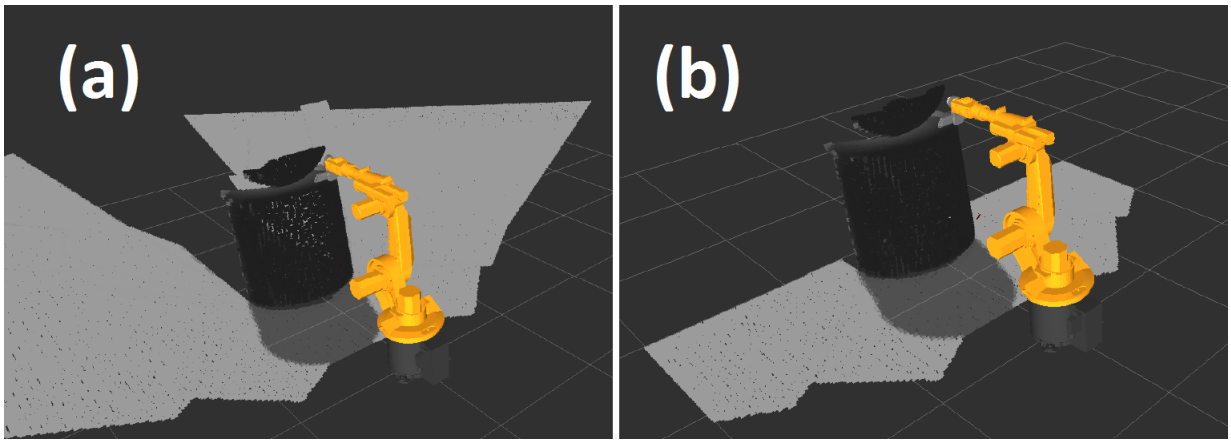

Figura 3.9: (a) *Point Cloud* sin filtrar. (b) *Point Cloud* con filtro usando *Voxel Grid* (menor densidad de puntos) y reducción de espacio de trabajo (menor extensión).

Los tres pasos explicados son iterados cada vez que se requiere capturar una nueva foto 3D a partir de un nuevo punto de vista. Como la cámara *Microsoft XBOX Kinect* está ubicada cerca del efector del brazo robótico, entonces se pueden obtener imágenes del entorno desde distintos ángulos, simplemente actualizando la posición del efector del robot.

3.7.2. Ruta de colisión

Este nodo revisa si los *voxeles* que están en línea recta entre la posición deseada del efector y la posición real del efector del robot están ocupados o libres. Un *voxel* está ocupado, si al menos existe un punto de la *Point Cloud* global que está dentro de él. En el caso de que al menos un *voxel* ocupado sea intersectado por una línea recta que representa la trayectoria de movimiento del brazo, entonces aquella posición objetivo se transforma en una posición prohibida, y por lo tanto se niega el movimiento aún cuando el usuario lo requiera.

La librería *octree* de *PCL* contiene una función llamada *RayCast* la cual genera una línea recta desde un punto específico con una cierta dirección. Esta función entrega el centro de todos los *voxeles* ocupados. El resultados es una lista de espacios ocupados entre el punto de partida y el punto de llegada del robot, lo cual indica que esa trayectoria (en línea recta) no es posible ya que se puede producir una colisión.

Por lo tanto se tiene un control de prevención de colisiones del brazo robótico con el

entorno real, en base a los datos virtuales obtenidos en los pasos descritos anteriormente. En las Figuras 3.10 y 3.11 se observa la detección de trayectoria de colisión.

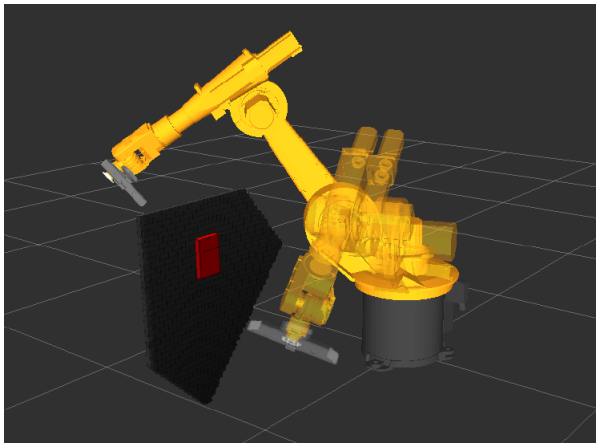

Figura 3.10: Detección de *voxel* ocupado. Ruta de colisión detectada entre posición actual y posición deseada del robot. Un marcador rojo destaca el o los *voxeles* ocupados por los cuales se traza la trayectoria.

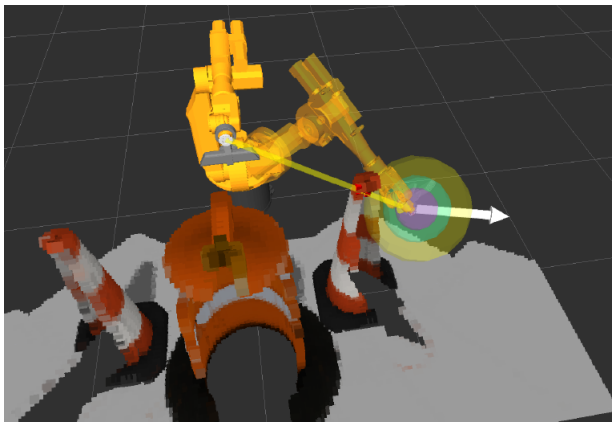

Figura 3.11: Detección de *voxel* ocupado. Ruta de colisión detectada entre posición actual y posición deseada del robot. Un marcador amarillo destaca la trayectoria de colisión.

3.8. Calibración de posición de la cámara

La etapa de calibración de la cámara es fundamental, ya que debe asegurar una correcta correlación entre el modelo virtual del robot simulado (*virtual workspace*) y el entorno de trabajo real del robot. La cámara *Kinect* se ubica a 30$[cm]$ de la punta del efector del brazo robótico, con una inclinación de 90° con respecto al efector.

Realizar un ajuste angular de la cámara en el robot, implica un trabajo de precisión mecánico que podría tardar horas en caso de que se requiera un ajuste muy fino. Es por este motivo que el ajuste angular y de posición de la cámara (con respecto al brazo robótico) se realiza en el modelo del robot simulado. Al mover la cámara en el robot simulado (a través de

un *frame* de referencia dinámico en *Rviz*), se produce también el movimiento de los puntos de la *Point Cloud*, ya que estos se guardan con respecto al punto de captura (en este caso la cámara RGB-D). Conociendo algunos puntos de calibración fijos (*Calib 1, Calib 2, Calib 3 y Calib 4*) en el panel de pruebas, el proceso de calibración es el siguiente: Primero se mueve el efector del *robot-real* a esos puntos fijos hasta hacer contacto con ellos (de a uno a la vez), luego se obtiene una imágen 3D, para finalmente hacer coincidir la *Point Cloud* que representa al panel virtual con la posición del panel real en el *workspace* del brazo robótico real usando el *frame* dinámico. Este proceso se realiza manualmente cada vez que se desajusta la posición de la cámara *Kinect*, tomando pocos minutos realizar la calibración completa de posición y ángulo.

Si la calibración de la posición de la cámara no es lo suficientemente buena, entonces el error entre la posición del efector del robot y los objetos en el entorno de trabajo aumenta y por lo tanto se podrían ocasionar colisiones.

3.9. Controlador *Phantom Omni*

El dispositivo *Phantom Omni* (ver Figura 3.12) que es usado como controlador maestro en esta arquitectura de teleoperación, tiene 6DOF (grados de libertad) tal como el robot esclavo. En los métodos típicos de control en un brazo robótico, se utiliza su cinemática directa o inversa, dependiendo del objetivo que se desea lograr.

Figura 3.12: Dispositivo de interfaz háptica *Phantom Omni* de la empresa *Geomagic*. Imagen obtenida de [64]

Usando cinemática directa (*Forward Kinematics*) el usuario puede manipular la posición (ángulo) de cada *joint* del robot, modificando los ángulos de los *joints* del dispositivo *Phantom*. Es fácil de implementar ya que basta con relacionar cada *joint* del dispositivo Maestro, con su homólogo *joint* en el dispositivo esclavo, y asignar de manera proporcional los rangos de magnitud angular de desplazamiento de cada uno de ellos. A pesar de que es fácil de implementar, no es intuitivo de utilizar para un operario poco entrenado, ya que en este caso, el dispositivo maestro no cuenta con exactamente la misma ubicación de los *joints* comparado

con el robot esclavo. Para que este enfoque pueda ser utilizado con éxito se debe tener un dispositivo de control maestro que sea una copia (a escala) del dispositivo esclavo, y así poder definir el mismo tipo de movimiento para cada *joint* de manera proporcional.

Para este proyecto, se utiliza la cinemática inversa (*Inverse Kinematics*) del robot, a través del software *MoveIt* [65] en *ROS*. La cinemática inversa es más intuitiva para esta aplicación de teleoperación en donde se debe interactuar con los objetos del entorno de trabajo a través del efector final del brazo robótico. Cada punto muestreado en el *workspace* real tiene una representación virtual en el visualizador *Rviz*. El dispositivo háptico maestro puede interactuar con los puntos de la representación virtual, a través del algoritmo de renderización háptica explicado anteriormente. Se tiene entonces una representación a escala en el espacio de trabajo del dispositivo maestro, que corresponde proporcionalmente al espacio de trabajo del dispositivo esclavo (ver Figura 3.13). En resumen, se prefiere la operación a través de cinemática inversa ya que la interacción que presenta mayor valor (para esta aplicación en particular) entre los dispositivos maestro y esclavo, es a través del efector de cada uno de ellos, sin importar cómo se realizan los movimientos de los *joints* para poder alcanzar la posición objetivo.

Se desarrollan dos tipos de teleoperación con cinemática inversa, los cuales se denominan "*Workspace* proporcional" y "*Deltas* de movimientos del efector". A continuación se especifican las diferencias entre estos métodos:

3.9.1. *Workspace* proporcional

En la teleoperación con "*Workspace* proporcional" la posición del efector del dispositivo *Phantom Omni* $P_{eff}(x, y, z)$ (ubicado en la punta del *stylus*) en el *Phantom workspace*, representa la posición deseada del efector del brazo robótico $K_{eff}(x, y, z)$ proporcionalmente en el *robot workspace*. Las constantes de proporcionalidad son distintas para cada dimensión, y están determinadas por los límites de ambos espacios de trabajo. Los *workspaces* del *Phantom* y del brazo robótico se imponen de cuerpo geométrico paralelepípedo por facilidad para realizar la relación de proporcionalidad en las tres dimensiones, con lados (H_X, H_Y, H_Z) y (K_X, K_Y, K_Z) respectivamente (ver Figura 3.13). La zona de operación del brazo robótico debe asegurar que su cinemática inversa tenga solución dentro de ella, lo cual limita bastante el volumen en el cual puede desplazarse su efector. La zona de operación del *Phantom* debe asegurar que el usuario pueda alcanzar todas las posiciones posibles que el dispositivo permita, con el movimiento de la mano, tomando el *stylus* del *Phantom* como si fuera un lápiz.

Una vez que se especifican las zonas de operación del lado Maestro y el lado Esclavo (mostradas en el siguiente Capítulo), se obtienen las constantes de proporcionalidad entre ambos *workspaces*. Entonces se tiene que las constantes de proporcionalidad de los espacios de trabajo para cada dimensión son:

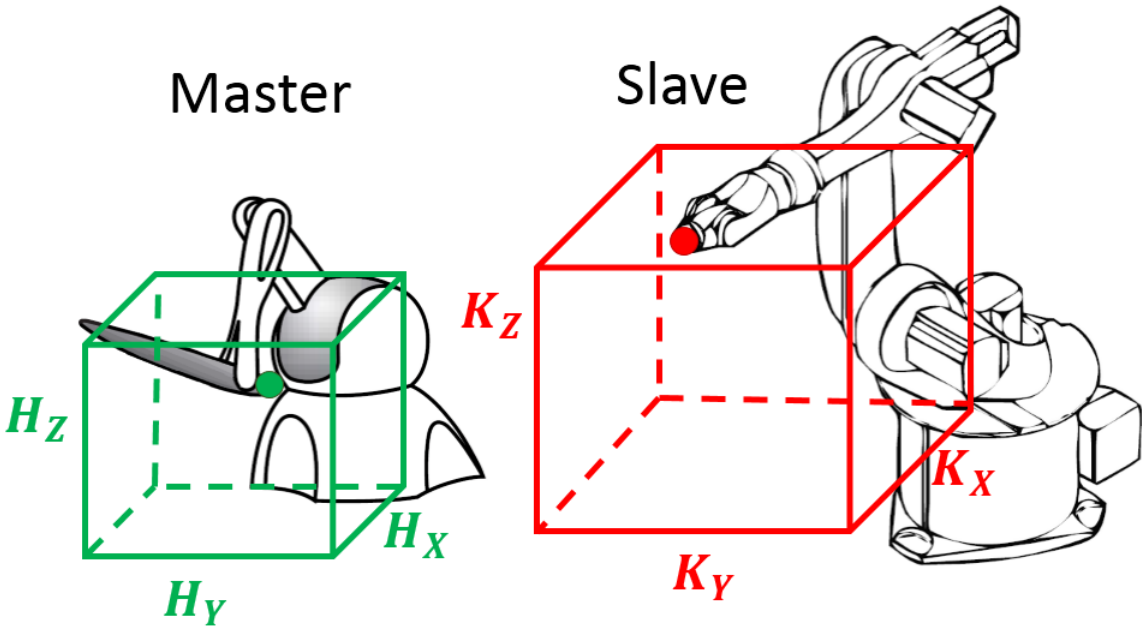

Figura 3.13: Arquitectura de control Maestro (*Phantom Omni*) - Esclavo (Brazo robótico), y la representación de sus respectivos *workspaces* de tamaños proporcionales.

$$\begin{aligned} C_X &= K_X/H_X \\ C_Y &= K_Y/H_Y \\ C_Z &= K_Z/H_Z \end{aligned} \tag{3.11}$$

Estas constantes se mantienen fijas durante toda la teleoperación y permitirán transformar la posición del efector del *Phantom* desde su espacio de trabajo propio, hacia el espacio de trabajo del brazo robótico. Entonces se tiene:

$$\begin{aligned} K_{eff}(x) &= C_X \cdot P_{eff}(x) \\ K_{eff}(y) &= C_Y \cdot P_{eff}(y) \\ K_{eff}(z) &= C_Z \cdot P_{eff}(z) \end{aligned} \tag{3.12}$$

3.9.2. *Deltas* de movimientos del efector

Con este sistema de control para la teleoperación, el *workspace* del *Phantom Omni* es segmentado en tres zonas por cada eje de operación (x, y, z), obteniendo un total de 27 áreas (paralelepípedos) como se muestra en la Figura 3.14. La región central representa la zona de reposo, lo cual significa que la velocidad de movimiento (*Delta* de posición) del efector del esclavo es cero. Si el efector del *Phantom* se desplaza en uno de los ejes, saliendo de la zona de reposo, entonces el efector del brazo robótico se moverá en la misma dirección con una velocidad proporcional a la distancia entre el centro del *workspace* del *Phantom* y su efector. Entre más alejado esté el efector del *Phantom* del centro de su espacio de trabajo, con más

velocidad se desplazará el efector del robot en esa dirección. Se define una velocidad máxima, la cual se obtiene cuando el efector del *Phantom* alcanza su posición más lejana en ese eje, y se define una velocidad mínima, cuando este efector pasa justo desde la zona de reposo a la zona de movimiento.

En este caso el dispositivo Maestro no controla la posición efectiva del dispositivo esclavo, sino que maneja la velocidad y dirección de desplazamiento del efector del robot. Una combinación de movimientos en los tres ejes en el *workspace* del *Phantom*, proporcionará una combinación de velocidades de desplazamiento del efector del brazo robótico en cada uno de esos ejes en forma independiente.

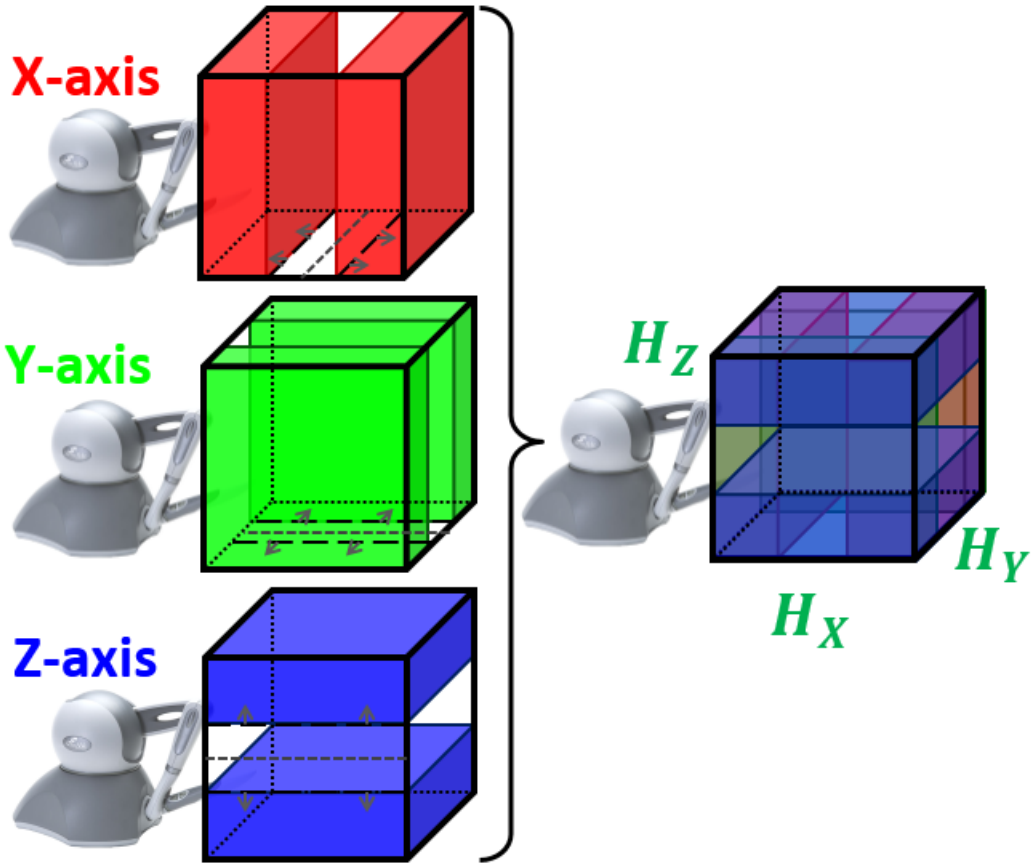

Figura 3.14: Teleoperación *Phantom Omni* por zonas. Se muestran las 27 zonas de operación del *Phantom* con las que se pueden controlar de manera independiente las tres dimensiones espaciales.

La ventaja del sistema de "*Workspace* proporcional" sobre el sistema de "*Deltas* de movimientos del efector" es la sensación más natural de la fuerza de retroalimentación, ya que esta es entregada con mayor rapidez y la respuesta del usuario hace que el efector del *Phantom* se retire de la zona de contacto con mayor velocidad. Por otra parte, el sistema de "*Deltas* de movimientos del efector" permite movimientos más precisos del efector del dispositivo esclavo, y por lo tanto se puede alcanzar una teleoperación de mayor precisión, aunque con menor velocidad de reacción frente a una colisión. En este caso se ve restringida la velocidad de reacción ya que el efector del dispositivo Maestro tiene que posicionarse en una zona de dirección contraria a la acción anterior y esperar que el efector del Esclavo cambie de direc-

ción y salga de la zona de colisión para finalmente dejar de sentir la fuerza de repulsión del objeto.

Ambos métodos se ponen a prueba en la siguiente sección, comparando los resultados obtenidos.

3.10. Latencia Bilateral

La latencia desde el dispositivo Maestro al Esclavo está dada por la latencia propia de la arquitectura servidor-cliente que utiliza el protocolo TCP/IP, y por la cola de prioridad de mensajes y peticiones diseñada para asegurar que los mensajes que poseen mayor importancia sean entregados al robot antes que el resto de los mensajes. En total existe un retardo temporal de 100[ms] a 900[ms] entre el emisor y el receptor, lo cual es un tiempo considerable para aplicaciones de teleoperación, pero se asegura que los mensajes sean recibidos de manera correcta (*error-checked delivery*).

La latencia desde el dispositivo Esclavo al Maestro está determinada principalmente por la captura de la imagen de profundidad (RGB-D) y el procesamiento de estos datos en el nodo de *Point Cloud Server* con el cual se construye el entorno virtual. Por lo tanto, el retardo temporal se obtiene en función de la resolución del *voxel*. Si la grilla de *voxeles* es más densa (*voxel* más pequeño), entonces el retardo temporal aumenta porque el algoritmo de renderizado háptico tiene que procesar una mayor cantidad de datos para obtener la fuerza de retroalimentación correspondiente (ver Figura 3.15a).

3.11. Precisión del Sistema

Los parámetros del sistema de teleoperación deben ser ajustados con el objetivo de reducir el error en la posición del *proxy* en el entorno virtual. Estos parámetros son escalables y fáciles de cambiar en cada ejecución del programa, con el objeto de que los usuarios pueden crear un nuevo *set* de parámetros cuando la tarea asignada cambia o cuando los obstáculos en el espacio de trabajo presentan un cambio en su tamaño. Es requisito fundamental conocer *a priori* el tamaño de los obstáculos en el *workspace* del robot real para optimizar el desempeño de esta metodología. Cualquier objeto más pequeño que el tamaño de un *voxel* no será renderizado adecuadamente, por lo tanto, la adecuada configuración de parámetros es un paso importante para este procedimiento.

La Figura 3.15 se obtiene con varias configuraciones de parámetros, y las pruebas se realizan sobre el mismo panel de pruebas (entorno estático). En la Figura 3.15a se muestra el *trade-off* entre el tamaño de *voxel* y la latencia. La configuración ideal es obtener la mayor densidad de *voxeles* posible (tamaño de *voxel* más pequeño) evitando un incremento excesivo de la latencia. Por otra parte, un incremento en el tamaño de los *voxeles* (densidad menor) produce un aumento en el error de posición del *proxy* con respecto al entorno virtual. En la Figura 3.15b se muestra la importancia del tamaño del paso de actualización de posición del

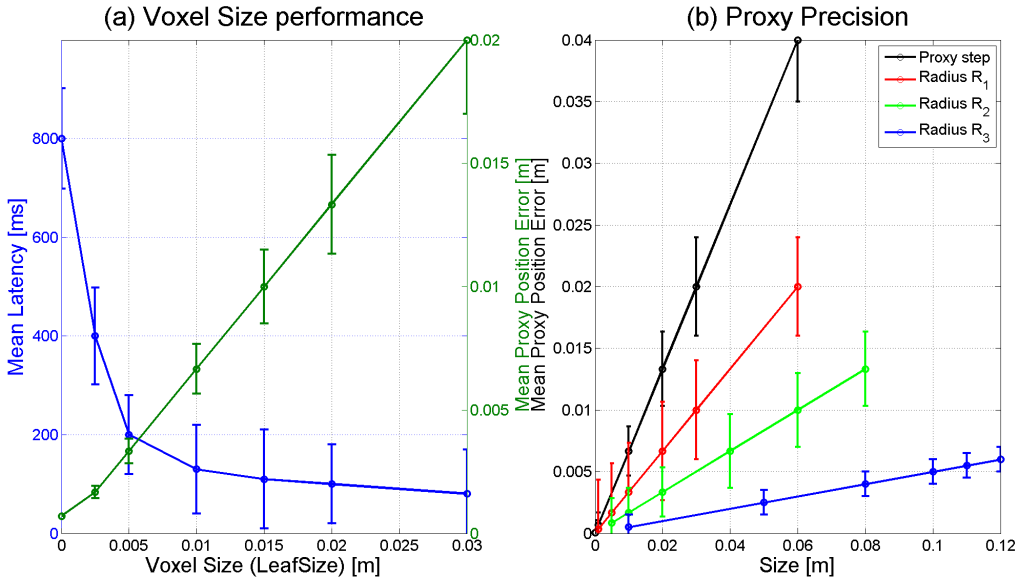

Figura 3.15: (a) Tamaño de *voxel* vs latencia media y vs error promedio de posición del *proxy*. (b) Tamaño de parámetros del *proxy* vs error promedio de posición del *proxy*.

proxy (*proxy step*) y el tamaño de los radios del *proxy* al momento de obtener el error medio de su posición con respeto a los objetos del *workspace* virtual.

Cada parámetro utilizado en la sección de Validación Experimental (Tabla 4.2) garantiza un error de posición promedio del *proxy* de 0,005$[m]$ como máximo (aproximadamente).

3.12. Marcadores de visualización en *Rviz*

Para hacer más fácil la visualización de las etapas del algoritmo explicado en las secciones anteriores, se incorporan en el programa *Rviz* una serie de marcadores, que son elementos gráficos dinámicos presentes en el visualizador. Estos marcadores permiten detectar errores en el algoritmo cuando se está realizando el proceso de desarrollo. Además son una guía visual para el operador, facilitando la realización de la tarea asignada.

En la Figura 3.16 se observa el marcador correspondiente al *HIP* y al *proxy* separados. El *proxy* queda "atrapado" por la *Point Cloud* siguiendo las reglas explicadas anteriormente, mientras que el *HIP* se desplaza libremente por el espacio virtual. La Fuerza es visualizada a través de un marcador tipo vector de color rojo, a partir del *HIP* y en dirección al *proxy*. El vector normal a la superficie de la nube de puntos se representa por un marcador tipo vector de color blanco. Además, se observa que el *proxy* está compuesto por marcadores esféricos concéntricos, con radios R_1 (azul), R_2 (verde) y R_3 (amarillo). En la Figura 3.17 se observan los mismos elementos que en la Figura 3.16, pero además incluye el modelo del robot virtual, y la posición del *HIP* está determinada por la posición del efector del robot virtual.

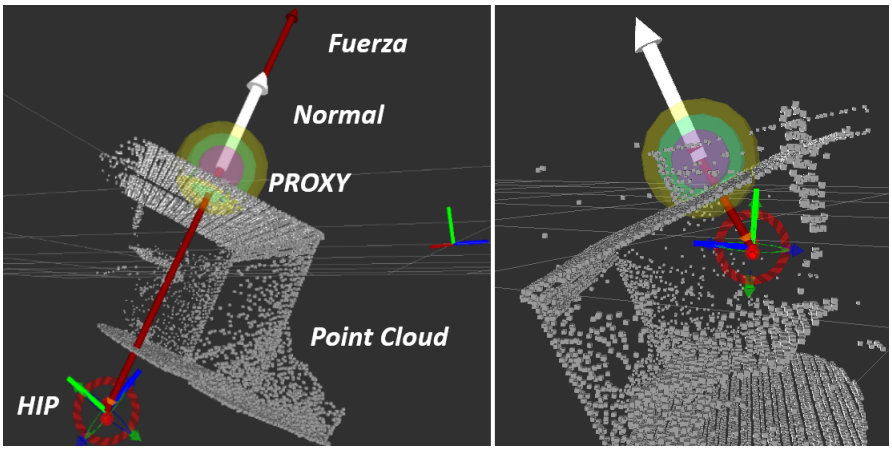

Figura 3.16: Marcadores en *Rviz* sin modelo virtual del robot *KUKA*.

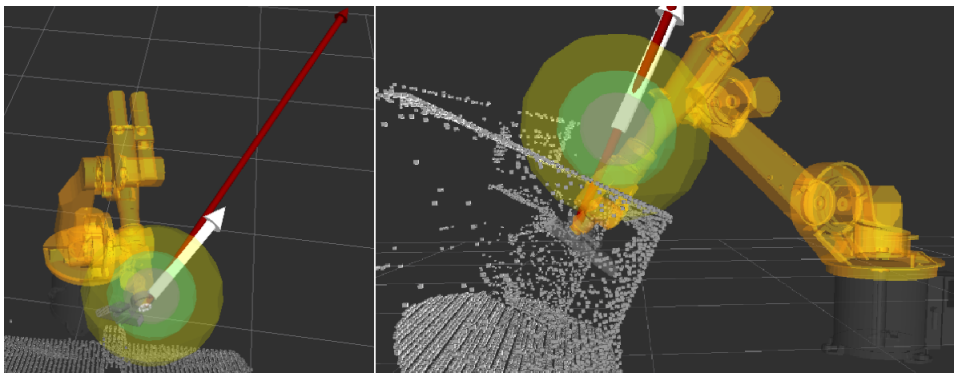

Figura 3.17: Marcadores en *Rviz* con modelo virtual del robot *KUKA*.

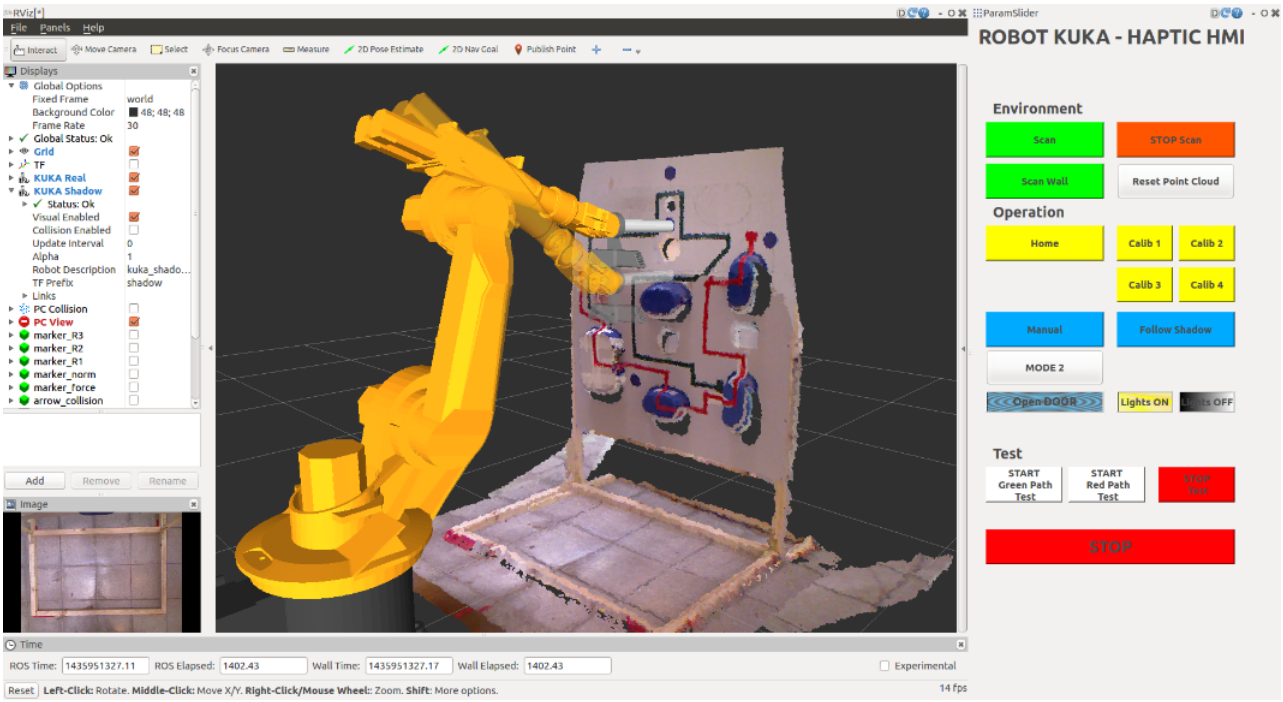

Figura 3.18: Interfaz gráfica para el usuario. Se observa la pantalla de *Rviz* con los elementos *robot-shadow*, *robot-real*, recuadro de video en tiempo real, *Point Cloud*, y el menú principal.

3.13. Interfaz gráfica para el usuario

Módulo *Rviz*

El programa *Rviz* es un visualizador que proporciona vistas interactivas (rotación, desplazamiento y acercamiento) del entorno de trabajo del robot virtual, y el nube de puntos que representa al entorno de trabajo real. En la Figura 3.18 se observa al lado izquierdo el programa *Rviz* con todas las opciones de configuración, en donde se pueden quitar o agregar componentes, como por ejemplo la *Point Cloud*, los links del *robot-shadow* o del *robot-real*, etc. Además, en la esquina inferior izquierda se puede observar un recuadro con el video en vivo de la cámara RGB que incorpora el *Kinect*. En Anexos se agregan las Figuras 7.17, 7.18, 7.19 y 7.20 de la interfaz gráfica.

Menú principal

En el lado derecho de la Figura 3.18 se observa el panel de Menú Principal con todas las opciones que los usuarios o teleoperadores necesitan. Estas opciones se dividen en tres secciones: *Environment*, *Operation*, y *Test*. A continuación se detalla la función de cada uno de los botones que conforman estas tres subsecciones:

1. *Environment*:

   - *Scan*: Obtiene seis imágenes 3D desde distintos puntos de vista cubriendo distintos ángulos del *workspace* del robot.
   - *Scan Wall*: Obtiene una imágen 3D del *workspace* del robot.
   - *Stop Scan*: Interrumpe el proceso de captura de imágenes 3D del entorno.
   - *Reset Point Cloud*: Elimina la *Point Cloud* acumulada hasta el momento (luego de las sucesivas capturas de imágenes).

2. *Operation*:

   - *Home*: Envía al robot a la posición central (ángulo cero) de cada *joint*.
   - *Calib 1, Calib 2, Calib 3 y Calib 4*: Envía al robot a las posiciones de calibración del panel de prueba. Este proceso se lleva a cabo cuando se desea ajustar la ubicación del panel con respecto al robot.
   - *Manual*: Modo de teleoperación manual, en el cual el usuario debe presionar el botón gris 1 del *stylus* del *Phantom Omni* para mover el brazo robótico desde la posición *robot-real* hasta la posición del *robot-shadow*.
   - *Follow Shadow*: Modo de teleoperación experimental, el en cual el *robot-real* sigue continuamente (con un *delay* inherente) a la posición *robot-shadow*. Este método de teleoperación sólo sirve para hacer algunas pruebas en la cola de prioridad de mensajes y no se utilizó para realizar la validación experimental del método.
   - Otros botones: Definen posiciones fijas en cada *joint* para hacer algunas pruebas de funcionamiento.

3. *Test*:

- *START Green Path Test*: Ubica la efector del robot al inicio del camino de operación verde e inicia el cronómetro. Inicia el proceso de grabación de todos los parámetros del robot y del entorno.
- *START Red Path Test*: Ubica la efector del robot al inicio del camino de operación rojo e inicia el cronómetro. Inicia el proceso de grabación de todos los parámetros del robot y del entorno.
- *STOP Test*: Detiene el cronómetro y detiene la grabación de parámetros del robot y el entorno.
- *STOP*: Parada de emergencia. Detiene el movimiento del robot sin importar en que etapa de la teleoperación se encuentra.

En el siguiente Capítulo se muestra la configuración experimental construida para estudiar la influencia de la retroalimentación háptica en la precisión y desempeño general que tienen los usuarios, cuando utilizan la metodología de teleoperación presentada en esta tesis.

4 Validación Experimental

Se diseñan experimentos para validar el sistema propuesto de Teleoperación Háptica con tiempo de retraso en la comunicación, además de comparar esta metodología utilizando retroalimentación visuo-háptica (a través del sentido de la vista y el tacto) con una retroalimentación visual pura, con el objetivo de comprobar la hipótesis de que el sistema de teleoperación háptico mejora el rendimiento del operador. Por otra parte se busca comparar los desempeños de las dos metodologías de cinemática inversa propuestas ("*Workspace* proporcional" y "*Deltas* de movimientos del efector").

En el presente capítulo se muestra el diseño de los experimentos realizados junto a la configuración experimental (dispositivos utilizados), la descripción de los sujetos de prueba (usuarios del sistema de teleoperación) y el procedimiento seguido.

4.1. Configuración Experimental

4.1.1. Dispositivos utilizados

Para poner a prueba el sistema de teleoperación propuesto se utilizan los siguientes dispositivos:

- Una cámara RGB-D *Microsoft XBOX Kinect*.
- Un dispositivo de control háptico *Phantom Omni SensAble Technologies* de *3D Systems-Geomagic Solutions*.
- Un robot industrial *KUKA KR 6-2* de 6 grados de libertad (6-DOF) con su computador original que se identificará con el nombre *ROBOT-PC* (*Microsoft Windows 95*, 32 bits, procesador *Intel Pentium* 200 MHz, 128 Mb RAM).
- Un computador "cliente" que se identificará con el nombre de *CLIENT-PC* (*Ubuntu 14.04*, 64 bits, procesador *Intel Core i7-3770* 3.4 GHz, 8 Gb RAM) en donde se conecta la cámara RGB-D, el dispositivo de control háptico y la interfaz de comunicación con el computador del robot teleoperado.

4.1.2. Robot *KUKA KR 6-2*

El *KUKA KR 6-2* (Ver Figura 4.1) es un brazo robótico industrial que cuenta con 6-DOF *Degrees-Of-Freedom* o grados de libertad, utilizado comúnmente para realizar tareas repetitivas las cuales se programan una vez y luego se ejecutan varias veces sin necesidad de detenerse [68]. Por este motivo, el robot *KUKA KR 6-2* es ideal para operaciones de montaje, soldadura o ensambles en industrias manufactureras o fábricas en serie (como por ejemplo en la industria automotriz). Este tipo de brazo robótico no está diseñado para realizar teleoperación, ni recibir instrucciones "*online*" a medida que ejecuta los movimientos pedidos, ya que por el contrario se especializa en ejecutar ciclos de instrucciones previamente grabadas. Para que este robot sirva para los propósitos planteados en la presente tesis, se recurre a una *API* externa (no original de la marca *KUKA*) cuya funcionalidad se detalla más adelante.

Figura 4.1: Robot industrial *KUKA KR 6-2*. (Imagen obtenida de [68])

La empresa *KUKA Roboter GmbH* diseña y construye los robots industriales *KUKA*. El mecanismo de comunicación por defecto habilitado es a través de la interfaz serial que está en el computador que viene con el *KUKA*. Este sistema de comunicación, como se mencionó anteriormente, no está diseñado para dar instrucciones de movimiento en tiempo real, sino que para configurar diversos parámetros del sistema. Para poder dar instrucciones en tiempo real al robot *KUKA* existen alternativas de software externos que se pueden instalar, los cuales son:

- *KUKA RobotSensorInterface*: El cual permite realizar acciones en tiempo real sobre el programa en ejecución, utilizando datos de sensores. Además permite la comunicación no cíclica con un computador externo vía *TCP/IP*.
- *KUKA Ethernet KRL (KUKA Robot Language) XML*: El cual permite establecer una comunicación no cíclica entre el computador interno del *KUKA* y un computador externo, utilizando transmisión de datos (instrucciones) por cadenas de XML a través del protocolo *TCP/IP*.

Los dos paquetes de software mencionados sólo son compatibles con versiones más ac-

tualizadas del sistema operativo del *KUKA* denominado *KSS (KUKA System Software)*. Actualizar el sistema operativo del robot *KUKA* instalado en el Laboratorio de Mecatrónica del Departamento de Ingeniería Eléctrica (*DIE*) de la Universidad de Chile (a la fecha de la publicación de la presente tesis) implica un alto costo, ya que se trata de robots industriales, los cuales no están dirigidos a realizar investigación e incluso podría ser más conveniente realizar la compra de un robot *KUKA* más nuevo que incluya estas prestaciones. Es por este motivo, que se busca una solución alternativa para dejar operativo el robot con el que cuenta el *DIE*, no sólo para realizar la validación experimental de esta tesis, sino que para futuras posibles investigaciones.

En el extremo del efector final (articulación número 6) del brazo robótico *KUKA* se fija un tubo de PVC de 5[*cm*] de diámetro y 30[*cm*] de largo. Esto se hace para tener una punta efectora a una distancia prudente de la estructura del robot y del lugar donde se ubica la cámara RGB-D, de tal forma, de proteger estas estructuras ante una colisión con los elementos del entorno. La orientación (ángulo de inclinación con respecto al suelo) del efector final del robot se deja fija para que se mantenga perpendicular al panel de pruebas, y de esta forma, facilitar la operación de los usuarios quienes sólo deben preocuparse de la posición del efector.

Joints del *KUKA* y *masterización*

Las 6 articulaciones o *joints* del robot *KUKA* poseen *encoders*, con los cuales se puede conocer con precisión el ángulo en el cual está cada uno de ellos. Se utiliza esta información para definir la posición *home* (Figura 4.2), la cual representa la posición de reposo o inicial del robot. El ángulo *home* de cada *joint* es considerando como referencia (ángulo 0°) para definir todos los demás ángulos posibles. En la Tabla 4.1 se muestra la transformación (*offset*) que hay que agregar a los ángulos con respecto al *KRL (KUKA Robot Language)* para obtener el sistema de referencia de *ROS*. Por otra parte, cada *joint* está limitado superior e inferiormente por el *KSS (KUKA System Software)* para evitar colisiones con el suelo, y evitar alcanzar el límite físico de giro que entregan los motores.

Tabla 4.1: Ángulos para posición *home* según dos sistemas de referencia

Joint	*home* en *KRL*	*home* en *ROS*
A_1	0°	0°
A_2	−90°	0°
A_3	+90°	0°
A_4	0°	0°
A_5	0°	0°
A_6	0°	0°

La *masterización* del robot se refiere a la calibración de los *encoders* de cada *joint*. Para asegurar una calibración perfecta, el fabricante sugiere que se lleve a cabo por un servicio técnico de personal especializado, ya que ellos cuentan con los sensores necesarios para realizar las mediciones precisas y las herramientas en caso de que se requiere un ajuste de piezas en el robot.

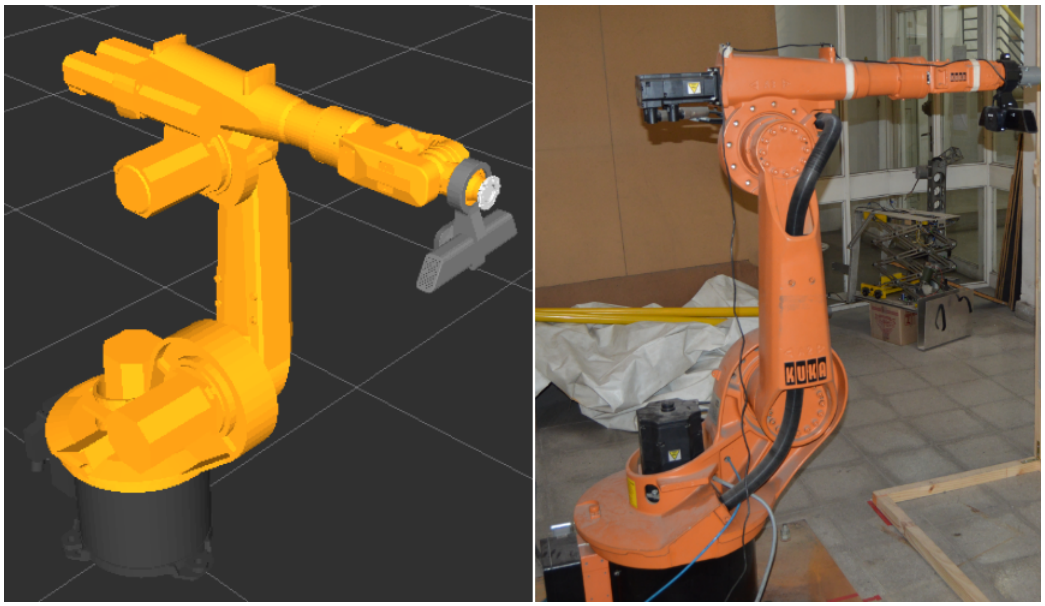

Figura 4.2: Izq: *KUKA* simulado en posición *home*. Der: *KUKA* real en posición *home*. Ambas imágenes fueron tomadas desde ángulos distintos.

4.1.3. Panel de Pruebas

Se necesita una zona segura en la cual el robot pueda ejecutar la tarea teleoperada reduciendo riesgos de accidentes, por lo cual se diseña y construye un panel de pruebas (Ver Figura 4.3) hecho de madera y cartón piedra, perpendicular al suelo tal como se ve un bastidor colocado en un atril. El objetivo de las pruebas hápticas es que el robot teleoperado interactúe directamente con el panel (que el efector final se posicione lo más cerca posible de este), lo cual implica un alto riesgo de colisión. Es por este motivo que el panel es liviano, resistente a golpes y flexible, para que pueda soportar choques con el brazo y que este último no resulte dañado. El panel está diseñado para ser reparado fácilmente en caso de una colisión importante. Además se colocan carcasas de plástico de manera de crear plataformas irregulares sobre el plano del panel de pruebas.

4.1.4. Software de control y *API*

El sistema de control del robot *KUKA* cuenta con dos sistemas operativos ejecutados de forma paralela. El primero de ellos es *Microsoft Windows 95* el cual se encarga de la visualización y manejo de archivos, y el segundo corresponde a *VxWorks* de la empresa *Wind River Systems*, el cual es un sistema operativo con ejecuciones en tiempo real encargado del control del movimiento del robot. El *KSS (KUKA System Software)* es el programa intermediario que se encarga de comunicar los dos sistemas mencionados anteriormente, el cual posee librerías de control eficientes, para obtener un sistema de comunicación con la menor cantidad de retardos posibles (entre *VxWorks* y *Windows*).

El acceso a las variables de control es restringido (para leer, editar y eliminar) debido a

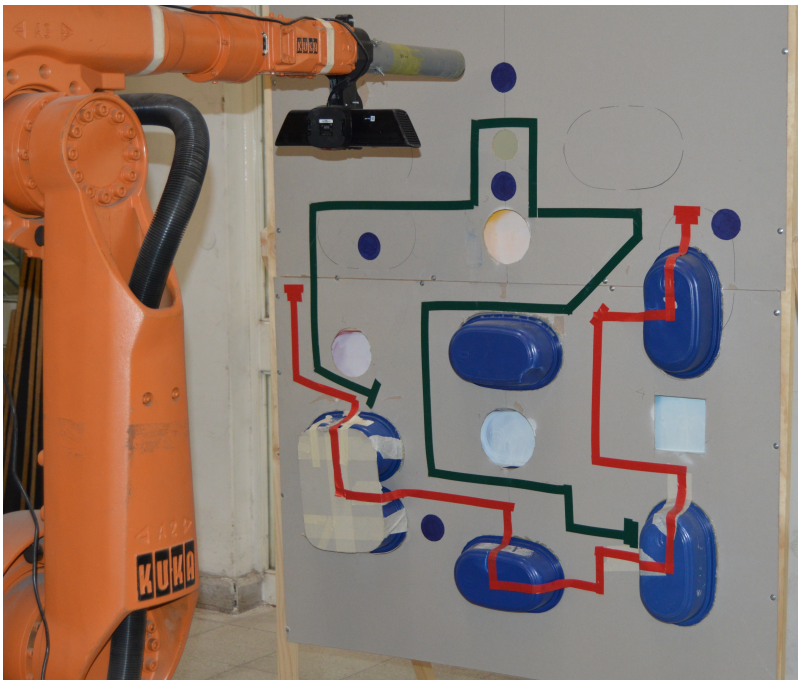

Figura 4.3: Panel de pruebas hápticas.

la naturaleza del robot *KUKA* el cual no está orientado a investigación o desarrollo como se explicó anteriormente. Realizar un cambio en la posición de los *joints* del robot resulta ser el primer gran desafío de este proyecto. El software se desarrolla sobre una *API* para el *KUKA* creada por Chavent et al. [69] [70] el cual es utilizado para definir la posición de cada *joint* del robot a través de una arquitectura cliente-servidor basado en el protocolo de comunicación TCP/IP (ver Figura 3.2). Esta librería creada por Chavent et al. [69] [70] monta un servidor RPC (*Remote Procedure Call*) en el sistema operativo *Windows* y permite acceder y modificar variables del *KSS (KUKA System Software)*. Para modificar estas variables, el cliente (*CLIENT-PC*) debe estar conectado a la misma red local que el servidor y así poder enviar peticiones al *ROBOT-PC* (para ver más detalle de estas conexiones, ver sección de Anexos).

El software utilizado en el computador *ROBOT-PC*, se desarrolla en *KRL (KUKA Robot Language)*. Estos códigos permiten generar un sistema de control de movimiento básico, creando rutinas que ejecutan acciones específicas dependiendo del valor de ciertas variables (que pueden ser cambiadas externamente a través de la conexión *TCP/IP* utilizando el desarrollo de Chavent et al. explicado anteriormente). El Código Fuente 7.3 (ver Anexos) muestra un código *KRL (KUKA Robot Language)* que ejecuta un movimiento del robot desde la posición actual hacia las coordenadas especificadas en la variable *API MOVE AXIS*, el cual es accionado por en cambio de valor en variable *API MOVE ID*. Este tipo de movimiento se denomina *PTP (Point to Point)* y permite definir la posición deseada de cada *joint* del robot *KUKA*. Las variables que se desean cambiar de forma remota se deben definir de manera global, siendo declaradas en el archivo *config.dat*. Se han definido una serie de variables de uso libre (para ser modificadas de manera remota), cuyo listado se puede encontrar en Anexos, Código fuente 7.4.

El software utilizado en el computador *CLIENT-PC*, se desarrolla en los lenguajes *C++* y *Python* (entorno virtual de teleoperación y sistema de control) trabajando sobre *Robotic Operating System* (versión *ROS Indigo Igloo*) [28] usando *Point Cloud Library (PCL)* [66] y *OpenHaptics Toolkit* [71]. En la sección Anexos se incluyen las Figuras 7.2, 7.3, 7.4 y 7.5 que representan los nodos y tópicos principales de *ROS* que componen el software del proyecto.

La API desarrollada por Chavent et al. está escrita en lenguaje C por lo que se puede integrar a *ROS* fácilmente, implementando el código en C++. Se realizaron cambios en esta librería, haciendo más sencillo el manejo de errores para evitar algún conflicto con los mensajes internos de *ROS*, que también posee un tratamiento de detección de errores utilizando variables booleanas. La API sólo provee de funciones para asegurar la comunicación con el servidor RPC (*Remote Procedure Call*) en las que incluye petición o modificación de una variable global, por lo tanto, el manejo envío del mensaje deseado se realiza en una clase independiente que permite encapsular estos mensajes.

Realizar un cambio en las variables internas del robot, como también realizar una petición de estado actual de posiciones del mismo, toma aproximadamente 200$[ms]$, lo cual hace que la comunicación entre el *CLIENT-PC* (computador donde se corre el algoritmo de control háptico y que trabaja con la arquitectura de *ROS*) y el *ROBOT-PC* (computador interno del robot) sea limitada. Se permite entonces una sola petición *get position o set position* desde el computador cliente hacia el servidor (y viceversa) hasta que se termine de completar. Este tiempo de retraso en la comunicación con un dispositivo teleoperado no es un problema nuevo (ya se ha visto por ejemplo en sistemas aeroespaciales), y se ha abordado con diferentes metodologías. El retraso en la comunicación está dado principalmente por las condiciones del canal de comunicación entre emisor y receptor (largas distancias, ruido que podría modificar el mensaje, redundancia y comprobación de mensajes, entre otros motivos).

La limitación principal de la *KUKA API* es que sólo permite la comunicación en una dirección a la vez entre el *CLIENT-PC* y el *ROBOT-PC*, por lo tanto, para evitar que las instrucciones de cambio de posición se acumulen y desordenen, se debe respetar una prioridad asignada a cada mensaje. Esto se lleva a cabo en una cola de prioridad implementada en el nodo *ROBOTIC ARM DRIVER* dentro del *CLIENT-PC* . Cuando la cola está vacía, el nodo envía una petición de *Joint States* al robot, que entrega la posición actual de cada *joint* del robot (ver Figura 4.4). La prioridad de los mensajes es ordenada según la importancia que tienen estos para reducir las probabilidades de colisión (o reaccionar más rápidamente en caso de un posible accidente), por lo tanto se asignan en el siguiente orden:

Prioridad 1: *Stop*. Detiene todos los movimientos del robot.

Prioridad 2: *Point to Point*. Envía a cada *joint* del robot a la posición deseada.

Prioridad 3: *Set Velocity*. Cambia la velocidad de movimiento del robot.

El nodo *ROBOTIC ARM DRIVER* (cuyo nombre dentro del proyecto es *kuka_driver*), utiliza un servidor de *ROS* para implementar la cola de prioridad, ya que de esta forma es directo realizar una petición de cambio de posición de los *joints* desde cualquier otro nodo, para posteriormente esperar una respuesta del servidor, que contiene la información de posición actual del robot o cualquier otra variable de estado del sistema. En la sección

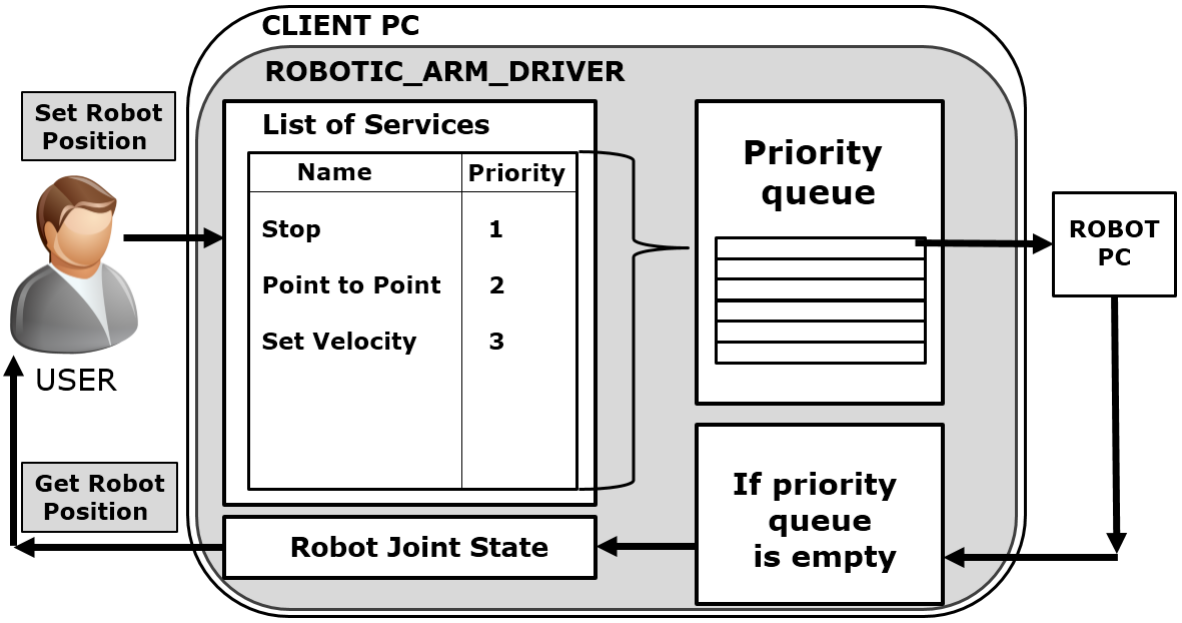

Figura 4.4: Esquema del nodo *ROBOTIC ARM DRIVER* de *ROS*.

de Anexos (Código fuente 7.6) se muestran las líneas de comando necesarias para ejecutar el servidor *kuka_ driver* y un *script* de ejemplo (Código fuente 7.5) que ejecuta un procedimiento de movimientos del robot *KUKA* preestablecido.

4.1.5. Parámetros del Software

Los parámetros utilizados en el software de control háptico (el cual es explicado en detalle en los capítulos anteriores) se muestran en la Tabla 4.2.

Tabla 4.2: Parámetros usados en la evaluación experimental

Símbolo	Parámetro	Valor
R_1	Esfera de radio R_1 del *Proxy*	0,01$[m]$
R_2	Esfera de radio R_2 del *Proxy*	0,04$[m]$
R_3	Esfera de radio R_3 del *Proxy*	0,10$[m]$
d_k	Tamaño del paso del movimiento del *Proxy*	0,01$[m]$
K_{spring}	Constante elástica del resorte	80$[N/m]$
H_X	Eje X del Workspace Háptico	0,15$[m]$
H_Y	Eje Y del Workspace Háptico	0,30$[m]$
H_Z	Eje Z del Workspace Háptico	0,26$[m]$
K_X	Eje X del Workspace *KUKA*	0,70$[m]$
K_Y	Eje Y del Workspace *KUKA*	1,40$[m]$
K_Z	Eje Z del Workspace *KUKA*	1,00$[m]$

4.2. Usuarios

Un total de 20 usuarios voluntarios (una persona de sexo femenino y 19 personas de sexo masculino) entre 19 y 31 años de edad, participaron en los experimentos. Cada uno de ellos utilizó su mano dominante (con la cual se siente más cómodo escribiendo) para manipular el dispositivo *Phantom Omni*. Un total de 4 zurdos y 16 diestros fueron puestos a prueba.

Ningún teleoperador presentó problemas motores o neurológicos. Algunos problemas visuales que se identificaron en estos usuarios fueron: 2 personas con astigmatismo, 3 con miopía, y 1 daltónico. Sin embargo, estos problemas a la vista no presentan ningún impedimento para los experimentos y no presentaron una alteración en los resultados.

Todos los sujetos de prueba entregaron su consentimiento previo a las pruebas, y mostraron entusiasmo por colaborar con este estudio. Durante las pruebas de teleoperación, los usuarios sólo pueden observar la pantalla, en donde se muestra el entorno virtual (junto a los robots guía) donde deben ejecutar la tarea de teleoperación. Ninguno de ellos tuvo la posibilidad de ver el entorno real en donde el robot *KUKA* realiza los movimientos. La idea de que los usuarios sólo tengan acceso visual al entorno virtual del robot (y no al entorno real) es para reproducir las condiciones en las que la teleoperación se hace a kilómetros de distancia, en un lugar peligroso o de difícil acceso a personas.

4.3. Procedimiento de los experimentos

Para evaluar el sistema de teleoperación, se crean dos "caminos" (verde y rojo) que son usados como guía para tareas de seguimiento de trayectoria. Las trayectorias ideales que debe seguir el teleoperador utilizando la punta del efector del robot *KUKA* están determinadas por estos "caminos" que son líneas demarcadas sobre la superficie del panel de pruebas. La primera tarea pedida a los usuarios consiste en desplazar la punta del efector final del robot sobre la trayectoria en 2D (línea verde) del panel, y la segunda tarea consiste en desplazar esta punta sobre la trayectoria 3D (línea roja) del panel. Estas trayectorias ideales se muestran en la Figura 4.5 en colores verde y rojo respectivamente. En la Figura 4.6 se puede observar el set experimental completo en el mundo real (izq) y el mundo virtual (der), en donde se destaca el panel de pruebas, las trayectorias objetivo que deben seguir los operadores.

Ambas tareas de seguimiento de trayectoria (2D y 3D) deben ser completadas de manera tal que la punta del efector final del robot esté lo más cerca del panel de pruebas como sea posible, evitando cualquier colisión entre ellos y terminando el recorrido en el menor tiempo posible. El punto de inicio de las trayectorias (2D y 3D) está ubicado en el lado izquierdo del panel y el punto final se ubica en el lado derecho. Los operadores deben mover el efector del robot *KUKA* real en segmentos de aproximadamente 20 a 30$[cm]$ para obligarlos a realizar varios ajustes de posición al avanzar por la línea del experimento, evitando de esta manera que muevan el robot en largas trayectorias rectas. Para mover el robot *KUKA* real, los usuarios deben ocupar la información entregada por el simulador (entorno virtual), ensayando el desplazamiento del *robot-real* a través del movimiento del *robot-shadow* el cual proporciona

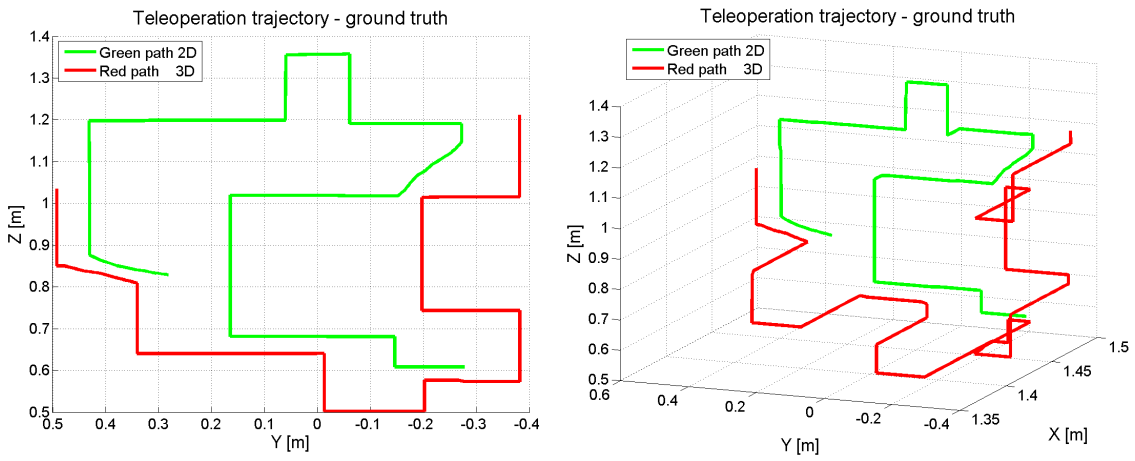

Figura 4.5: Trayectorias objetivo (ideales) para teleoperación en el panel de pruebas: 2D (*Green path*) y 3D (*Red path*). Vistas frontal (izq) e isométrica (der).

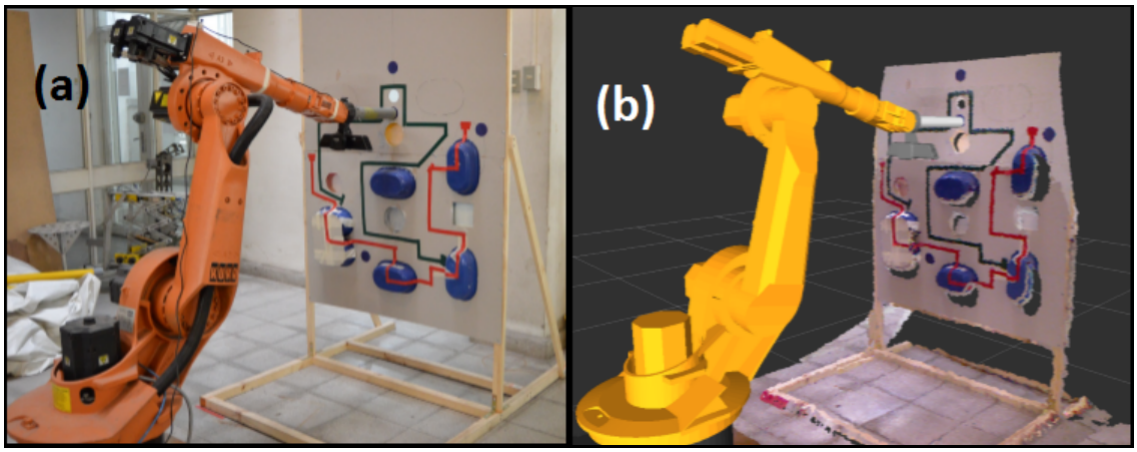

Figura 4.6: (a) Entorno real de operación del robot *KUKA*. (b) Entorno virtual de operación del robot *KUKA*, utilizando los modelos de *robot-shadow* y *robot-real*.

la información háptica y/o visual en los casos correspondientes. Estas indicaciones son dadas a cada usuario antes de empezar con las pruebas.

Cada usuario debe completar estas dos tareas (seguimiento de trayectoria en 2D y 3D) usando los siguientes modos de teleoperación, en el orden que se define a continuación:

Teleop1: *Workspace* proporcional con retroalimentación visuo-háptica.

Teleop3: *Workspace* proporcional con retroalimentación visual (no-háptica).

Teleop2: *Deltas* de movimientos del efector con retroalimentación visuo-háptica.

Teleop4: *Deltas* de movimientos del efector con retroalimentación visual (no-háptica).

La hipótesis que se quiere probar en esta tesis es que la teleoperación háptica mejora el rendimiento del usuario comparado con la teleoperación que no posee retroalimentación háp-

tica y se desea cuantificar esa mejora al utilizar el sistema bilateral con tiempo de retardo. El orden en que se realiza las pruebas afecta los resultados obtenidos, ya que los usuarios adquieren experiencia usando el primer modo de teleoperación y entonces naturalmente pueden obtener mejores resultados cuando realizan la prueba con el segundo o tercer modo de operación. Es por este motivo, que se separan a los usuarios en dos grupos. El primer grupo (10 operadores) empieza con las pruebas de teleoperación hápticas, mientras que el segundo grupo (10 operadores) empieza con las pruebas de teleoperación no-hápticas.

En el siguiente Capítulo se muestran los Resultados obtenidos en la etapa de validación experimental del sistema de teleoperación propuesto.

5 Resultados y Análisis

En el presente Capítulo se muestran los resultados obtenidos a partir de la validación experimental del sistema de teleoperación bilateral con retardo temporal que se propone en esta tesis. Estos resultados se dividen en tres secciones: Fuerza de retroalimentación, Resultados de Trayectoria 2D y Resultados de Trayectoria 3D. Finalmente se realiza un Análisis general de todos los resultados.

5.1. Fuerza de retroalimentación

La fuerza de retroalimentación se obtiene utilizando el algoritmo de renderizado háptico detallado en los capítulos anteriores, en donde el *proxy* y el *HIP* interactúan con la *Point Cloud* obtenida a partir del entorno de operación. La magnitud de la fuerza está determinada por la diferencia de posición entre el *proxy* y el *HIP*, por lo tanto, la velocidad de respuesta háptica está determinada por la velocidad en la cual se detecta esta diferencia de posición. Para mostrar la sincronización entre la fuerza obtenida, y la diferencia de posición entre el *HIP* y el *proxy*, se muestran a continuación los gráficos correspondientes a la magnitud de la fuerza (en la dirección del eje x) junto a la posición del *HIP* y el *proxy* (en la misma dirección del eje x) en función del tiempo de operación. El análisis para los ejes y y z es análogo a lo que ocurre para el eje x ya que el algoritmo es exactamente el mismo para cada dirección.

En las Figuras 5.1, 5.2 y 5.3 se observa la acción del usuario desplazando en línea recta el *HIP* en dirección al panel de pruebas, acercando la punta del efector del robot de manera perpendicular al plano. Una vez que el *proxy* queda detenido por la acción de la *Point Cloud* (generada a partir del panel de pruebas) a 1,45$[m]$ desde la base del robot hacia adelante (dirección eje x), se produce una diferencia en la posición del *proxy* (que queda detenido) y el *HIP* que se mueve libre por el espacio. A partir del momento en el cual se produce esta diferencia de posición, se genera la fuerza que es retroalimentada al usuario a través del dispositivo *Phantom Omni*. La magnitud de esta fuerza es directamente proporcional a la diferencia de posiciones mencionada anteriormente, lo cual permite dar la sensación al usuario de que a mayor profundidad de penetración en un objeto virtual con el efector del robot, mayor es la fuerza de repulsión generada, lo cual obliga al operador a salir de esa posición.

En la Figura 5.1 se observa que el usuario (a través del efector del robot) ingresa a la zona interior del objeto con el cual está interactuando y realiza movimientos suaves hacia dentro

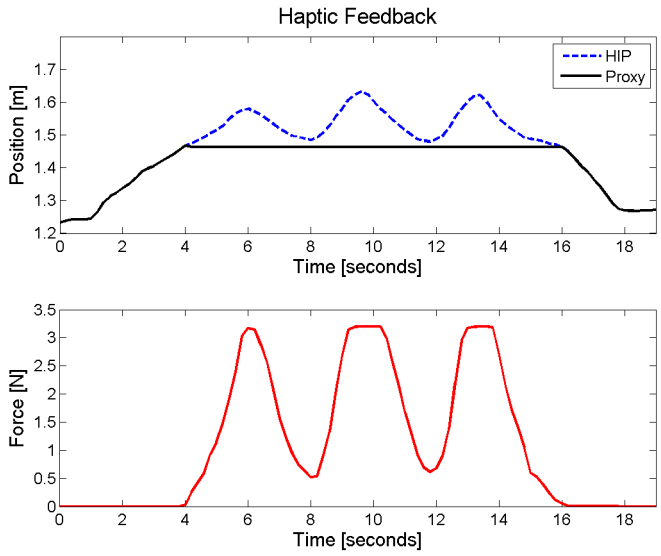

Figura 5.1: Gráfico de medición de fuerza de retroalimentación háptica unidimensional junto a la posición del *proxy* y *HIP* (sólo para eje x) versus el tiempo. El usuario ejerce presión en tres ocasiones contra del muro de *Point Clouds* una vez que el *HIP* atraviesa el umbral.

y hacia afuera del objeto, sin salir de su interior, produciendo una ondulación en la fuerza de repulsión. Además, se observa que la fuerza generada está saturada por 3,3$[N]$ por lo tanto, a partir de cierta distancia hacia el interior del obstáculo, la fuerza generada no superará los 3,3$[N]$, el cual es un límite dado por el dispositivo háptico.

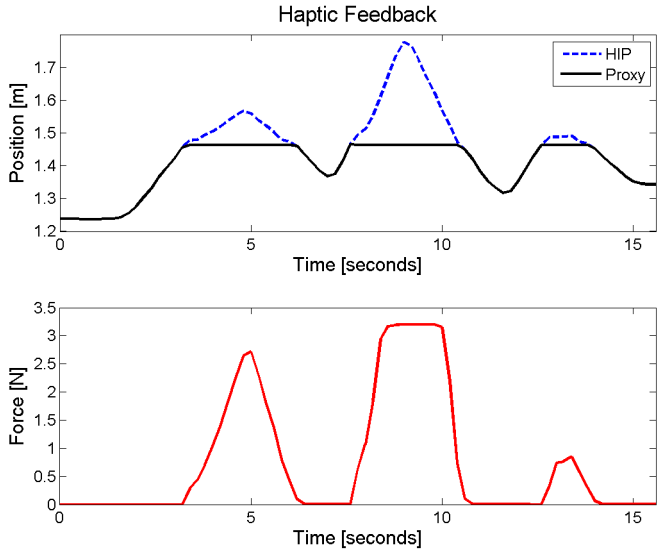

Figura 5.2: Gráfico de medición de fuerza de retroalimentación háptica unidimensional junto a la posición del *proxy* y *HIP* (sólo para eje x) versus el tiempo. El usuario penetra el muro de *Point Clouds* en tres ocasiones, con distinta velocidad y distinta profundidad.

En la Figura 5.2 se observa un ejercicio similar al de la Figura 5.1 pero esta vez, se permite al *HIP* desplazarse desde el interior hacia afuera del objeto, lo cual produce fuerza de 0$[N]$.

No se observan retardos en la respuesta háptica, lo cual permite asegurar un comportamiento natural y realista del algoritmo cuando se esté interactuando con los objetos del *workspace* virtual. El segundo *peak* sobrepasa el valor límite de profundidad, lo cual produce una saturación de la fuerza háptica, pero a continuación con el tercer *peak* se verifica que la sensibilidad de la fuerza obtenida no se ve alterada, a pesar de sobrepasar el valor límite anteriormente.

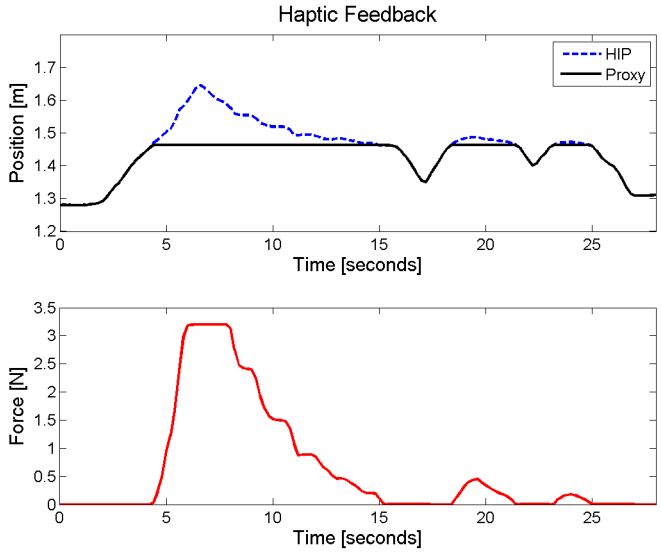

Figura 5.3: Gráfico de medición de fuerza de retroalimentación háptica unidimensional junto a la posición del *proxy* y *HIP* (sólo para eje x) versus el tiempo. El usuario penetra el muro de *Point Clouds* rápidamente, para luego retirar el *HIP* de manera lenta, y manteniendo fijo el efector durante unos segundos. Las otras dos ocasiones en donde el *HIP* atraviesa el muro, se realiza de manera sutil.

En la Figura 5.3 se observa un retiro escalonado del *HIP* desde el interior del objeto virtual, de manera de poder mantener por un instante el valor de la fuerza de retroalimentación constante y lo más estable posible. El resultado es el esperado, con lo cual se puede asegurar que la fuerza de retroalimentación es estable y responde de manera natural al comportamiento que tienen el *proxy* y el *HIP* con los objetos con los cuales interactúan.

5.2. Resultados de Trayectoria 2D

En la presente sección se muestran y analizan los resultados obtenidos con las pruebas de seguimiento de trayectoria 2D sobre el panel de pruebas, utilizando los cuatro modos de teleoperación bilaterial con retraso temporal.

En la Figura 5.4 se muestran las trayectorias del efector del robot sobrepuestas, obtenidas a partir de los 20 usuarios cuando completaron la tarea utilizando el Modo *Teleop1* y *Teleop3*. A simple vista se observa que los puntos azules, que corresponden al Modo *Teleop3* (visual no háptica), se encuentran más dispersos sobre el panel que los puntos negros (Modo *Teleop1*), lo cual es reflejo de un mayor error de cada punto comparado con la posición más cercana de la trayectoria ideal que se ubica sobre el panel.

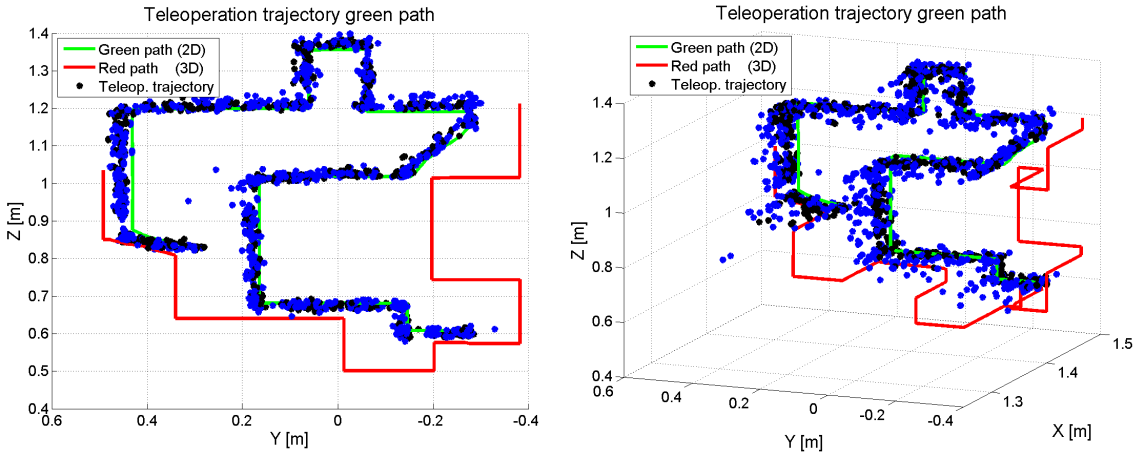

Figura 5.4: Trayectorias de teleoperación - Camino verde (2D): A la izquierda vista frontal y a la derecha vista isométrica. Los puntos corresponden a las muestras de las trayectorias de teleoperación de los 20 usuarios. Los puntos negros con el Modo *Teleop1* (visuo-háptica) y los puntos azules con el Modo*Teleop3* (visual no háptica).

Cada punto de la Figura 5.4 representa una posición de pausa del efector del robot real, lo cual significa que estas muestras no son tomadas mientras el robot está en movimiento, sino que son posiciones guardadas una vez que se detiene el movimiento del robot. Como se explicó anteriormente, se les indicó a los usuarios que movieran el efector del robot en tramos cortos de unos 20 a 30$[cm]$ aproximadamente, para que tengan que actualizar la posición del robot en varias ocasiones (el mismo número de veces que se guarda el dato de la posición del efector), evitando que desplazaran la punta efectora en largas líneas rectas, lo cual facilitaría la tarea y por otra parte, reduciría la cantidad de datos a ser analizada para obtener conclusiones sobre los beneficios del sistema teleoperado con retroalimentación háptica.

5.2.1. Trayectoria 2D: Error promedio de posición del efector

El error de la posición del efector está determinado por la distancia (perpendicular al panel de pruebas) mínima, desde la punta del efector hasta el punto más cercano al camino objetivo (verde 2D o rojo 3D). Entonces, por cada posición nueva del efector del robot real, se tiene asociado un error. Finalmente, el promedio de esos errores entrega una visión general de la precisión de cada uno de los usuarios al realizar la tarea de seguimiento de trayectoria.

La Figura 5.5 muestra el error promedio del efector en un gráfico de barras para la tarea de seguimiento de trayectoria 2D (camino verde), junto a la respectiva desviación estándar que se representa por una línea sobre cada una de las barras. Las barras de color gris representan los modos de teleoperación que poseen retroalimentación háptica (modos *Teleop1* y *Teleop2*) en cambio las barras blancas sólo tienen retroalimentación visual en sus modos de control.

Con los datos obtenidos, se realizan "Pruebas-T para muestras apareadas" cuyo fundamento teórico se presentó en el Capítulo 2. Como se mencionó anteriormente, se realiza una

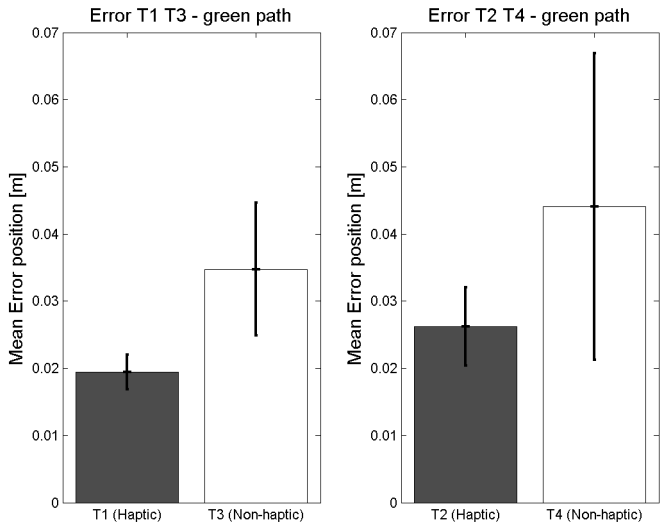

Figura 5.5: Error promedio y desviación estándar de la posición del efector usando los cuatro Modos de teleoperación - Camino verde (2D).

comparación entre los resultados obtenidos utilizando retroalimentación háptica y los resultados que se obtuvieron cuando no se utilizó esta tecnología. El nivel de significancia elegido para todas las pruebas es de $P = 0{,}1$ lo cual representa el umbral para definir si el cambio mostrado al utilizar retroalimentación háptica es estadísticamente significativo o no. En caso de que no sea estadísticamente significativo, quiere decir que la diferencia mostrada en los resultados cuando se utiliza retroalimentación háptica v/s cuando no se utiliza háptica, es producto del azar (aleatoriedad en las muestras de ambos grupos) con una probabilidad mayor a 0,1.

Comparando los resultados obtenidos (Figura 5.5) de *Teleop1* con *Teleop3* en la tarea de seguimiento de trayectoria 2D (camino verde), cuando se utiliza retroalimentación háptica (*Teleop1*) se reduce en un 44,0 % el error promedio. Con *Teleop2* se obtiene un 40,5 % menos de error promedio comparado con *Teleop4* que no posee retroalimentación háptica. El *paired-sample T-test* muestra que estas diferencias son estadísticamente significativas, obteniendo un $P < 0{,}001$ en ambos casos.

Se observa además que con el modo de teleoperación denominado "*Workspace* proporcional" (*Teleop1* y *Teleop3*) se obtiene un menor error de posición promedio que con el modo "*Deltas* de movimientos del efector" (*Teleop2* y *Teleop4*) para los casos hápticos y no-hápticos, a pesar de que los usuarios comenzaron las pruebas con el modo "*Workspace* proporcional". Se espera que entre mayor sea el tiempo de práctica en la utilización del sistema de teleoperación, los usuarios mejoren su propio rendimiento en las tareas asignadas (debido a la curva natural de aprendizaje de un sistema tecnológico nuevo), sin embargo se observa que a los métodos "*Workspace* proporcional" y "*Deltas* de movimientos del efector" tienen curvas de aprendizaje independientes. Antes de realizar estas pruebas de manera oficial, se comprobó que realizar un modo de operación antes que el otro no afecta en el desempeño de los operadores, siendo más determinante el tiempo que utiliza cada uno en entrenarse en los modos de teleoperación.

5.2.2. Trayectoria 2D: Error promedio de posición del efector en función del tiempo

Para analizar como evoluciona el error de posición del efector del robot real a través de la trayectoria completa, se calcula una trayectoria "normalizada" en el tiempo para cada uno de los modos de teleoperación y cada uno de los usuarios sometidos a la validación experimental. La "normalización en tiempo" consiste en ajustar los datos de posición del efector a una escala temporal en el intervalo entre $0[s]$ y $100[s]$ desde el inicio de la trayectoria hasta el punto de final de esta. De esta manera todos los usuarios tendrán un tiempo de operación de $100[s]$ por lo que es posible promediar en cada segundo el error total de posición (considerando las 20 trayectorias correspondientes al total de sujetos de prueba), obteniendo finalmente una curva del error promedio de posición en función del tiempo normalizado. Este tipo de gráfico permite visualizar la variación del error a medida que se completa la tarea a realizar. Se espera que el error disminuya de manera gradual, ya que en los tramos finales del recorrido los usuarios llevan más tiempo practicando la labor de teleoperación, por lo que van refinando la sensibilidad que tienen con el dispositivo de control.

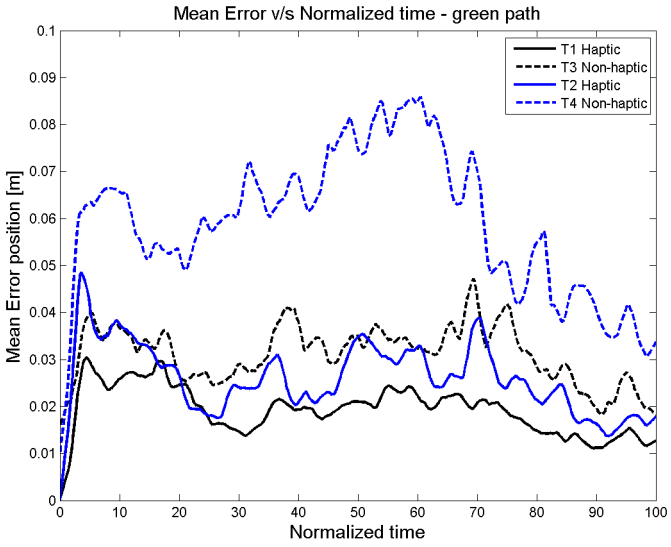

Figura 5.6: Error promedio de posición del efector en función del tiempo normalizado para los cuatro Modos de teleoperación - Camino verde (2D).

En la Figura 5.6 se observa que el error promedio en el tiempo (para la tarea de seguimiento de trayectoria 2D) disminuye continuamente de manera no monótona para los cuatro modos de teleoperación. El modo *Teleop1* mostró el menor error medio en prácticamente todos los segmentos temporales de operación. Le sigue el modo *Teleop2* con el segundo menor error de posición, lo que ratifica que la utilización de la interfaz háptica mejora el rendimiento de los usuarios para este tipo de tareas de teleoperación.

5.2.3. Trayectoria 2D: Número de colisiones

El número de colisiones representa la cantidad de veces en las que choca la punta del efector del robot real con el panel de pruebas. Se clasifican en dos tipos: colisiones débiles y colisiones fuertes. Las colisiones débiles ocurren cuando la punta del efector del robot real toca el panel pero no lo logra desplazar o empujar, en cambio las colisiones fuertes ocurren cuando la punta del efector golpea al panel de pruebas, ocasionando que este se desplace de su posición inicial (lo cual provoca un fuerte ruido). La cantidad de colisiones detectadas son anotadas por la persona que supervisa la evaluación experimental y no representan una complejidad adicional para el usuario que realiza la teleoperación, ya que este debe continuar con la prueba experimental sin importar el número de colisiones o la gravedad de estos.

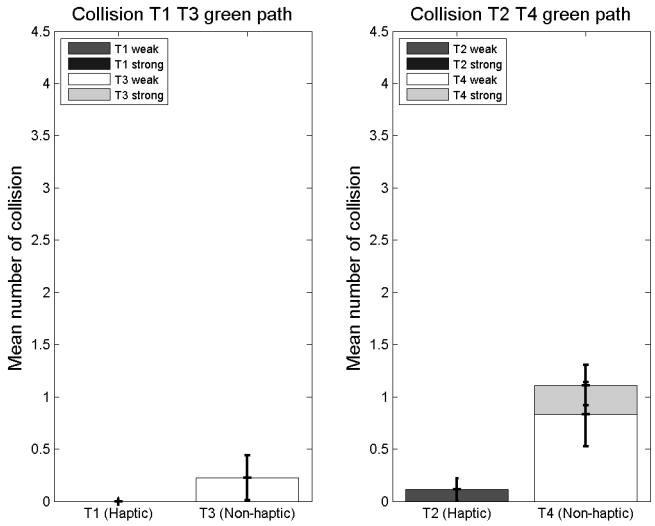

Figura 5.7: Número de colisiones promedio y desviación estándar para los cuatro Modos de teleoperación - Camino verde (2D). Se clasifican en colisiones débiles y fuertes.

En la Figura 5.7 se muestra el número de colisiones (fuertes y débiles) en promedio para los cuatro modos de teleoperación utilizados en la tarea de seguimiento de trayectoria 2D (camino verde). El gráfico de barras muestra por separado el número de colisiones débiles y fuertes, las cuales se sobreponen para obtener el número de la suma total de colisiones en promedio además de mostrar la línea (sobre cada barra) que representa la desviación estándar de los mismos.

Las colisiones detectadas (Figura 5.7) en estas pruebas muestran que la teleoperación con retroalimentación háptica no es infalible, pero puede reducir el número de colisiones de manera considerable. Comparando el modo *Teleop1* con *Teleop3* el número total de colisiones promedio se reduce en un 100 % al utilizar retroalimentación háptica, con un nivel de significancia de $P < 0{,}1$. Comparando el modo *Teleop2* con *Teleop4* el número total de colisiones promedio se reduce en un 90 % cuando se utiliza retroalimentación háptica, con un nivel de significancia de $P < 0{,}01$, lo cual implica que estos resultados son estadísticamente significativo para ambos casos. El modo de teleoperación "*Workspace* proporcional" presenta menor cantidad de colisiones promedio que al utilizar el modo "*Deltas* de movimientos del efector", comparando para los casos hápticos y no-hápticos.

5.2.4. Trayectoria 2D: Número de ajustes de posición y Tiempo promedio de teleoperación

El número de ajustes de posición del efector del robot se define como el número de veces que el operador tiene que cambiar la posición del efector real del robot, debido a que desea mejorar la posición con respecto al punto del camino más cercano o porque desea avanzar en la ruta. Cada operador buscará avanzar en el camino objetivo de manera de minimizar la distancia que existe desde la punta del efector hasta el panel de pruebas, por lo tanto, se podrá contabilizar un mayor número de ajustes cuando el usuario no se sienta seguro de la posición y requiera mejorarla pero siempre avanzando por el camino (nunca vuelve atrás, aunque se encuentre muy alejado del camino objetivo).

Por otra parte, se obtiene el tiempo promedio que demoran los usuarios en completar el recorrido completo pedido, utilizando los cuatro modos de teleoperación. Los resultados del número de ajustes promedio y el tiempo promedio de teleoperación se encuentran en la Tabla 5.1.

Tabla 5.1: Número de ajustes de posición promedio y Tiempo promedio de teleoperación - Camino verde (2D)

Modo	Número de ajustes de posición promedio	Tiempo promedio de teleoperación
T_1 Háptico	57,60	194,20[s]
T_3 No Háptico	62,20	227,24[s]
T_2 Háptico	60,80	183,84[s]
T_4 No Háptico	71,00	197,28[s]

En general, se observa (Tabla 5.1) que la teleoperación háptica necesita en promedio una menor cantidad de ajustes del efector, que las teleoperaciones que no utilizan retroalimentación háptica. Comparando los modos *Teleop1* con *Teleop3*, el número de ajustes promedio se reduce en un 7,4 % (no representa una diferencia estadísticamente significativa) cuando está activada la fuerza de retroalimentación. Comparando la *Teleop2* con la *Teleop4*, el número se ve reducido en un 14,4 % con retroalimentación háptica (estadísticamente significativo, ya que $P < 0{,}01$). Como entre *Teleop1* y *Teleop3* no existe una diferencia estadísticamente significativa, no se puede asegurar que la diferencia del número de ajustes del efector esté dada por efectos relacionados con la utilización de la fuerza de retroalimentación háptica, por el contrario, entre *Teleop2* y *Teleop4* sí se observa una diferencia, lo cual permite inferir que el método de *Deltas* de movimientos del efector es más susceptible al uso de la tecnología háptica lo cual influye en la toma de decisiones de la posición del efector.

Al comparar el modo *Teleop1* con el modo *Teleop3* se puede observar que el tiempo en completar la tarea de teleoperación usando retroalimentación háptica se reduce en un 14,5 % con una diferencia estadísticamente significativa ($P < 0{,}1$), y comparando el modo *Teleop2* con *Teleop4*, el tiempo se reduce en un 6,8 % cuando la fuerza de retroalimentación está activada, aunque no resultó ser una diferencia estadísticamente significativa.

Por otro lado, cuando se comparan los modos *Teleop1* con *Teleop2* (ambos hápticos) se observa que se requiere un menor número de ajustes en el primer caso, pero más tiempo

en completar la tarea. De forma similar con el modo *Teleop3* se necesita menor número de ajustes pero más tiempo toma en completar la tarea comparando con el modo *Teleop4*. Por lo tanto, es posible inferir que con el modo "*Deltas* de movimientos del efector" (*Teleop2* y *Teleop4*) los usuarios pueden cambiar la posición del robot real de manera más rápida, alcanzando el punto final objetivo en menos tiempo, pero con una menor precisión (más ajustes, más colisiones y errores promedio más grandes).

5.3. Resultados de Trayectoria 3D

Para los resultados de Trayectoria 3D en el panel de pruebas (camino rojo) se realiza el mismo análisis que para el caso del recorrido sobre el panel de pruebas en 2D (camino verde), utilizando de la misma forma los cuatro modos de teleoperación bilaterial con retraso temporal para que los usuarios completen el recorrido pedido. En la Figura 5.8 se muestran los recorridos realizados por los 20 teleoperadores al completar la tarea de seguimiento de trayectoria (sobre el camino rojo) con el efector del robot utilizando el Modo *Teleop1* y *Teleop3*.

En la Figura 5.8 los puntos azules corresponden al Modo*Teleop3* (visual no háptica) y los puntos negros corresponden al Modo*Teleop1* (modo háptico). Estos últimos se observan más cercanos al panel y con una menor dispersión, lo cual es un indicador (a primera vista) que en promedio las operaciones que utilizan retroalimentación háptica son más precisas. En las siguientes secciones se detallará este análisis.

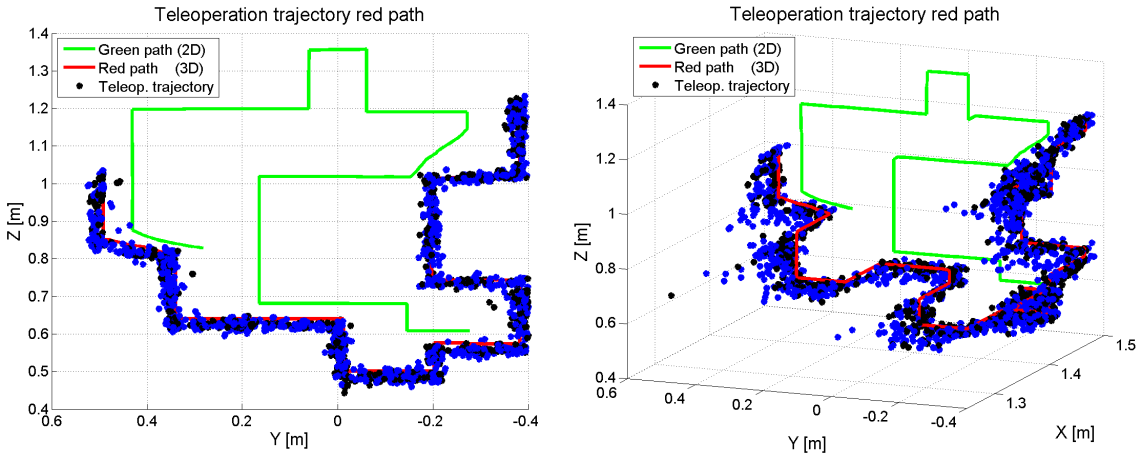

Figura 5.8: Trayectorias de teleoperación - Camino rojo (3D): A la izquierda vista frontal y a la derecha vista isométrica. Los puntos corresponden a las muestras de las trayectorias de teleoperación de los 20 usuarios. Los puntos negros con el Modo *Teleop1* (visuo-háptica) y los puntos azules con el Modo*Teleop3* (visual no háptica).

La dificultad adicional que se presenta en la ruta roja 3D sobre el panel de pruebas, es que los usuarios deben alejarse del plano del panel una distancia aproximada de 10 a 15[*cm*] para pasar por sobre los obstáculos de color azul adheridos al panel. Este es el motivo por el que se denomina "camino 3D" a esta prueba, en donde los teleoperadores se ven en la necesidad

de modificar la distancia de la punta del efector con respecto al plano de pruebas de forma sistemática, para sortear los obstáculos instalados en mitad del camino.

5.3.1. Trayectoria 3D: Error promedio de posición del efector

En la Figura 5.9 se observa un gráfico de barras que representa el error promedio de la posición del efector final, de todos los usuarios, a lo largo de todo el camino 3D sobre el panel de pruebas. Cuando se compara el modo *Teleop1* con *Teleop3* se observa una reducción del error promedio de un 39,7 % cuando se utiliza retroalimentación háptica. Comparando *Teleop2* con *Teleop4*, el error promedio es reducido en un 43,3 % cuando se utiliza la tecnología háptica. La Prueba-T para muestra apareadas revela que las diferencias son estadísticamente significativas ($P < 0{,}01$) en ambos casos. En términos generales esta medida muestra la importancia de la retroalimentación háptica al incrementar la precisión del operador. El modo de teleoperación denominado "*Workspace* proporcional" presenta menor error promedio, al igual que los resultados mostrados para el caso del camino 2D (verde).

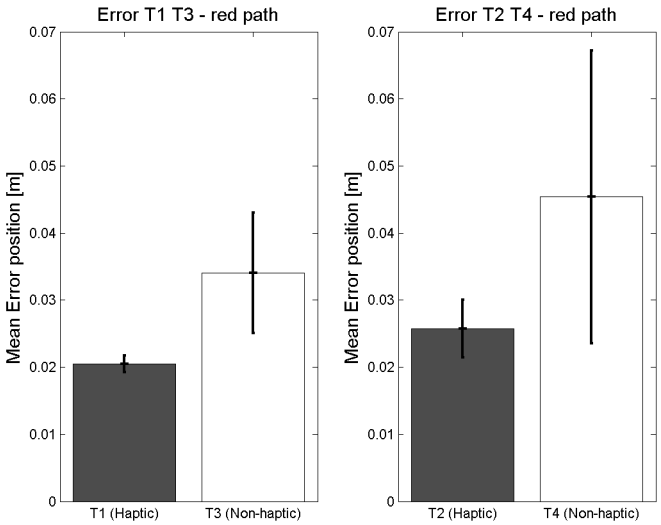

Figura 5.9: Error promedio y desviación estándar de la posición del efector usando los cuatro Modos de teleoperación - Camino rojo (3D).

Los errores promedio obtenidos para las pruebas sobre el camino 3D son bastante similares a los resultados obtenidos para las pruebas sobre el camino 2D. Sin embargo, hay que destacar que los usuarios completaron el recorrido 3D una vez que finalizaron todas las pruebas sobre el recorrido 2D, por lo que ya se presentan con una experiencia significativa al desplazar el efector del robot real sobre la superficie del panel de pruebas.

5.3.2. Trayectoria 3D: Error promedio de posición del efector en función del tiempo

La Figura 5.10 muestra el gráfico del error promedio de la posición del efector del robot real en función del tiempo normalizado, usando el procedimiento descrito en la sección anterior.

En este caso, a diferencia de lo mostrado en la Figura 5.6 (camino verde 2D) no es posible ver una tendencia de disminución del error a medida que se acerca el final del camino que se debe recorrer. Los obstáculos que están adheridos al panel de pruebas obligan a los usuarios a alejarse del plano de operación por lo que deben reajustar la distancia al panel en varias ocasiones, lo cual incrementa la probabilidad de un aumento no uniforme del error de la punta del efector con respecto al recorrido ideal, lo cual se ve reflejado en un gráfico de errores medios en el tiempo bastante irregular, sin alguna tendencia que se pueda destacar.

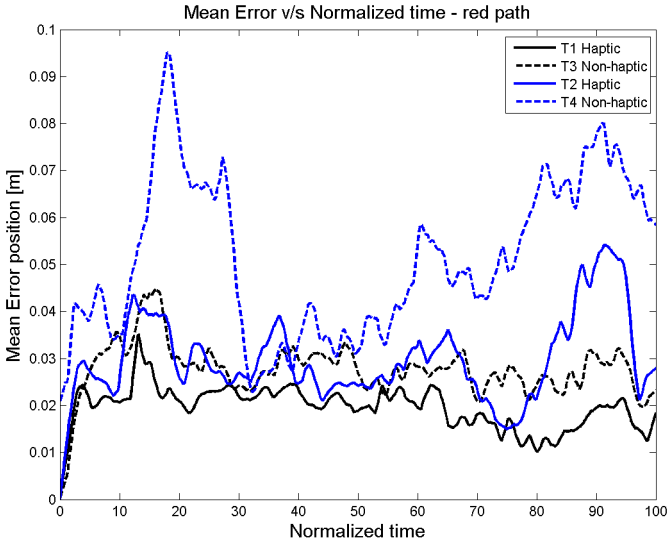

Figura 5.10: Error promedio de posición del efector en función del tiempo normalizado para los cuatro Modos de teleoperación - Camino rojo (3D).

Además la Figura 5.10 muestra que los modos de operación *Teleop4* y *Teleop2* son respectivamente el primer y segundo lugar en las mayores variaciones de error a través del tiempo. Esto significa que el modo de operación "*Deltas* de movimientos del efector" no es el más adecuado cuando se presentan cambios obligados de la distancia entre el efector y el panel de pruebas. En alguna de las pruebas obtenidas para estos modos de operación es posible ver incluso el momento en el cual los usuarios se enfrentan a alguno de los 4 obstáculos presentes en el camino que deben recorrer, ya que el error en esos instantes de tiempo aumenta considerablemente.

5.3.3. Trayectoria 3D: Número de colisiones

En la Figura 5.11 se muestra el número de colisiones débiles y fuertes en promedio registradas para la tarea de teleoperación 3D (camino rojo). Se utiliza el mismo criterio descrito en la sección anterior (camino verde 2D) para contabilizar las colisiones débiles o fuertes.

Comparando *Teleop1* con *Teleop3* el número promedio de colisiones se reduce en un 18,9 % con un valor $P > 0{,}1$ por lo tanto no es un efecto estadísticamente significativo. Comparando *Teleop2* con *Teleop4*, al utilizar retroalimentación háptica se reduce el número de colisiones promedio en un 30,6 % con un valor $P < 0{,}0001$ lo cual representa un efecto estadísticamente

significativo. Se observa en general que usar tecnología háptica reduce el número de colisiones totales, aunque se destaca la reducción del número promedio de colisiones fuertes.

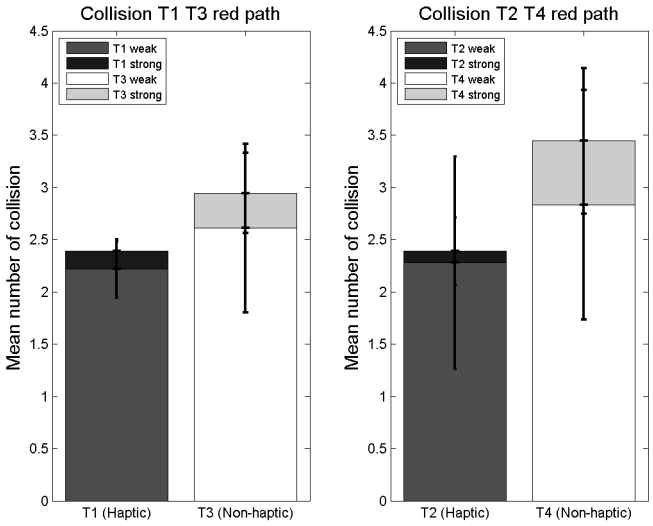

Figura 5.11: Número de colisiones promedio y desviación estándar para los cuatro Modos de teleoperación - Camino rojo (3D). Se clasifican en colisiones débiles y fuertes.

Cuando se realiza la comparación de los modos hápticos (*Teleop1* con *Teleop2*) y los modos no hápticos (*Teleop3* con *Teleop4*) no es posible observar diferencias estadísticamente significativas con respecto al número promedio de colisiones, lo cual significa que no existe un modo de operación que destaque por ser el que tiene menor probabilidad de obtener alguna colisión.

Comparando los resultados de colisiones promedio obtenidos para la ruta roja 3D (Figura 5.11) con los resultados obtenidos para la ruta verde 2D (Figura 5.7) se presenta un aumento significativo del número de colisiones fuertes y por sobre todo, un aumento de colisiones débiles al operar sobre obstáculos en el camino. Este aumento en las colisiones está asociado al reposicionamiento constante del efector del robot real con respecto al plano de la ruta, lo cual provoca que los usuarios tengan que volver a acercarse al plano del panel de pruebas cada vez que un obstáculo nuevo aparece.

5.3.4. Trayectoria 3D: Número de ajustes de posición y Tiempo promedio de teleoperación

La Tabla 5.2 muestra el número de ajustes y tiempo promedio de las pruebas de teleoperación usando los cuatro modos. Al comparar el modo *Teleop1* con el modo *Teleop3* se observa una reducción del 19,5 % en el número promedio de ajustes cuando se utiliza la fuerza retroalimentada al usuario. Comparando el modo *Teleop2* con el modo *Teleop4* el número de ajustes promedio se reduce en un 14,8 % al utilizar el *feedback* háptico. Las diferencias son estadísticamente significativas con un valor $P < 0{,}01$ para ambos casos, lo cual significa que la utilización de la fuerza de retroalimentación provoca un cambio en la estrategia de movimiento que está utilizando el usuario, por lo tanto, influye en el desempeño global de la

tarea. Una reducción en el número de ajustes refleja que los usuarios están más conformes con la posición del efector en cada paso realizado sobre el camino total, comparado con los casos con un mayor número de ajustes.

Además se observa (Tabla 5.2) que la teleoperación háptica toma levemente menos tiempo promedio para completar el recorrido que la teleoperación que no posee dicha retroalimentación. Comparando el modo *Teleop1* con *Teleop3*, el tiempo promedio de teleoperación se reduce en un 13,3 %, y comparando el modo *Teleop2* con el *Teleop4* se reduce en un 9,6 % el tiempo promedio en finalizar la tarea cuando la fuerza háptica está activa, con una diferencia estadísticamente significativa de valor $P < 0{,}1$ en ambos casos.

Tabla 5.2: Número de ajustes de posición promedio y Tiempo promedio de teleoperación - Camino rojo (3D)

Modo	Número de ajustes de posición promedio	Tiempo promedio de teleoperación
T_1 Háptico	68,60	242,16[s]
T_3 No Háptico	85,20	279,28[s]
T_2 Háptico	70,60	212,04[s]
T_4 No Háptico	85,20	234,52[s]

Así como en el caso de la trayectoria 2D (Tabla 5.1), los usuarios terminan de completar el recorrido 3D de manera más rápida utilizando los modos de teleoperación "*Deltas* de movimientos del efector" pero con más ajustes promedio y un error promedio mayor. Esto se debe a que al utilizar los *Deltas* de posición, el usuario puede modificar la dirección de desplazamiento del efector con respecto a cada eje coordenado en forma individual, lo que permite deslizarse paralelo a una superficie que está perpendicular a alguno de estos ejes. Los usuarios utilizan esta ventaja comparativa del modo "*Deltas*" para avanzar una distancia mayor sobre la superficie, pero con la desventaja de que sólo se puede realizar en línea recta. La prueba del camino 3D justamente impide que los operadores puedan deslizarse una gran distancia sobre el plano, ya que se encuentran con una serie de obstáculos que obligan a sacar la punta del efector desde su posición cercana al panel, lo que además explica por qué aumenta el error medio y el número de colisiones total (el reposicionamiento del efector sobre algún plano implica un aumento en los ajustes de ubicación antes de continuar con el recorrido).

5.4. Análisis General

A pesar de que la evaluación experimental dura aproximadamente 40 minutos por cada uno de los usuarios, es posible observar que en los últimos dos o tres modos puestos a prueba los operadores se sienten más cómodos con el sistema de teleoperación. En tan sólo 40 minutos es posible que una persona que nunca ha tenido contacto con un sistema de teleoperación con retardo temporal de este tipo, pueda desenvolverse de manera cómoda tomando la iniciativa para probar nuevas combinaciones de desplazamientos con la punta del efector del robot. El incremento del número de ajustes del efector y simultáneamente la disminución del tiempo necesario para completar la tarea pedida, es una consecuencia del

entrenamiento y la experiencia que los usuarios han adquirido al momento de llegar a las instancias finales de la evaluación experimental. Los usuarios se sienten con más confianza al desplazar el efector del robot real, y además están dispuestos a sacrificar precisión para ganar velocidad y así poder terminar la evaluación en el menor tiempo posible.

En general se observa que la teleoperación que se realiza estrictamente sobre un plano (camino verde) posee una curva de aprendizaje rápida, en cambio cuando se realiza una teleoperación en donde es necesario regular la distancia del efector del robot con respecto al plano de manera repetida (camino rojo) es más difícil que los usuarios se acostumbren (curva de aprendizaje más lenta), debido a que cada obstáculo se encuentra en una posición y ángulo distinto, requiriendo cada uno de ellos más minutos de práctica. Como se le ha dado el mismo tiempo de entrenamiento a cada usuario para el recorrido 2D como para el recorrido 3D, es natural que presenten mayores dificultades en el segundo caso.

6 Conclusiones

En la presente tesis se propuso un novedoso sistema de teleoperación bilateral con retardo temporal basado en la interacción háptica entre una *Point Cloud* y un robot simulado dentro de un entorno de realidad virtual, logrando con éxito la evaluación experimental del sistema y la verificación de la hipótesis planteada al inicio de la tesis. Con este método de teleoperación, los usuarios aumentan la precisión al realizar una tarea de seguimiento de trayectoria con la punta del efector de un robot sobre una superficie en el entorno real, a pesar de que no se tiene acceso directo al lugar de operación (porque está muy lejos o se presentan riesgos para la salud del operador) y a pesar que se presenta un retraso temporal en la comunicación entre el operador (lado maestro) y el robot (lado esclavo) que se encuentra trabajando en la faena (causado por la extrema lejanía del robot esclavo o porque existen problemas de conectividad).

Este estudio demostró que a través de la metodología de teleoperación háptica propuesta, los operadores (usuarios) pueden alcanzar mayor precisión al completar una tarea de seguimiento de trayectoria en un entorno de operación desconocido (antes de acceder a la visualización del entorno virtual, el usuario no sabe cómo es el espacio de trabajo, ni la proporción que tiene el robot comparado con los obstáculos y objetos que hay en dicho entorno). Además, al utilizar la retroalimentación háptica es posible completar la tarea más rápido, con una menor cantidad de ajustes de posición del efector final del robot y con una menor cantidad de colisiones del robot con el entorno.

El modo de teleoperación denominado "*Workspace* proporcional" resultó ser más intuitivo para los usuarios que el modo "*Deltas* de movimientos del efector" ya que se observó menor cantidad de colisiones promedio y un error promedio más bajo cuando se compararon estos modos usando retroalimentación háptica y cuando no se utilizó esta tecnología.

La metodología propuesta puede ser utilizada en sistemas satelitales espaciales, ya que se mejora la precisión al manipular herramientas robóticas desde una gran distancia (desde la estación base de monitoreo en la Tierra hasta el satélite o estación espacial). Estas herramientas son fundamentales para realizar la mantención de dispositivos que están en órbita y de esta forma aumentar su vida útil. Además, este sistema puede ser utilizado en la industria de la minería, reduciendo el tiempo de producción y mejorando la experiencia de los usuarios durante las largas horas de operación. Esta tecnología de retroalimentación háptica es ideal para controlar remotamente martillos picadores de roca, en donde el entorno de trabajo al cual se expone el brazo robótico es irregular y presenta cambios constantemente, por lo cual no es posible tener un modelo *a priori* del entorno.

6.1. Trabajos futuros

Como trabajo futuro se propone extender la interacción del modelo virtual del robot con la nube de puntos del entorno de trabajo. La metodología presentada en la presente tesis considera sólo la utilización del efector del robot como zona de contacto o de interacción con otros elementos del entorno real, pero de esta manera no es posible evitar colisiones de los demás *joints* del robot con el entorno. La interacción extendida del modelo virtual con la *Point Cloud* puede ser llevada a cabo con múltiples esferas *proxy* ubicadas en cada uno de los *joints*, o bien, a través de una serie de restricciones calculadas con la información del sensor 3D, utilizando la metodología planteada en la publicación de A. Leeper et al. [55].

Una aplicación directa de la metodología de teleoperación háptica propuesta es la utilización de un efector que realice una acción física sobre los materiales del entorno de trabajo. Por ejemplo, se puede tener un brazo robótico con un martillo neumático como efector (ver Figura 6.1), incorporando las vibraciones y choques con el material de trabajo como parte de la retroalimentación, mejorando la percepción del operador. En específico se puede utilizar el martillo neumático para fracturar rocas en una operación minera. Por otra parte, el sistema de simulación mostrado puede ser utilizado como plataforma para capacitar a nuevos operarios, reduciendo el tiempo de entrenamiento y mejorando su desempeño al operar el brazo robótico real.

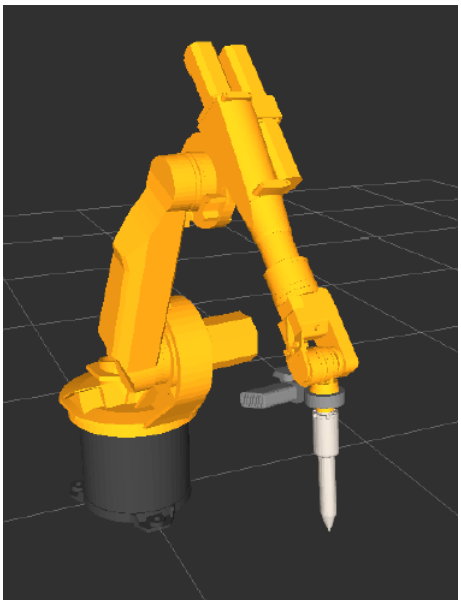

Figura 6.1: Brazo robótico con martillo neumático simulado.

Bibliografía

[1] V. Lumelsky, "On Human Performance in Telerobotics," *IEEE Transactions on Systems*, vol. 21, no. 5, pp. 971–982, September-October 1991.

[2] I. Ivanisevic, and V. Lumelsky, "A Human – Machine Interface for Teleoperation of Arm Manipulators in a Complex Environment," *Intl. Conference on Intelligent Robots and Systems*, pp. 1590–1595, October 1998.

[3] A. Goto, R. Inoue, T. Tezuka, and H. Yoshikawa, "A Research On Tele-operation Using Virtual Reality," *IEEE International Workshop on Robot and Human Communication*, pp. 147–152, July 1995.

[4] K. Kosuge, K. Takeo, T. Fukida, T. Sugiura, A. Sakai, and K. Yamada, "Unified Approach for Teleoperation of Virtual and Real Environment for Skill Based Teleoperation," *Advanced Robotic Systems and the Real World IROS*, vol. 2, pp. 1242–1247, September 1994.

[5] A. Nakai, T. Ohashi, and H. Hashimoto, "7 DOF Arm Type Haptic Interface for Teleoperation and Virtual Reality Systems," *Intl. Conference on Intelligent Robots and Systems*, pp. 1266–1271, October 1998.

[6] E. Duff, C. Caris, A. Bonchis, K. Taylor, C. Gunn, and M. Adcock, "The Development of a Telerobotic Rock Breaker," *Springer Tracts in Advanced Robotics*, vol. 62, pp. 411–420, 2010.

[7] A. Boeing, "A Remotely Operated Robotic Rock Breaker with Collision Avoidance for the Mining Industry," *Proceedings of the 30th International Association for Automation and Robotics in Construction*, pp. 875–884, 2013.

[8] F. Rydén, S. Nia Kosari, and H. Chizeck, "Proxy Method for Fast Haptic Rendering from Time Varying Point Clouds," *Proc. IEEE/ RSJ Int'l Conf. Intelligent Robots and Systems (IROS '12)*, pp. 2614–2619, 2011.

[9] F. Rydén, and H. Chizeck, "A Proxy Method for Real-Time 3-DOF Haptic Rendering of Streaming Point Cloud Data," *IEEE Transactions on Haptics*, vol. 6, no. 3, pp. 257–267, July-September 2013.

[10] D. Caldwell, K. Reddy, O. Kocak, and A. Wardle, "Sensory Requirements and Perfor-

mance Assessment of Tele-Presence Controlled Robots," *Intl. Conference on Robotics and Automation*, pp. 1375–1380, April 1996.

[11] Oxford Dictionary of English - Haptic definition. [Online]. Available: http://www.oxforddictionaries.com/definition/english/haptic

[12] Anderson, Douglas M, *Mosby's Medical, Nursing and Allied Health Dictionary*, Fourth Edition ed. Mosby imprint of Elsevier, 1994, p. 1285.

[13] K. Kyung, J. Lee, and J. Park, "Comparison of Force, Tactile and Vibrotactile Feedback for Texture Representation Using a Combined Haptic Feedback Interface," *Lecture Notes in Computer Science*, vol. 4813, pp. 34–43, 2007.

[14] Receptores sensoriales de la piel. [Online]. Available: http://histologiadecesarfcruzycarloacadme.blogspot.cl/2013/02/receptores-sensoriales-de-la-piel.html

[15] SensAble PHANTOM Omni. [Online]. Available: http://www.quarcservice.com/ReleaseNotes/files/sensable_phantom_omni.html

[16] A. Ollero Baturone, *Robótica: Manipuladores y Robots Móviles*, Primera Edición ed. Marcombo, 2001.

[17] OSHA Technical Manual. [Online]. Available: https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html

[18] L. Shapiro and G. Stockman, *Computer Vision.* Prentice Hall, 2001.

[19] J. Kramer, M. Parker, N. Burrus, *Hacking the Kinect.* Springer Verlag GmbH, 2012.

[20] J. Foley, A. van Dam, J. Hughes, S. Feiner, *Spatial-partitioning representations, Surface detail.* Addison-Wesley, 1990.

[21] Imagen Voxel grid. [Online]. Available: http://www.frontiersin.org/files/Articles/65786/fncom-07-00160-r2/image_m/fncom-07-00160-g001.jpg

[22] D. Meagher, *Octree Encoding - A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer.* Rensselaer Polytechnic Institute, 1980.

[23] Imagen Octree. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/3/35/Octree2.png

[24] R. Paul, *Robot manipulators: mathematics, programming, and control : the computer control of robot manipulators.* MIT Press, Cambridge, 1981.

[25] Ejemplo Forward Kinematics. [Online]. Available: https://www.cs.princeton.edu/courses/archive/fall00/cs426/lectures/kinematics/img004.gif

[26] Ejemplo Inverse Kinematics. [Online]. Available: http://www.cs.princeton.edu/courses/

archive/fall99/cs426/lectures/kinematics/img008.gif

[27] J. Fisher Box, "Guinness, Gosset, Fisher, and Small Samples," *Statistical Science*, vol. 2, p. 45–52, 1987.

[28] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," *Proc. Open-Source Software workshop of the Intl. Conference on Robotics and Automation (ICRA)*, 2009.

[29] J. Du, Y. Wang, C. Yang, Y H. Wang, "Hardware-in-the-loop simulation approach to testing controller of sequential turbocharging system," *Proceedings of the IEEE International Conference on Automation and Logistics*, 2007.

[30] *Gazebo*: Robot simulation made easy. [Online]. Available: http://gazebosim.org/

[31] J. Park, and O. Khatib, "A Haptic Teleoperation Approach Based on Contact Force Control," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 575–591, May-June 2006.

[32] A. Ugural and S. Fenster, *Advanced Strength and Applied Elasticity*, 4th ed. Prentice Hall, 2003.

[33] P. Chotiprayanakul, and D. Liu, "Workspace Mapping and Force Control for Small Haptic Device based Robot Teleoperation," *Intl. Conference on Information and Automation*, pp. 1613–1618, June 2009.

[34] M. Mamdouh, and A. Ramadan, "Development of a Teleoperation System With a New Workspace Spanning Technique," *Intl. Conference on Robotics and Biomimetics*, pp. 1570–1575, December 2012.

[35] T. B. Sheridan, "Space teleoperation through time delay: review and prognosis," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 592–606, 1993.

[36] National Aeronautics and Space Administration - Goddard Space Flight Center. Satellite Servicing Capabilities Office. [Online]. Available: http://ssco.gsfc.nasa.gov/

[37] R. Kraft and D. Washington, "NASA's refueling demonstration proves viability of satellite-servicing technologies," NASA, NASA press release 13-046, 2013.

[38] C. Joppin and D.E. Hastings, "On-orbit upgrade and repair: The Hubble Space Telescope example," *Journal of Spacecraft and Rockets*, vol. 43, no. 3, pp. 614–625, 2006.

[39] L.J. Lanzerotti, "Assessment of options for extending the life of the Hubble Space Telescope: Final Report," National Academy Press, 2005.

[40] E. Stoll, J. Letschnik, U. Walter, J. Artigas, P. Kremer, C. Preusche, and G. Hirzinger, "On-orbit servicing," *IEEE Robotics and Automation Magazine*, vol. 16, no. 4, pp. 29–33, 2009.

[41] S. Vozar, S. Leonard, P. Kazanzides, and L. Whitcomb, "Experimental Evaluation of Force Control for Virtual-Fixture-Assisted Teleoperation for On-Orbit Manipulation of Satellite Thermal Blanket Insulation," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4424–4431, May 2015.

[42] T. Xia, S. Leonard, I. Kandaswamy, A. Blank, L. Whitcomb, and P. Kazanzides, "Model-Based Telerobotic Control with Virtual Fixtures For Satellite Servicing Tasks," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1479–1484, May 2013.

[43] T. Xia, S. Leonard, A. Deguet, L. Whitcomb, and P. Kazanzides, "Augmented Reality Environment with Virtual Fixtures for Robotic Telemanipulation in Space," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5059–5064, October 2012.

[44] J. Zainan, L. Hong, W. Jie, and H. Jianbin, "Virtual Reality-based Teleoperation with Robustness Against Modeling Errors," *Chinese Journal of Aeronautics*, vol. 22, no. 3, pp. 325–333, June 2009.

[45] Z. Jiang, Y. Liu, H. Liu, and J Zou, "Flexible Virtual Fixture Enhanced by Vision and Haptics for Unstructured Environment Teleoperation," *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2643–2648, December 2013.

[46] Z. Boroujeni, M. Mohammadi, and A. Jalali, "Stable Adaptive Time-Variable Delayed Bilateral Teleoperation for a Surgery Robot," *RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pp. 301–306, February 2013.

[47] S. Hirche, and M. Buss, "Human-Oriented Control for Haptic Teleoperation," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 623–647, March 2012.

[48] P. Griffiths, and R. Gillespie, "Shared control between human and machine: haptic display of automation during manual control of vehicle heading," *Intl. Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 358–366, March 2004.

[49] J. van Oosterhout, J. Wildenbeest, H. Boessenkool, C. Heemskerk, M. de Baar, F. var der Helm, and D. Abbink, "Haptic Shared Control in Tele-Manipulation: Effects of Inaccuracies in Guidance on Task Execution," *IEEE Transactions on Haptics*, vol. 8, no. 2, pp. 164–175, April-June 2015.

[50] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *Intl. Journal of Robotics Research*, vol. 5, no. 1, 1986.

[51] Coursera: Robotics Motion Planning, Constructing Artificial Potential Fields. [Online]. Available: https://es.coursera.org/learn/robotics-motion-planning/lecture/J9sC0/4-1-constructing-artificial-potential-fields

[52] L. Kavraki, and J. Latombe, "Randomized preprocessing of Configuration Space for path Planning: articulated robots," *Intl. Conference on Intelligent Robots and Systems*, vol. 3, pp. 1764–1771, 1994.

[53] Coursera: Robotics Motion Planning, Introduction to Configuration Space. [Online]. Available: https://es.coursera.org/learn/robotics-motion-planning/lecture/0auId/2-1-introduction-to-configuration-space

[54] P. Mitra and G. Niemeyer, "Haptic Simulation of Manipulator Collisions Using Dynamic Proxies," *Presence: Teleoperators and Virtual Environments*, vol. 16, no. 4, pp. 367–384, August 2007.

[55] A. Leeper, K. Hsiao, M. Ciocarlie, I. Sucan, and K. Salisbury, "Methods for Collision-Free Arm Teleoperation in Clutter Using Constraints from 3D Sensor Data," *Intl. Conference on Humanoid Robots*, pp. 520–527, October 2013.

[56] N. El-Far, N. Georganas, and A. El Saddik, "An Algorithm for Haptically Rendering Objects Described by Point Clouds," *Canadian Conference on Electrical and Computer Engineering*, pp. 1443–1448, May 2008.

[57] A. Leeper, S. Chan, and K. Salisbury, "Point Clouds Can Be Represented as Implicit Surfaces for Constraint-Based Haptic Rendering," *Intl. Conference on Robotics and Automation*, pp. 5000–5005, May 2012.

[58] J. Wildenbeest, D. Abbink, C. Heemskerk, F. van der Helm, and H. Boessenkool, "The Impact of Haptic Feedback Quality on the Performance of Teleoperated Assembly Tasks," *IEEE Transactions on Haptics*, vol. 6, no. 2, pp. 242–252, April-June 2013.

[59] V. Nitsch, and B. Färber, "A Meta-Analysis of the Effects of Haptic Interfaces on Task Performance with Teleoperation Systems," *IEEE Transactions on Haptics*, vol. 6, no. 4, pp. 387–398, October-December 2013.

[60] Z. Ju, C. Yang, Z. Li, L. Cheng, and H. Ma, "Teleoperation of Humanoid Baxter Robot Using Haptic Feedback," *Intl. Conference on Multisensor Fusion and Information Integration for Intelligent Systems*, pp. 1–6, September 2014.

[61] A. Talasaz, A. Trejos, and R. Patel, "Effect of Force Feedback on Performance of Robotics-Assisted Suturing," *Intl. Conference on Biomedical Robotics and Biomechatronics*, pp. 823–828, June 2012.

[62] I. Sarakoglou, N. Garcia-Hernandez, N. Tsagarakis, and D. Caldwell, "A High Performance Tactile Feedback Display and Its Integration in Teleoperation," *IEEE Transactions on Haptics*, vol. 5, no. 3, pp. 252–263, July-September 2012.

[63] H. Boessenkool, D. Abbink, C. Heemskerk, F. van der Helm, and J. Wildenbeest, "A Task-Specific Analysis of the Benefit of Haptic Shared Control During Telemanipulation," *IEEE Transactions on Haptics*, vol. 6, no. 1, pp. 2–12, January-March 2013.

[64] Geomagic. (2010) Phantom omni haptic device. [Online]. Available: http://www.dentsable.com/haptic-phantom-omni.htm

[65] Ioan A. Sucan and Sachin Chitta. MoveIt! [Online]. Available: http://moveit.ros.org

[66] R. Rusu, and S. Cousins, "3D is here: Point Cloud Library (PCL)," *IEEE Intl. Conference on Robotics and Automation (ICRA)*, May 2011.

[67] C. Basdogan and M. Srinivasan, *Haptic rendering in virtual environments. Handbook of virtual environments*, 2002.

[68] KUKA-robotics. (2010) Kuka kr 6-2 description. [Online]. Available: http://www.kuka-robotics.com/es/products/industrial_robots/low/kr6_2/start.htm

[69] P. Chavent. (2003, September) Rapport de stage de deuxième année. [Online]. Available: http://paul.chavent.free.fr/kuka/doc/rapport.pdf

[70] P. Chavent. (2003, September) API Kuka Manuel du devellopeur Version 0.0.5. [Online]. Available: http://paul.chavent.free.fr/kuka/doc/kuka_api_man_devl.pdf

[71] Sensable. (2010) Openhaptics toolkit. [Online]. Available: http://www.dentsable.com/products-openhaptics-toolkit.htm

[72] X. Xu, B. Cizmeci, A. Al-Nuaimi, and E. Steinbach, "Point Cloud-Based Model-Mediated Teleoperation With Dynamic and Perception-Based Model Updating," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 11, pp. 2558–2569, November 2014.

[73] F. Rydén, A. Stewart, and H. Chizeck, "Advanced telerobotic underwater manipulation using virtual fixtures and haptic rendering," *Oceans - San Diego 2013*, pp. 1–8, September 2013.

[74] P. Mitra, and G. Niemeyer, "Model-mediated telemanipulation," *Int. Journal of Robotics Research*, vol. 27, no. 2, p. 253–262, February 2008.

[75] W. Book, L. Love, and Shimon Y. Nof, *Handbook of Industrial Robotics*. Wiley, 1985, ch. 9 - Teleoperation, Telerobotics, and Telepresence.

7 Anexos

7.1. Mapa de Nodos *ROS*

En *ROS* existe la herramienta llamada *rqt graph* que proporciona una representación gráfica de la conexión existente entre los nodos, a través de las publicaciones y suscripciones de tópicos. En la Figura 7.2 se muestra el mapa general de nodos y tópicos correspondiente al proyecto presentado en esta tesis. En las Figuras 7.3, 7.4 y 7.5, se muestra en detalle los principales nexos entre los bloques *omni1*, *point cloud server* y *tf* respectivamente.

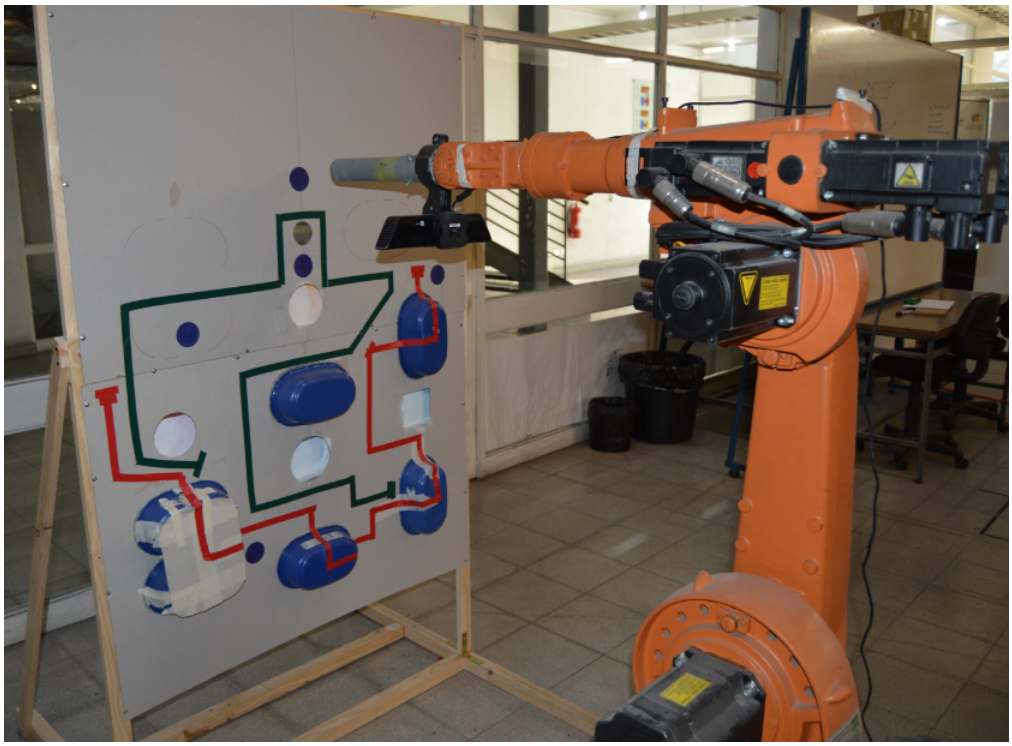

Figura 7.1: *KUKA workspace.*

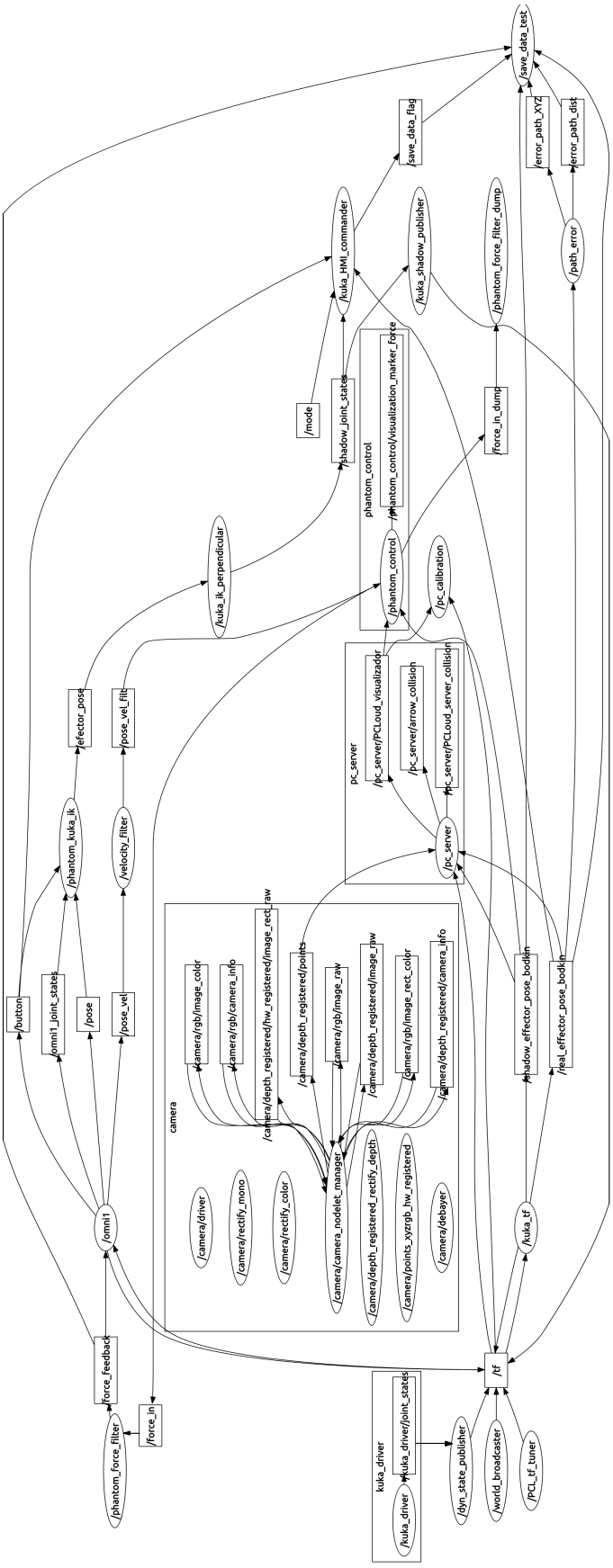

Figura 7.2: Mapa de nodos y tópicos de *ROS* completo.

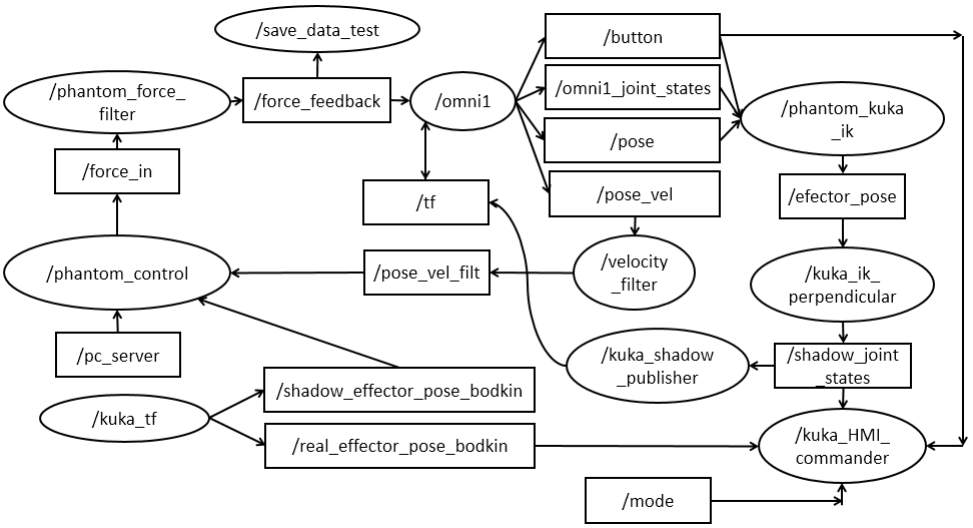

Figura 7.3: Mapa de los principales nodos y tópicos de *ROS* en torno a *omni1*.

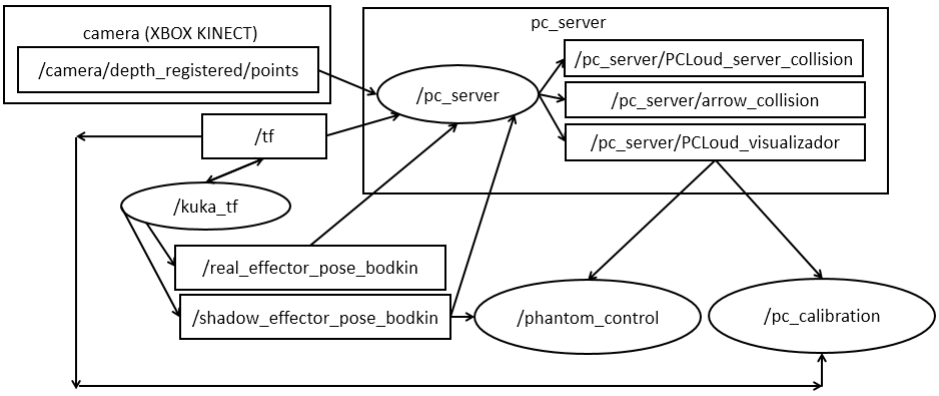

Figura 7.4: Mapa de los principales nodos y tópicos de *ROS* en torno a *point cloud server* (*pc server*).

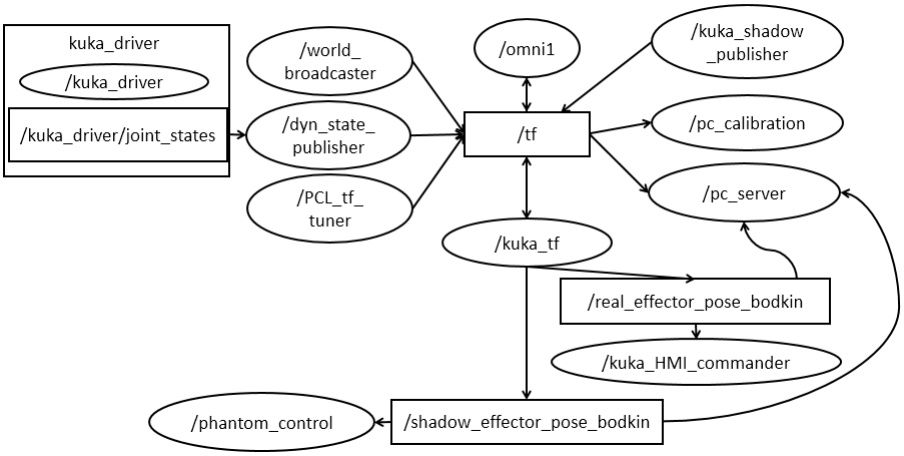

Figura 7.5: Mapa de los principales nodos y tópicos de *ROS* en torno a *tf*.

7.2. Instrucciones de encendido robot *KUKA KR 6-2* y ejecución del programa de teleoperación

El gabinete (Figura 7.6) del brazo robótico *KUKA* contiene los controladores PLC, un computador, las interfaces de comunicación con el robot, y el sistema de alimentación que cuenta con una UPS que suministra de energía al computador interno en caso de emergencias. El gabinete se debe conectar a la red trifásica.

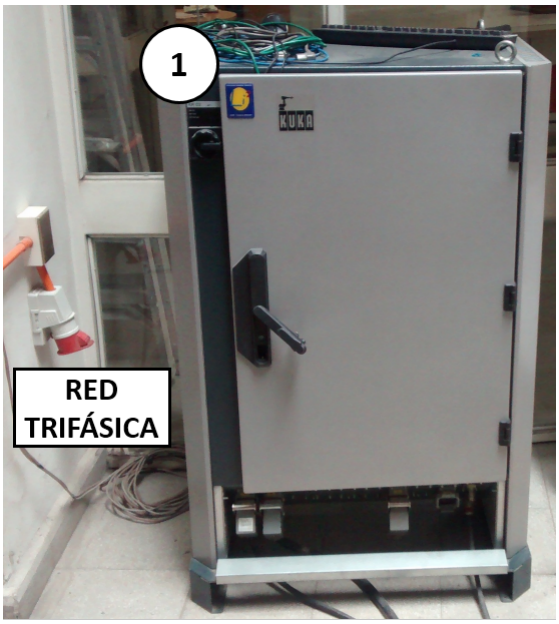

Figura 7.6: Gabinete del robot *KUKA*.

A continuación se detallarán los pasos necesarios para encender el robot *KUKA* y ejecutar el programa de comunicación desarrollado para realizar la teleoperación.

PASO 1: Verificar conexiones.

El primer paso (ver Figura 7.7) es conectar el computador interno del *KUKA* a la red local (LAN) a través de un cable de red (**Punto 2** en Figura 7.7 correspondiente a la **tarjeta de red PCI**) y además alimentar la fuente de poder del computador (**Punto 3** en Figura 7.7). El **Punto 1** corresponde al SWITCH general de encendido del robot, y el **Punto 4** corresponde al SWITCH de encendido del computador interno.

Por otra parte, se debe conectar el cable de alimentación de la *Kinect* (**Punto 5** en Figura 7.8) a un enchufe de alimentación monofásica tradicional y el cable de comunicación USB (**Punto 6** en Figura 7.8) a un puerto USB disponible del computador cliente, en donde se realizará la teleoperación. El dispositivo *Phantom Omni* debe estar conectado (a través del puerto *Firewire*) al computador cliente, además de estar conectado a un enchufe monofásico. Una vez que se haya conectado todos los dispositivos, se continua con el proceso de encendido.

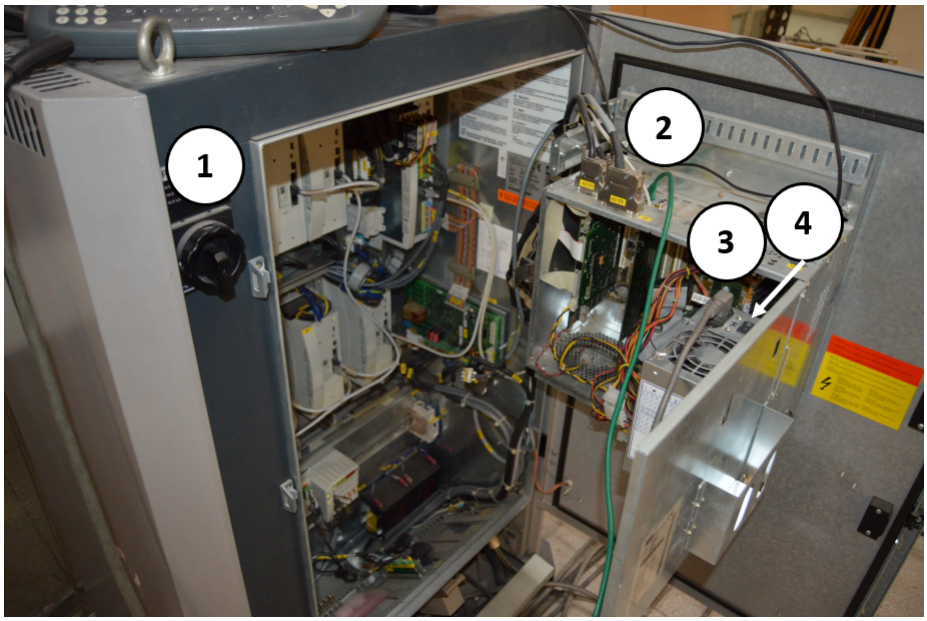

Figura 7.7: Conexiones en gabinete del robot *KUKA*.

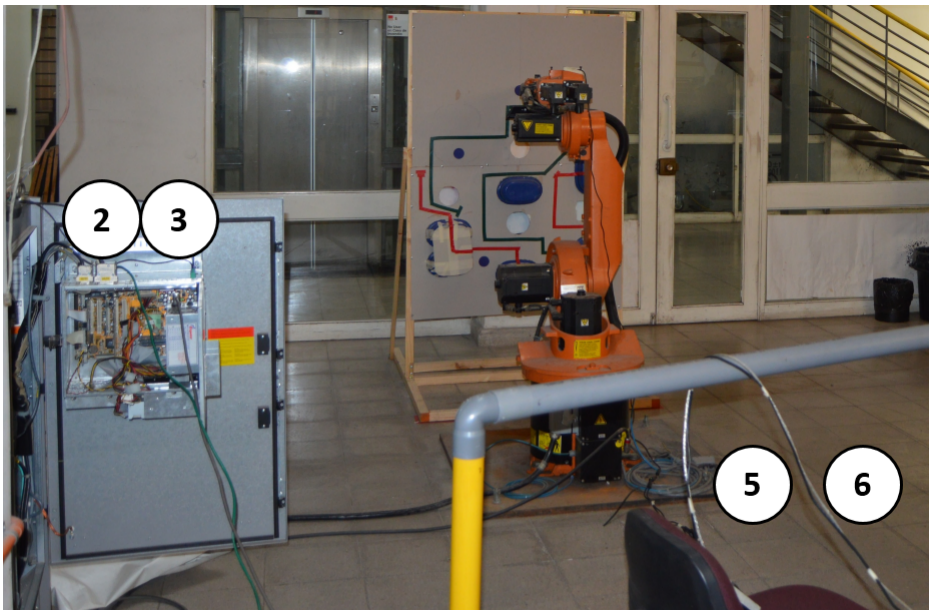

Figura 7.8: Conexiones en gabinete del robot *KUKA* y *Kinect*.

PASO 2: Encendido *KUKA*

Girar el SWITCH de encendido general del robot. Corresponde al **Punto 1** en las Figuras 7.6, 7.7, y 7.9.

Figura 7.9: Switch encendido *KUKA*.

PASO 3: Encendido computador *KUKA* y cliente

Se debe presionar el SWITCH del computador interno del *KUKA*, correspondiente al **Punto 4** de la Figura 7.7 y luego esperar hasta que se inicie *Windows 95*. También se debe encender el computador cliente.

PASO 4: Iniciar programa *CROSS* y *KUKA-API*

Doble clic en el ícono de "*CROSS*" (sistema operativo de *KUKA*) presente en el escritorio del computador del robot. Esperar de 3 a 5 minutos que inicie completamente.

En el menú superior seleccionar **CONFIGURAR**. Luego seleccionar opción 5 **GRUPO USUARIO**. Seleccionar **EXPERTO**. A continuación se pide la **CONTRASEÑA** que es: "kuka".

Presionar el botón de "*windows*" en el teclado externo conectado al computador del robot, para volver al escritorio de *Windows 95*.

Ir a **INICIO**, **PROGRAMAS**, **KUKA_API**, y abrir el programa **KUKA_API**. Se abrirá una ventada de *MS-DOS* que mostrará el mensaje *DONE* cuando se haya finalizado la conexión.

Finalmente se vuelve al programa *CROSS* abierto anteriormente.

PASO 5: Iniciar programa de instrucciones *KUKA_ROS*

En el control de mando del *KUKA* (ver Figura 7.10) se encuentra el *MODE SELECTOR SWITCH* representado por el **Punto 7**. Seleccionar modo **AUTOMATIC** tal como lo indica la Figura 7.11.

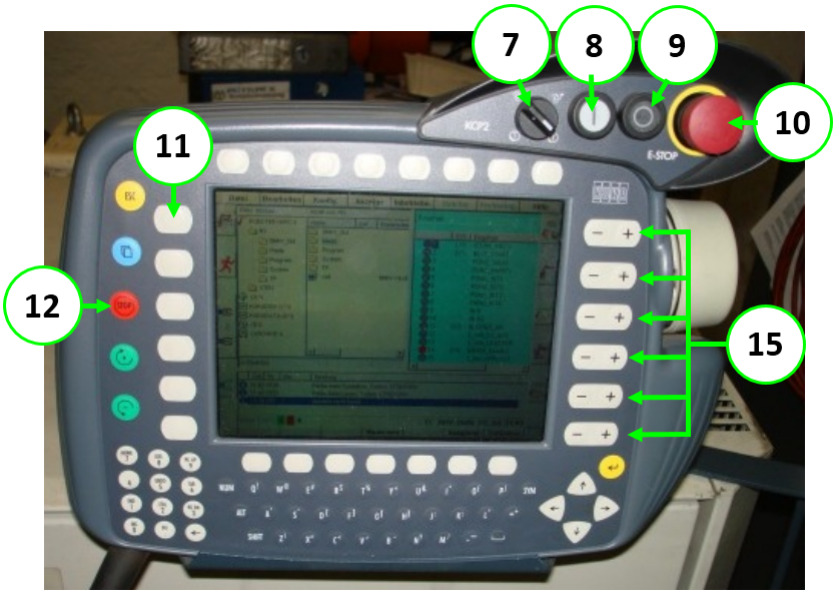

Figura 7.10: Control de mando *KUKA*.

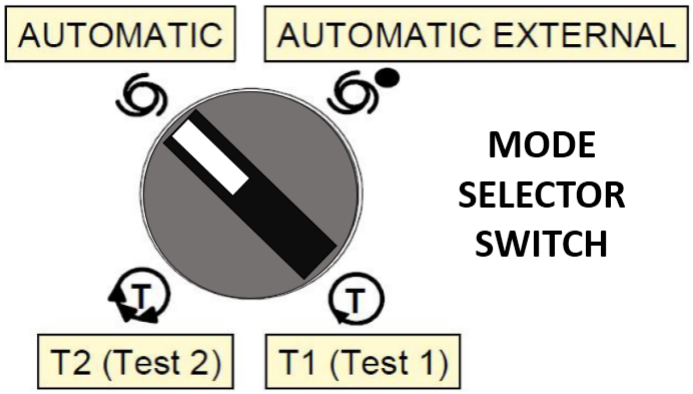

Figura 7.11: *Mode Selector SWITCH* - Automatic.

Luego, utilizando la pantalla del control de mando del *KUKA* (ver Figura 7.10) seleccionar la opción **PROGRAM**, y seleccionar la ruta en donde está el archivo que realiza la conexión para teleoperación: **KUKA_ROS** y finalmente **KUKA_ROS.SRC**. Ahora en la pantalla del control de mando del robot, debe aparecer el código escrito en *KRL (KUKA Robot Language)*, con un puntero de flecha apuntando en la primera línea de código.

Después se necesita "activar los accionamientos", para lo cual se presiona el botón blanco llamado **DRIVES ON** marcado con el **Punto 8** en la Figura 7.12.

Finalmente se debe mantener presionado el botón de color verde que está en la zona

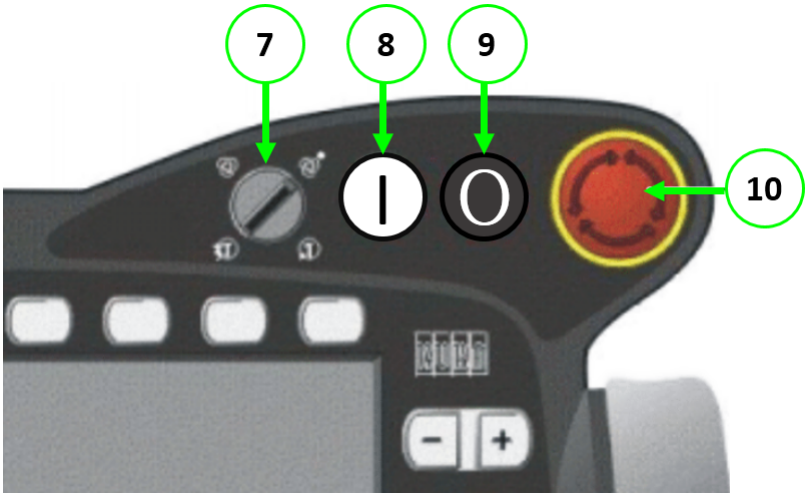

Figura 7.12: Esquema botones principales del control de mando.

posterior del control de mando (ver **Punto 13** de la Figura 7.13) hasta que la flecha indicadora en el programa *KRL* avance hasta el inicio del *loop* principal. Luego se presiona nuevamente el mismo botón verde para que la flecha ahora aparezca desplazándose de arriba a abajo dentro del *loop* principal, lo cual significa que el programa ya está leyendo los mensajes TCP/IP que llegan desde el computador cliente.

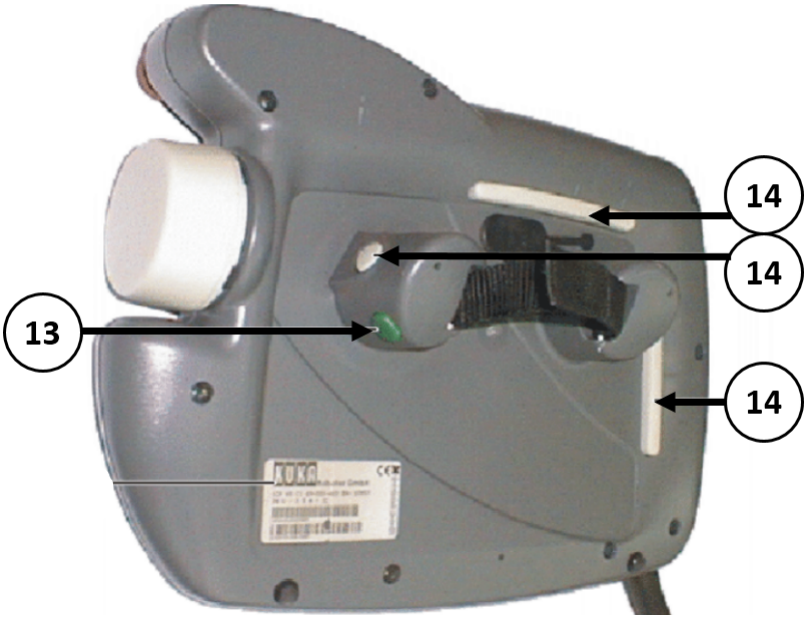

Figura 7.13: Vista posterior del control de mando.

Ahora el computador del robot está listo para recibir instrucciones desde el computador cliente, donde se debe lanzar en primer lugar el **KUKA_DRIVER** a través de los comandos mostrados en la sección Código fuente 7.6, que incluyen un ejemplo simple de ejecución de movimientos.

7.3. Instrucciones de movimiento manual robot *KUKA KR 6-2*

Una vez que el robot *KUKA* se encuentra encendido y con el programa *CROSS* abierto, se debe colocar el **Mode Selector SWITCH** en la posición **T1 (Test 1)** como lo indica la Figura 7.14.

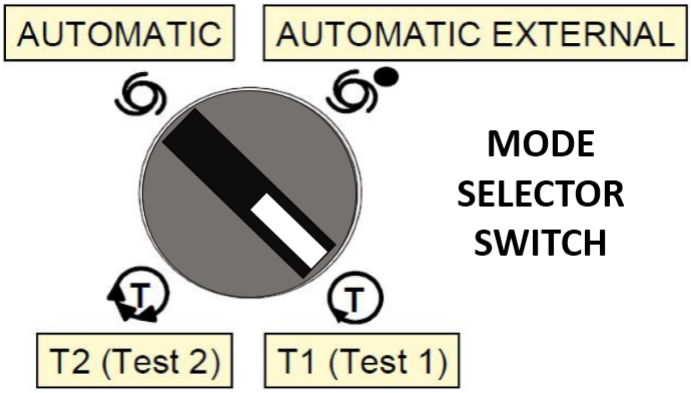

Figura 7.14: *Mode Selector SWITCH* - T1 (Test 1).

Luego, usando el primer botón de selección que está al lado izquierdo de la pantalla del control de mando (**Punto 11** Figura 7.10), se selecciona el modo de operación manual, que corresponde al primer ícono a la izquierda de la Figura 7.15.

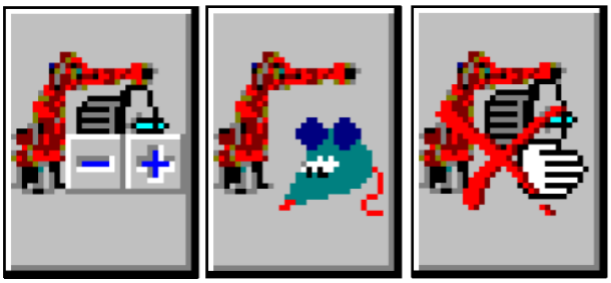

Figura 7.15: Opciones de operación manual *KUKA*.

Después, hay que mantener presionado (hasta la mitad) uno de los 3 botones blancos (1 redondo y 2 alargados) presentes en la parte posterior del control de mando, correspondiente a los **Puntos 14** de la Figura 7.13. Se deben presionar hasta la mitad, ya que si se presiona completamente la función de movimiento manual se bloquea. Este sistema está diseñado para que el operador manual del robot esté atento a cada movimiento del robot, y en caso de una distracción, momento de tensión o accidente, el botón blanco se presionará en forma débil o bien demasiado fuerte, bloqueando el movimiento del robot.

Manteniendo presionado hasta la mitad uno de los tres botones blancos mencionados anteriormente, se deben presionar simultáneamente los botones con símbolos + o − presentes al lado derecho de la pantalla del control de mando (**Puntos 15** de la Figura 7.10). Cada uno de los seis pares de botones ordenados verticalmente, representa a cada *joint* del robot

KUKA, realizando movimientos de adición (+) o sustracción (−) en torno al ángulo cero definido previamente como *home*.

7.4. Masterización robot *KUKA KR 6-2*

Para realizar la masterización, se debe colocar el *Mode Selector SWITCH* en la opción *T2 (Test 2)*, tal como aparece en la Figura 7.16.

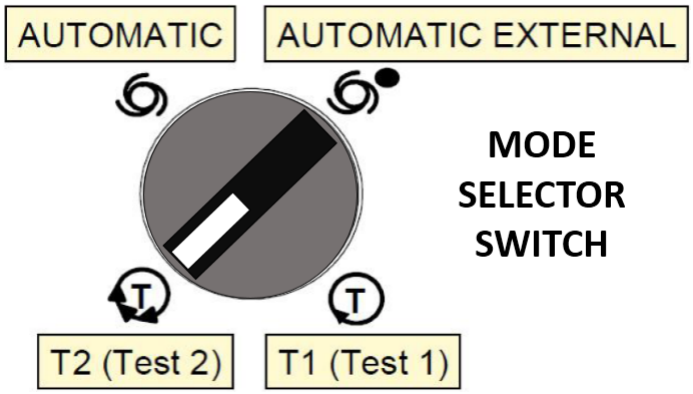

Figura 7.16: *Mode Selector SWITCH* - T2 (Test 2).

Después se debe presionar la opción **INICIALIZACIÓN**, luego presionar **1 AJUSTAR** y finalmente **0 COMPARADOR**. Para desplazar los *joints*, se deben seguir los pasos de la sección anterior "Instrucciones de movimiento manual robot *KUKA KR 6-2*".

7.5. Errores frecuentes en robot *KUKA KR 6-2*

Comandos Activos Bloqueados

Este error se produce cuando alguno de los *joint* del robot sale de la zona de seguridad, es decir, alcanza una posición peligrosa ya sea porque está llegando al límite de su rango de movimiento, o porque podría ocasionar una colisión entre alguna parte del robot y el suelo por ejemplo. Como consecuencia, todos los movimientos del robot se bloquean como medida de seguridad ante un riesgo de colisión.

Para poder salir de este bloqueo general del robot, primero se debe poner el modo *T1 (Test 1)* (ver Figura 7.14). Después ir al menú **CONFIGURAR**, **9 EXTRAS**, **4 WORKSPACE**, y presionar la opción **0 PUNTEAR VIGILANCIA DE ZONA DE TRABAJO**. Enseguida aparecerá el mensaje "*M_147*".

Posteriormente se mueve el el modo manual el robot (ver sección "Instrucciones de movimiento manual robot *KUKA KR 6-2*") hasta una posición que esté dentro de los límites de movimiento de cada *joint* (posición segura).

Finalmente se coloca en **AUTOMATIC** el *Mode Selector SWITCH* (Figura 7.11), para continuar con los pasos necesarios para ejecutar el programa "*KUKA_ROS.SRC*" y proceder con la teleoperación de manera normal.

Valor Fuera de Rango

Se detiene la ejecución del programa actual presionando el botón **STOP** de color rojo ubicado al costado izquierdo del control de mando, representado por el **Punto 12** en la Figura 7.10.

Luego se presiona el botón **PROCESAR** del menú superior, y después **7 CANCELAR PROGRAMA**.

Finalmente se repite el procedimiento para ejecutar el programa nuevamente.

7.6. Instrucciones de apagado robot *KUKA KR 6-2*

Cuando un programa está siendo ejecutado, el primer paso es detener su operación presionando el botón **STOP** de color rojo ubicado al costado izquierdo del control de mando, representado por el **Punto 12** en la Figura 7.10.

Luego se presiona el botón **PROCESAR** del menú superior, y después **7 CANCELAR PROGRAMA**.

Nuevamente se va al menú superior, pero ahora se presiona **CONFIGURAR**, **8 OPCIONES CONEXIÓN - DESCONEXIÓN**, y se escoge la opción **0 ARRANQUE EN FRÍO SELECCIONADO**.

Se apaga el SWITCH general del *KUKA* (Figura 7.9) y se espera hasta que se realice el proceso de apagado automático de *Windows 95* que demora aproximadamente entre 1 y 3 minutos.

Finalmente se apaga el SWITCH de la fuente de poder del computador del robot (**Punto 4** de la Figura 7.7).

7.7. Error de conexión *Phantom Omni*

El orden para conectar correctamente el dispositivo *Phantom Omni* es el siguiente:

1. Conectar el cable *Firewire* al puerto del *Phantom Omni*.

2. Conectar el cable *Firewire* al puerto del computador cliente.

3. Conectar el cable de alimentación al *Phantom Omni*.

4. Conectar el cable de alimentación a la red.

5. Esperar 1 minuto y ejecutar el programa o nodo controlador.

Siguiendo los pasos anteriores de forma estricta no se deberían presentar problemas de conexión. De todas maneras, al iniciar el *driver* se podría mostrar el mensaje "*Failed to initialize haptic device*". Para solucionar esto se vuelve a configurar las opciones del controlador original del *Phantom Omni* y luego se prueba que el dispositivo esté correctamente calibrado, usando los siguientes comandos:

Código fuente 7.1: Configuración *Phantom Omni*.

```
1 $ LANG=en_us /usr/sbin/PHANTOMConfiguration
2 $ LANG=en_us /usr/sbin/PHANTOMTest
```

7.8. Integración de *Kinect* en *ROS*

Se utiliza el driver para *Kinect* denominado *Freenect camera* basado en *OpenNi* de *OpenKinect* (implementado en *ROS*) el cual provee toda la información necesaria para hacer uso del sistema sensorial de profundidad y cámara RGB.

El driver posee una API que tiene un nodo llamado *freenect node* el cual contiene los tópicos, servicios y parámetros de *ROS* necesarios para la correcta obtención de las *Point Cloud* a través del *Kinect*. Este *package* de *ROS* tiene implementado un *tópico* llamado *camera/ depth_ registered/ raw* que publica la información de la nube de puntos detectada, junto con los datos de colores entregados por la cámara RGB, por lo que cada punto lleva asociado una coordenada (x, y, z) de posición y un color.

Para lanzar *freenect* se debe escribir el Comando 7.2.

Código fuente 7.2: Lanzar *freenect*

```
1 $ roslaunch freenect_launch freenect-registered-XYZ.launch
```

7.9. Paquetes y códigos fuente

A continuación se presenta la Tabla 7.1 con los paquetes de *ROS* desarrollados en el presente proyecto.

Tabla 7.1: Paquetes de *ROS* desarrollados.

Paquete	**Descripción**
jackhammer_model	Modelo de ejemplo para aplicaciones futuras
jackhammer_moveit	Configuración cinemática inversa de *jackhammer model.*
kinect_kuka	Integración del *Kinect* al desarrollo.
kuka_bringup	*Launch* final (definitivo) del proyecto.
kuka_control	Librería de control para el *KUKA* (versión final).
kuka_driver	Librerías de comunicación con el *KUKA* real.
kuka_gazebo	*Parsing* de trayectoria para *Gazebo.*
kuka_gazebo_plugins	Controladores para *Gazebo.*
kuka_model	Modelo del robot *KUKA.*
kuka_moveit	Configuración cinematica inversa
octo_server	Manejo de *Point Clouds.*
phantom_kuka	Algoritmos de retroalimentación háptica
phantom_omni	Driver del dispositivo *Phantom Omni.*
ros_tf_tuner	Códigos para *tf* dinámicos para pruebas.
rqt_param_gui	*GUI (Graphical User Interface)* final del proyecto
uchile_kuka	Librería de control (no definitiva) y máquina de estado.

Código fuente 7.3: Código *KRL (KUKA Robot Language)* de movimiento *PTP (Point To Point)* del robot *KUKA.*

```
SWITCH API_MOVE_ID
   CASE 1
      API_MOVE_ID=0
      API_MOVE_FLAG=TRUE
      PTP API_MOVE_AXIS
      API_MOVE_FLAG=FALSE
ENDSWITCH
```

Código fuente 7.4: Variables *KRL (KUKA Robot Language)* de uso libre. Estas variables se encuentran en el archivo *config.dat* y se pueden emplear cuando se requiera tener una variable que se pueda cambiar de forma remota

```
;Real
DECL REAL API_R_1
DECL REAL API_R_2
DECL REAL API_R_3
DECL REAL API_R_4
DECL REAL API_R_5
```

```
;Integer
DECL INT API_I_1
DECL INT API_I_2
DECL INT API_I_3
DECL INT API_I_4
DECL INT API_I_5
;Bool
DECL BOOL API_B_1
DECL BOOL API_B_2
DECL BOOL API_B_3
DECL BOOL API_B_4
DECL BOOL API_B_5
;POS
DECL POS API_P_1
DECL POS API_P_2
DECL POS API_P_3
DECL POS API_P_4
DECL POS API_P_5
;Axis
DECL AXIS API_A_1
DECL AXIS API_A_2
DECL AXIS API_A_3
DECL AXIS API_A_4
DECL AXIS API_A_5
```

Código fuente 7.5: *Script open_ door.py*. Ejecuta secuencia de movimientos para que el efector del robot pueda accionar el botón de apertura de puerta del laboratorio. ¡Atención!: Al crear un nuevo *script* de *python* se deben dar los permisos de ejecución correspondientes en el sistema operativo.

```
import rospy
from kuka_driver.kuka_commander import KukaCommander

if __name__=='__main__':
  rospy.init_node('kuka_commander')
  kuka = KukaCommander()
  kuka.set_vel(80)
  kuka.home()
  rospy.sleep(0.5)
  kuka.ptp([0.10772384992907114, 0.7154650789473394,
     -0.8016527609008506, -6.087224938771763e-06,
     1.0266153515233172e-05, 6.821469104602556e-05]) #OPEN DOOR
  rospy.sleep(0.5)
  kuka.ptp([0.10770660596114645, 0.7292428761611092,
     -0.801648366685086, -6.087224938771763e-06,
     2.0411520793821173e-05, 6.866109236713328e-05]) #OPEN DOOR
  rospy.sleep(0.5)
```

```
14    kuka.home()
```

7.10. Comandos útiles del proyecto y de *ROS*

Código fuente 7.6: Servicio *kuka_ driver* y *script* de ejemplo. ¡Advertencia! Mantener despejada el área de trabajo del robot.

```
1  # Launch del servidor kuka_driver:
2  $ roslaunch kuka_driver kuka_driver.launch
3  # Modo 1 Ejemplo script:
4  $ cd src/kuka_driver/scripts
5  $ python open_door.py
6  # Modo 2 Ejemplo script:
7  $ rosrun kuka_driver open_door.py
```

Código fuente 7.7: Comandos *GIT* útiles al iniciar el trabajo.

```
1  # Ir a la carpeta donde esta el repositorio GIT
2  $ git pull
```

Código fuente 7.8: Comandos *GIT* útiles al finalizar el trabajo.

```
1  # Ir a la carpeta donde esta el repositorio GIT
2  $ git status
3  $ git add .
4  $ git commit -m "NOMBRE_COMMIT"
5  $ git status
6  $ git pull
7  $ git push origin grupo4/devel
8  $ PASSWORD
```

Código fuente 7.9: Grafo de nodos y tópicos.

```
1  $ rosrun rqt_graph rqt_graph
```

Código fuente 7.10: Esquema jerárquico de *TF*.

```
1  $ rosrun rqt_tf_tree rqt_tf_tree
```

Código fuente 7.11: Scan del entorno virtual en *Gazebo*. Control proporcional del efector

```
1  $ roslaunch kuka_gazebo kuka_gazebo_phantom.launch
2  $ rosrun uchile_kuka kuka_scan.py
```

Código fuente 7.12: Scan del entorno virtual en *Gazebo*. Control deltas de efector.

```
$ roslaunch kuka_gazebo kuka_gazebo_phantom_zonas.launch
$ rosrun uchile_kuka kuka_scan.py
```

Código fuente 7.13: Ejecuta movimiento del camino verde ideal en panel de pruebas.

```
$ roslaunch kuka_driver kuka_driver.launch
$ rosrun kuka_driver green_path_go.py
```

Código fuente 7.14: Ejecuta movimiento del camino rojo ideal en panel de pruebas.

```
$ roslaunch kuka_driver kuka_driver.launch
$ rosrun kuka_driver red_path_go.py
```

Código fuente 7.15: Comandos usados para validación experimental.

```
$ roslaunch kuka_bringup kuka_teleop_1.launch path_file:=
   green_path_large2 save_data:=teleop1_green_user_NN.mat
$ roslaunch kuka_bringup kuka_teleop_3.launch path_file:=
   green_path_large2 save_data:=teleop3_green_user_NN.mat
$ roslaunch kuka_bringup kuka_teleop_2.launch path_file:=
   green_path_large2 save_data:=teleop2_green_user_NN.mat
$ roslaunch kuka_bringup kuka_teleop_4.launch path_file:=
   green_path_large2 save_data:=teleop4_green_user_NN.mat
$ roslaunch kuka_bringup kuka_teleop_1.launch path_file:=
   red_path_large3 save_data:=teleop1_red_user_NN.mat
$ roslaunch kuka_bringup kuka_teleop_3.launch path_file:=
   red_path_large3 save_data:=teleop3_red_user_NN.mat
$ roslaunch kuka_bringup kuka_teleop_2.launch path_file:=
   red_path_large3 save_data:=teleop2_red_user_NN.mat
$ roslaunch kuka_bringup kuka_teleop_4.launch path_file:=
   red_path_large3 save_data:=teleop4_red_user_NN.mat
```

7.11. Imágenes de la interfaz de usuario

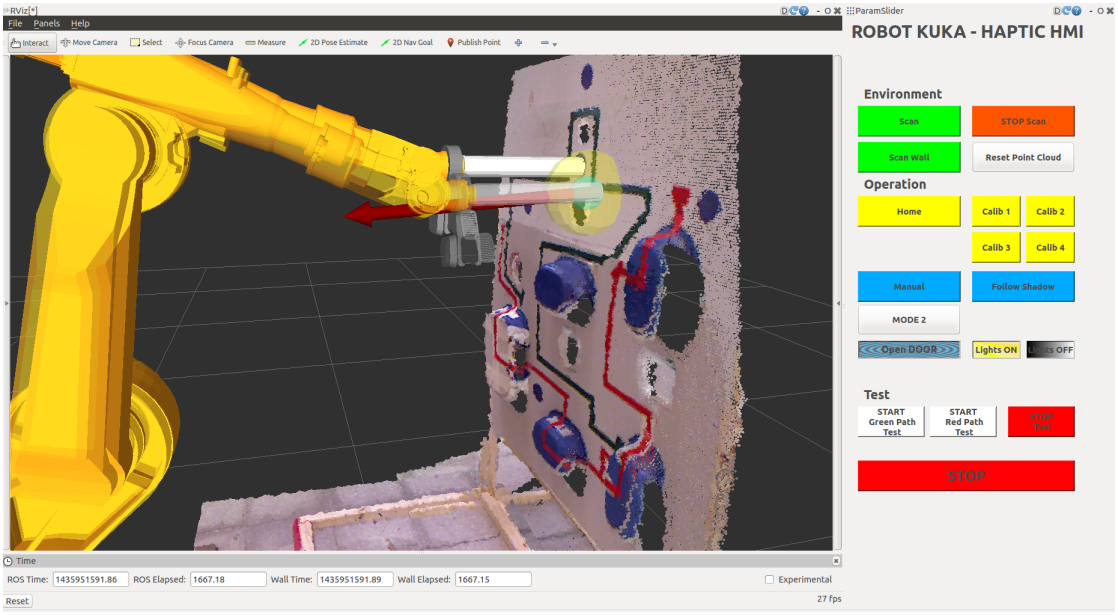

Figura 7.17: Interfaz de usuario 01.

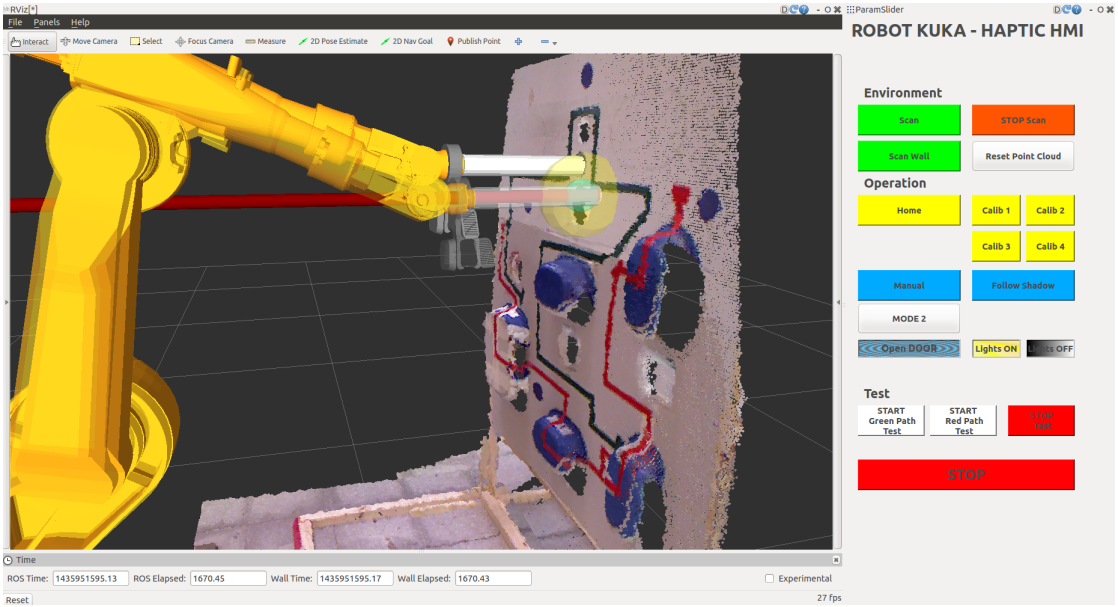

Figura 7.18: Interfaz de usuario 02.

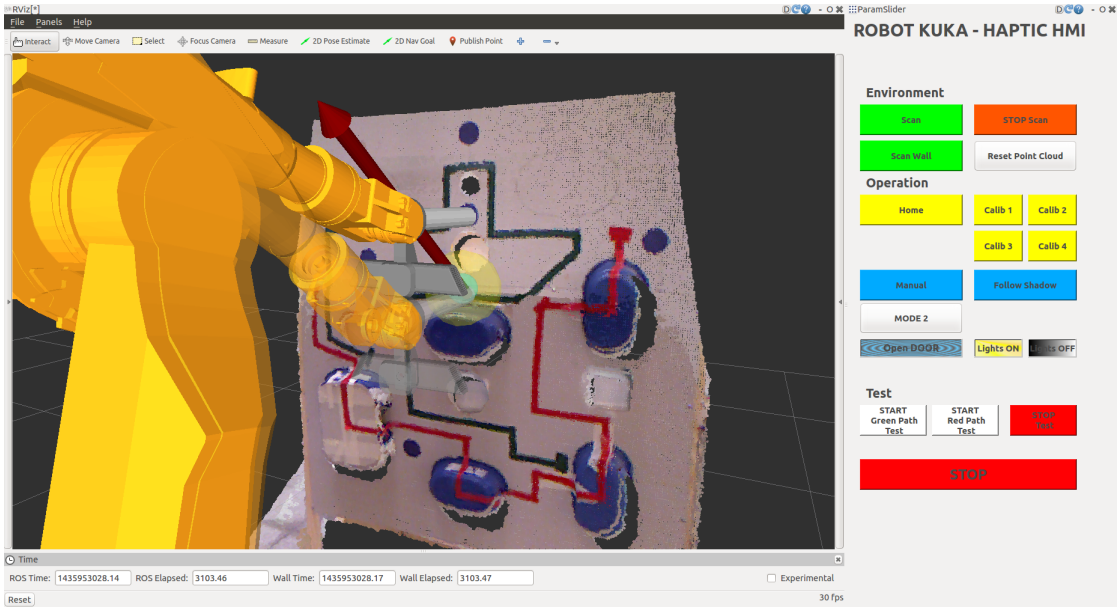

Figura 7.19: Interfaz de usuario 03.

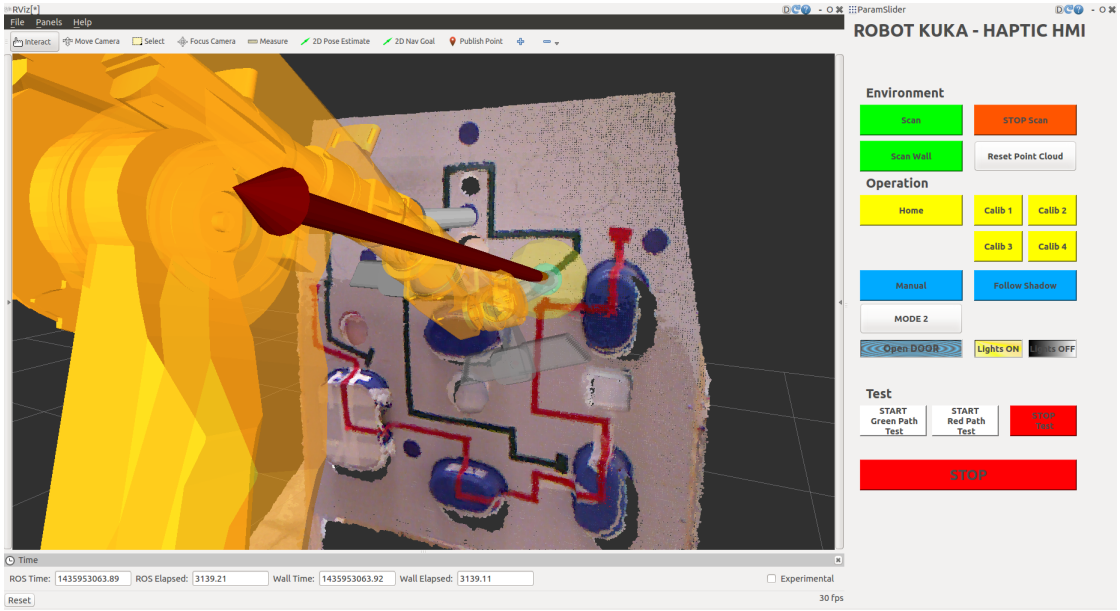

Figura 7.20: Interfaz de usuario 04.