UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

T E S I S

DISPLAY HÁPTICO DE SISTEMAS DINÁMICOS VIRTUALES Y TELEOPERADOS SUJETOS A RESTRICCIONES HOLONÓMICAS

QUE PARA OBTENER EL TÍTULO DE
INGENIERO ELÉCTRICO Y ELECTRÓNICO

PRESENTA:

JOSÉ ANTONIO VILLAVICENCIO CASTILLO

DIRECTOR DE TESIS
DR. MARCO ANTONIO ARTEAGA PÉREZ

CIUDAD UNIVERSITARIA, 2015

JURADO ASIGNADO:

Presidente:	Ing. Gabriel Alejandro Jaramillo Morales
Vocal:	Dr. Marco Antonio Arteaga Pérez
Secretario:	Ing. Gloria Mata Hernández
Suplente 1:	Dr. Jesús Savage Carmona
Suplente 2:	Dr. Paul Rolando Maya Ortíz

La tesis se realizó en el Laboratorio de Robótica del Departamento de Control de la Facultad de Ingeniería de la UNAM.

TUTOR DE TESIS:

Dr. Marco Antonio Arteaga Pérez

Agradecimientos

A la Universidad Nacional Autónoma de México,

a DGAPA-UNAM Proyecto IN116314.

Dedicatoria

Este camino no lo recorrí solo. Quiero dedicar esta tesis a mis mamás y mis papás que siempre estuvieron conmigo apoyándome y motivándome a llegar más lejos. También esta tesis va para Edgar, el primer amigo que tuve, espero que te motive a hacer una tesis 100 veces mejor. Para Taniz que desde que la conocí me hechó porras, me apoyó y me aguantó en mis momentos de mucho estrés; a todos mis amigos, que fueron mi equipo y mi apoyo dentro y fuera de las aulas y finalmente a mis maestros de la Facultad y de Oriente que me guiaron en el camino.

Índice general

Índice de figuras

Índice de tablas

Resumen

En este trabajo se abordan dos tipos de ambientes virtuales: una *interfaz háptica* y un *sistema de teleoperación*. Para la interfaz háptica, se pone en marcha el dispositivo háptico Geomagic Touch y se muestra un ejemplo de seguimiento de trayectorias. Para la interacción del operador con un ambiente virtual sujeto a una restricción holonómica se aborda el problema de la detección de colisiones del efector final con el objeto virtual y se realiza una simulación del comportamiento dinámico del objeto, incluyendo las fuerzas de reacción debido a las restricciones. Para el sistema de teleoperación, se considera un sistema maestro-esclavo compuesto por dos dispositivos hápticos; el esquema de control utilizado es el de *Posición Posición* implementado mediante controladores PD. En ambos casos se realiza el renderizado háptico utilizando las librerías de *OpenHaptics* y la parte gráfica es desarrollada con *OpenGL*. Finalmente, se muestran las conclusiones y trabajo futuro.

Capítulo 1

Introducción

"[...]En sus *Principios de la Filosofía*, [Descartes] dice 'Que por nuestros sentidos no sabemos nada de los objetos externos más allá de su figura, magnitud y movimiento'. Así, los cuerpos son percibidos como provistos de cualidades que en realidad no les pertenecen, cualidades que en realidad son puramente la prole de la mente. Así, la naturaleza recibe el crédito que en verdad debiera reservarse para nosotros; la rosa por su aroma; el ruiseñor por su cántico; y el Sol por su resplandor. Los poetas están enteramente equivocados. Debieran dirigir sus poemas a sí mismos, y convertirlos en odas de autofelicitación sobre la excelencia de la mente humana. La naturaleza es un asunto aburrido, sin sonidos, sin colores; meramente el apuro del material, interminablemente, carente de significado. No importa cómo lo disfraces, éste es el resultado práctico de la característica filosofía científica que creó el Siglo XVII" [1].

El problema de la realidad es un problema que aqueja a la mente humana desde hace varios siglos. ¿Qué es la realidad? Mucho se ha escrito al respecto, pero *definir* la realidad, lo que *realmente es*, seguramente sea una tarea que nunca pueda concretarse. Sin embargo, podemos estar de acuerdo en que el ser humano la ha modificado a lo largo de la historia para hacer su existencia, su *realidad* una experiencia más placentera. En los últimos años, el hombre ha ido más allá, al no solo modificarla, sino *crear* nuevas realidades. Este trabajo trata precisamente de eso, de crear una nueva realidad y poder interactuar con ella. La *Realidad Virtual* fue originalmente concebida como un espacio creado digitalmente en el que los humanos pudieran acceder utilizando equipo de cómputo sofisticado. Una vez dentro de ese espacio, las personas podrían ser transportadas a un mundo diferente, una realidad sustituta en la que uno pueda interactuar con objetos, personas y entornos, con una apariencia únicamente limitada por la imaginación humana [2].

Un *ambiente virtual* (AV) es un espacio digital en el cual los movimientos del usuario son *rastreados* y su entorno es *renderizado*, o compuesto digitalmente y presentado a los sentidos, de acuerdo con dichos movimientos [2].

En la Figura 1.1 se observa un ejemplo de la constitución de un AV. Otro tipo de ambientes virtuales son aquellos generados por los videojuegos, en donde el jugador interactúa con éste a través de un mando. Lo que tienen en común los dos ambientes mencionados anteriormente es que sólo proveen de realimentación visual al usuario, es decir, *pueden ver pero no tocar*. Para incorporar la sensación del tacto a un ambiente virtual, se recurre a la *háptica*.

Háptica es la ciencia de incorporar el sentido del tacto y control en aplicaciones de computadora a través de fuerzas (kinestésico) o realimentación táctil. Al utilizar dispositivos especiales de entrada/salida llamados dispositivos hápticos -principalmente robots- con una aplicación habilitada hápticamente, los usuarios pueden sentir y manipular objetos virtuales tridimensionales [3]. Otro tipo de ambiente virtual es el ambiente *teleoperado*. Los ambientes teleoperados son interesantes en el sentido de que son una *virtualización* de

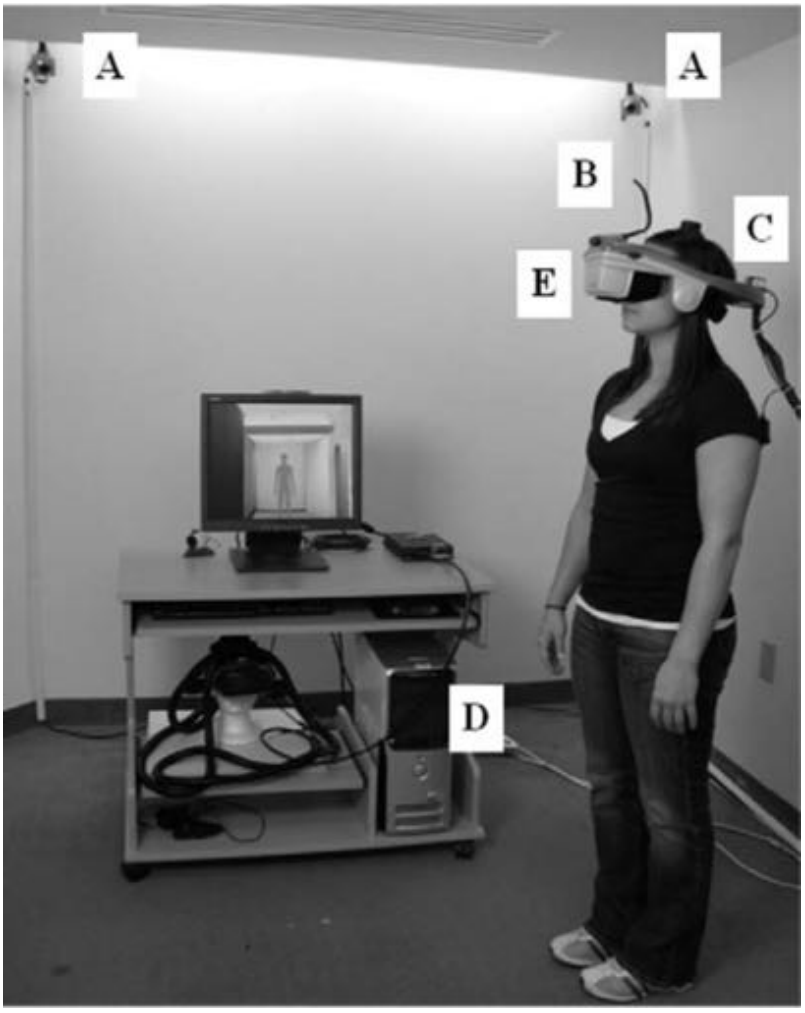


Figura 1.1: Ejemplo de disposición de un AV. Las cámaras (A) rastrean al sensor óptico (B) indicando la posición del participante en la habitación. Un acelerómetro (C) reúne información acerca de los movimientos de la cabeza del participante. Esta información es enviada a la computadora (D), que determina cómo es renderizada la habitación y lo que el participante ve en el display (E) [2]

un ambiente real, creando una réplica de algún entorno y permitiendo al operador interactuar con él como si se encontrara ahí. La realimentación puede ser visual y táctil, y hace uso de dispositivos hápticos para generar esta última. Aunque muy parecidas, una interfaz háptica y un sistema de teleoperación no son lo mismo. La diferencia entre una y otra es que un sistema de teleoperación consiste en que la señal de realimentación es enviada por la adquisición de datos de un sistema remoto (un robot esclavo, por ejemplo), mientras que en una interfaz háptica, las señales son obtenidas mediante modelos matemáticos del ambiente virtual [4].

La forma en que las sensaciones táctiles son generadas varía para los diferentes tipos de dispositivos. Generalmente, las fuerzas son generadas por algún tipo de motor que actúa sobre elementos mecánicos, hidráulicos o neumáticos. Dichos elementos pueden actuar directa o indirectamente sobre nosotros para generar sensaciones, que van desde simples colisiones hasta un renderizado complejo de fuerzas que permite identificar características como textura, rigidez o peso de algún objeto.

Los avances en la tecnología han hecho que el uso de ambientes virtuales, mediante interfaces hápticas y sistemas de teleoperación, se extiendan a un número cada vez mayor de áreas. Una aplicación común es el uso de ambientes virtuales en rehabilitación física, donde se recrean, de manera segura, entornos que pueden ser riesgosos para personas con alguna discapacidad. También los militares han dirigido su atención a ambientes virtuales y de teleoperación, principalmente como medios para entrenar a los soldados a tomar decisiones y reaccionar de la mejor manera en ambientes peligrosos y bajo condiciones de estrés extremas; además han sido útiles a los soldados en el desarrollo de habilidades que los preparan para realizar algún despliegue en otro país, con características muy diferentes al suyo [2]. Sin embargo, pese al uso extendido de AV en áreas como la militar, el entretenimiento o la rehabilitación, el área médica es seguramente el campo en el que el desarrollo de ambientes virtuales y teleoperados han tenido un crecimiento más acelerado.

Actualmente, se usan robots para una variedad de procedimientos quirúrgicos, tales como la fijación transpedicular, es decir, la fijación de dos o más vértebras mediante la inserción de tornillos y barras metálicas. Además, son usados en procedimientos quirúrgicos basados en agujas, tales como biopsias o vertebroplastias, o la ablasión y extracción de tumores. Incluso, hay investigaciones que buscan ayudarse de sistemas robóticos con interfaces hápticas que sean capaces de explorar y operar en el espacio sub-aracnoideo, el cual tiene apenas unos milímetros de ancho [5].

1.1. Planteamiento del problema

El desarrollo de ambientes virtuales (simulados y teleoperados) tiene varios retos que superar, con el objetivo de hacer la experiencia lo más realista posible. Algunos de esos retos son:

- **Adquisición de datos y sensado.** En el caso de sistemas de teleoperación, la información del ambiente percibida por el sistema remoto debe ser lo más exacta posible.
- **Mecánica.** El sistema maestro debe ser capaz de generar con bastante precisión las fuerzas calculadas que recreen diferentes sensaciones.
- **Canal de comunicaciones.** El flujo de información entre el sistema maestro y el esclavo debe ser contínuo e idealmente sin retardos.
- **Computación gráfica.** La virtualización de un ambiente debe ser fiel a la realidad.
- **Algoritmos de control y renderizado.** Se refiere al cálculo y la interpretación de fuerzas por el dispositivo háptico.

- **Modelado.** Cuando se diseña un simulador, las dinámicas de los objetos deben tener suficiente complejidad como para que el operador *sienta* que está interactuando con algo *real*.

El objetivo de este trabajo es desarrollar, por un lado, un ambiente virtual donde el operador pueda manipular un objeto sujeto a una restricción holonómica, y por otro, establecer las bases de un sistema de comunicación y un algoritmo de control para la teleoperación. El problema queda establecido como:

Desarrollar e implementar el código necesario que permita la interacción con ambientes virtuales y teleoperados a través de un dispositivo háptico.

Para el ambiente virtual, se considera un objeto rígido que se desliza sobre un riel. Para la parte de teleoperación, se considera que el único intercambio de información entre los sistemas maestro y esclavo es referente a su posición.

1.2. Estado del Arte

La tecnología háptica es un aspecto clave en la interacción hombre-máquina. La producción creciente de dispositivos hápticos ha dado pie al desarrollo de plataformas de desarrollo y librerías de programación que permiten no sólo la adición de una interfaz háptica a una aplicación gráfica existente (*e.g.* simuladores), sino extender las posibilidades e incluso crear nuevos esquemas en la interacción hombre-máquina.

1.2.1. Interfaces hápticas

Cuando un humano manipula algo, un procesamiento extensivo de la información recabada del entorno es realizado con el fin de controlar la acción. La información proviene de los impulsos generados por los sistemas sensoriales que consisten en los ojos, los oidos y receptores sensoriales dentro de los músculos, tendones, articulaciones y la piel [6, 7]. Información propioceptiva relacionada con el movimiento es llamada *kinestésica*, mientras que la realimentación *táctil* tiene que ver con colectar información acerca del mundo que nos rodea (como el sentido del tacto). Ambas trabajan juntas al proveer información del cuerpo humano y sus alrededores y ambas están ausentes cuando se utilizan robots manipuladores [8].

Realimentación kinestésica

Dentro de los músculos hay una serie de fibras nerviosas envueltas alrededor de zonas centrales del músculo que responden a variaciones longitudinales. De esta forma, cuando las fibras se contraen, disminuye la actividad sensorial y cuando se estiran, envían señales a la médula espinal que a su vez envían impulsos reflejos a los músculos. De esta forma, una persona puede sentir y responder cuando alguna parte de su cuerpo es extendida [6].

De la misma forma, hay fibras nerviosas envueltas en los tendones conectando los músculos a los huesos. Estas fibras responden cuando el músculo se contrae y el umbral de activación es un tanto bajo -responden a tensiones musculares de una décima de gramo[1] o menos-. Adicionalmente, hay fibras nerviosas dentro de las articulaciones, pero el rol de éstas aún no está bien definido. Investigaciones hechas sobre el tema indican que estas fibras responden a diferentes ángulos de las articulaciones, pudiendo ser un sistema de alerta cuando las articulaciones son dobladas hasta sus límites [9].

[1] Las unidades de *gramo* y *miligramo* en esta sección corresponden a unidades de fuerza.

Realimentación táctil

Cuando movemos alguna parte del cuerpo, tocamos algún objeto o nos recargamos sobre algo, se crea una sensación de desplazamiento o *toque*. Esto es debido a una serie de sensores, los *mecanoreceptores*, que están distribuidos a lo largo de la piel y que responden a las deformaciones mecánicas de esta. También hay sensores en la piel que responden a la temperatura y al dolor [7]. Los sentidos del tacto, temperatura y dolor también son conocidos como *realimentación táctil*. El sentido del movimiento pertenece a la realimentación kinestésica, sin embargo, está fuertemente relacionada con la realimentación táctil ya que los mismos sensores proveen de ambas experiencias sensoriales.

Como se mencionó antes, toda la información sensorial recogida del entorno es enviada a la médula espinal, donde se realiza un primer nivel de procesamiento capaz de enviar señales a los músculos, lo que conocemos como *reflejos*.

La sensibilidad táctil es muy alta. Aunque cambia de persona a persona, en promedio el umbral de detección es de aproximadamente 80 mg en el dedo y de 150 mg en la palma de la mano [10]. La habilidad de discriminar dos puntos diferentes mientras son estimulados simultáneamente en la piel se limita a 2.5 mm entre puntos en el dedo y a 11 mm en la palma de la mano [11]. El límite inferior para detectar y discriminar dos estímulos consecutivos es de 5 ms (el correspondiente límite para el ojo es de 25 ms). Similarmente, la mayor sensibilidad para detectar vibraciones es para frecuencias entre 200 y 250 Hz [12].

El *ancho de banda* nos permite definir la rapidez en que podemos percibir estímulos (entrada) y responder a ellos (salida). El ancho de banda de entrada es mayor que el de salida, lo que significa que podemos percibir estímulos más rápido de lo que podemos responder a ellos. En [11] se menciona que el ancho de banda del lazo cerrado entre el *sensor* y el *motor* de la mano y dedos es de aproximadamente 5-10 Hz. En cambio, la sensación kinestésica tiene un ancho de banda de 20-30 Hz y la táctil de 0-400 Hz. Esto quiere decir que somos más sensibles a sensaciones como la presión o temperatura que a movimientos.

1.2.2. APIs

Para esta tesis se utilizaron las librerías de *OpenHaptics* proporcionadas por Geomagic[2]. Además de *OpenHaptics*, existen otras librerías bajo licencias públicas que sirven para implementar aplicaciones hápticas. Las dos más famosas son *CHAI 3D* y *H3D API*.

CHAI 3D es un conjunto de librerías de código abierto escritas en C++ que son compatibles con dispositivos hápticos de varias marcas de 3, 6 y hasta 7 grados de libertad. De igual forma, *H3D API* es un conjunto de librerías de código abierto; están escritas en C++ y se basan en plataformas *open source* como *OpenGL*. Es multiplataforma y es independiente del dispositivo háptico utilizado. Como característica especial, soporta la integración de audio y estereografía (en pantallas que lo soporten).

[2]Geomagic adquirió Sensable Inc. en el 2012, es por ello el cambio de nombre de sus productos.

1.2.3. Dispositivos hápticos comerciales

En la Tabla 1.1 se presentan los principales dispositivos hápticos disponibles comercialmente.

Dispositivo	DOF	Espacio de trabajo	F_{max}	Resolución
Geomagic Touch	6 (3 actuados)	160W x 120H x 70D mm	3.3N	450 dpi $\backsim 0.055mm$
Geomagic Touch X	6 (3 actuados)	160W x 120H x 120D mm	7.9N	1100 dpi $\backsim 0.023mm$
Geomagic Phantom Premium 3.0	6	838W x 584H x 406D mm	22N	1000 dpi $\backsim 0.02mm$
Novint Falcon	3	101.6W x 101.6H x 101.6D mm	8.9N	400 dpi $\backsim 0.063mm$

Tabla 1.1: Principales dispositivos hápticos comerciales

El robot utilizado en esta tesis corresponde a un robot antropomórfico de 6 grados de libertad (únicamente 3 actuados). Específicamente, el modelo utilizado es el Geomagic Touch, anteriormente conocido como Sensable Phantom Omni (ver Figura 1.2). Este modelo cuenta con una interfaz EtherNet compatible con terminales RJ45 o USB (en el caso de USB, viene con adaptador RJ45-USB incluido). Una de sus características es un *stylus* que hace la función de muñeca esférica; además, cuenta con dos botones programables. Es uno de los dispositivos hápticos más populares al ofrecer *más valor por su dinero*. Es ampliamente usado en modelado 3D, investigación y simuladores médicos. Adicionalmente, varias compañías lo han integrado en sus productos, tales como *FiatLux* (modelado molecular), *Gemvision Corporation* (software de diseño de joyería en 3D), *Medic Vision* (simulador médico), etc.

Figura 1.2: Geomagic Touch

1.2.4. Cirugías asistidas por robots

En el área médica, existen varios sistemas que asisten al cirujano en una operación. Aunque hay varios desarrollos que involucran el uso de robots en el transcurso de una operación, como el Sistema Quirúrgico *da Vinci* [13], son pocos los que incluyen algún tipo de realimentación táctil. Es importante recalcar que, pese a la opinión popular, estos sistemas tienen un grado limitado de autonomía, siendo más bien su propósito principal el asistir al médico en la operación, ya sea a través de movimientos más precisos (*e.g.* sin temblores), de la fijación de restricciones en el entorno de la operación o el acceso a zonas difíciles, reduciendo el tamaño de la incisión.

***Cooperative Robotic Assistant* CoRA**

El robot CoRA (Fig. 1.3) es un dispositivo complejo capaz de automatizar el proceso de taladrado e inserción de tornillos en aplicaciones médicas, que además genera fuerzas de reacción que prohiben al cirujano acceder a zonas delicadas. CoRA ofrece además un sistema de perforación teleoperado con sensaciones hápticas realistas, a través de un efector final pequeño y ligero [14].

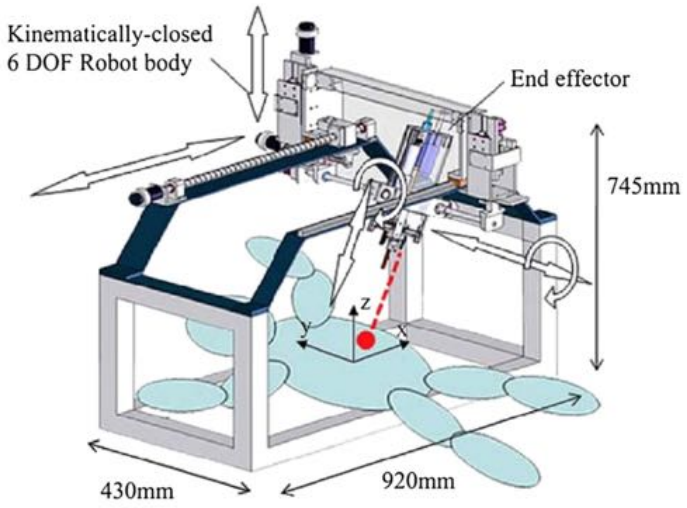


Figura 1.3: *Cooperative Robotic Assistant* (CoRA), robot con realimentación háptica para la inserción de tornillos

Robot in Medical Environment **RIME**

El proyecto *RIME* (*Robot in Medical Environment*) es un proyecto enfocado en la perforación en cirugías de fijación transpedicular. La principal contribución del proyecto es el desarrollo de un sistema de teleoperación, que permite al cirujano operar al paciente incluso si éste se encuentra a kilómetros de distancia. El sistema cuenta con realimentación háptica y ha sido probado entre dos ciudades a 35 km de distancia [14].

Robot Spinal Surgical System **RSSS**

El *RSSS* es un robot diseñado principalmente para la inserción de tornillos en los pedículos. Está basado en un robot SCARA de 5 grados de libertad, equipado con un dispositivo de rastreo infrarrojo. El diseño mecánico del RSSS asegura que el robot no se desplome en caso de una falla en la alimentación, protegiendo al paciente. Ofrece realimentación háptica, herramientas virtuales, un mecanismo de inserción de tornillos y una estrategia de control para perforado automático, la cual es capaz de identificar los perfiles de fuerza para cada etapa de la perforación y detenerse de manera automática antes de alcanzar la vértebra [14, 15].

1.2.5. AIREAL: Experiencias táctiles en el aire

En [16] se habla de AIREAL, una novedosa tecnología háptica que otorga sensaciones efectivas y expresivas en el aire sin la necesidad de usar un dispositivo físico. Combinándolo con gráficos computarizados interactivos, AIREAL le permite al usuario interactuar con objetos virtuales, percibir texturas en el aire y recibir realimentación háptica en gestos realizados en el aire (Figura 1.4).

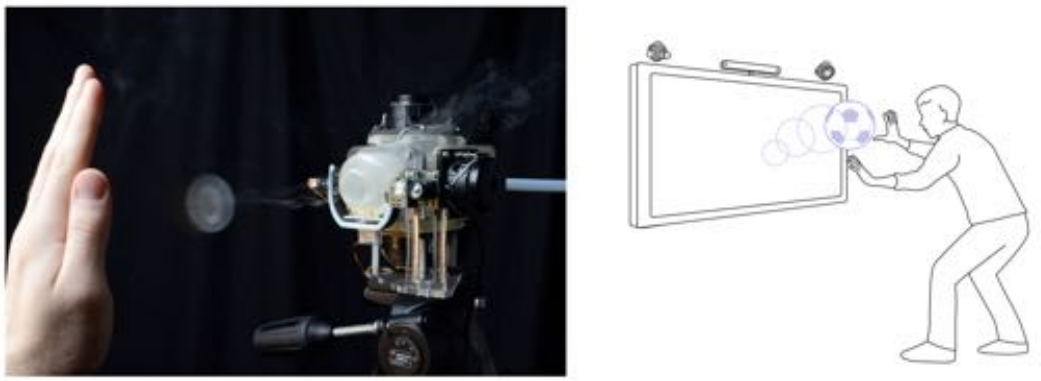


Figura 1.4: Sistema AIREAL [16]

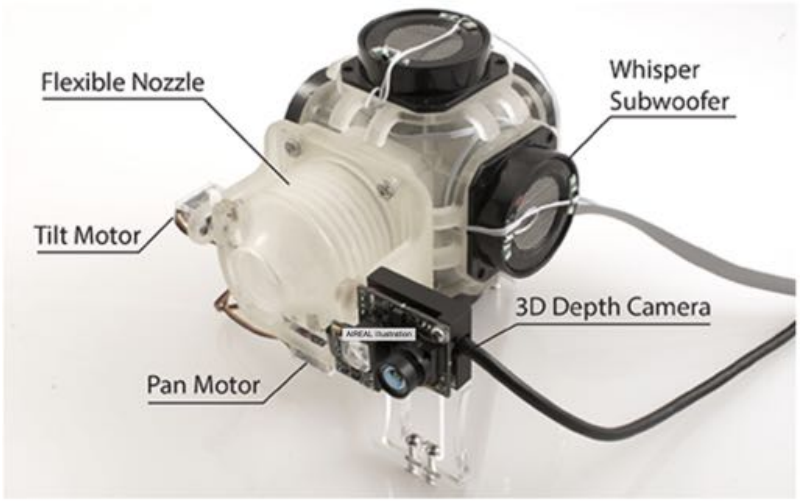


Figura 1.5: Composición de AIREAL [16]

Su principio de funcionamiento es a través de la creación de vórtices de aire que son dirigidos por una boquilla flexible, teniendo un ángulo de visión de 75° y una resolución de 8.5 cm a 1 metro de distancia. Utiliza 5 actuadores para dirigir la boquilla flexible y un par de cámaras que conforman una cámara 3D que mide la profundidad (Figura 1.5). Es escalable, de bajo costo y puede ser utilizado en una gran variedad de aplicaciones. Como dato interesante, el sistema está construido en su mayoría por piezas obtenidas de sistemas de impresión 3D.

1.3. Aplicaciones

El uso que se le puede dar a una interfaz háptica es prácticamente ilimitado. Pensemos en cualquier posible interacción que podamos tener con alguna computadora y será fácil encontrar la manera de incluir una interfaz háptica. Sin embargo, las principales aplicaciones hoy en día son:

- Dibujo y modelado por computadora
- Simuladores (de vuelo, manejo, cirugías, etc.)
- Entretenimiento
- Sustitución de periféricos (*e.g.* control remoto, teclados, etc.)

- Teleoperación
- Asistencia en procedimientos quirúrgicos

1.4. Contribución

Las principales contribuciones de esta tesis son:

- La puesta en marcha de un dispositivo háptico Geomagic Touch y la estructura de programación necesaria para probar distintos algoritmos de control y mostrarlos virtualmente.
- El desarrollo de un modelo dinámico sujeto a restricciones holonómicas que pueda interactuar con una interfaz háptica.
- Un esquema de teleoperación entre dos robots sin usar sensores de fuerza.
- La estructura de un canal de comunicación para sistemas de teleoperación que requieran de múltiples canales.

1.5. Organización de la tesis

En el Capítulo 1 se presenta el estado del arte de interfaces hápticas y ambientes virtuales. Se especifica el problema que aborda la tesis y la contribución que se hace.

En el Capítulo 2 se dan las herramientas que se utilizan para el diseño del controlador, el algoritmo de colisión, la obtención del modelo dinámico del sistema virtual y el renderizado de fuerzas.

En el Capítulo 3 se presenta la estructura de programación del dispositivo háptico Geomagic Touch y un ejemplo de seguimiento de trayectorias utilizando una serie de controladores lineales.

En el Capítulo 4 se presenta un algoritmo de colisión para el renderizado de fuerza en la interacción del dispositivo háptico con el ambiente virtual. También se incluye la simulación de las restricciones holonómicas a las que está sujeto el objeto virtual.

En el Capítulo 5 se desarrolla un sistema de teleoperación entre dos dispositivos hápticos Geomagic Touch. Adicionalmente se propone un método de comunicación entre un Geomagic Touch y un robot industrial para aplicaciones futuras.

En el Capítulo 6 se presentan las conclusiones y el trabajo futuro.

Capítulo 2

Preliminares

2.1. Restricciones holonómicas

Las restricciones cinemáticas impuestas por el ambiente pueden ser representadas por un conjunto de ecuaciones algebraicas de las variables que describen de manera efectiva la posición y orientación del objeto. Estas variables son las llamadas *coordenadas generalizadas* [17]. Las ecuaciones de restricción, escritas en términos de estas variables son de la forma

$$\boldsymbol{\phi}(\boldsymbol{\chi}) = 0. \tag{2.1}$$

El vector $\boldsymbol{\chi}$ es un vector de $n \times 1$, donde n es el número de coordenadas generalizadas. Este tipo de restricciones, que involucran únicamente las coordenadas generalizadas del sistema, son conocidas como *restricciones holonómicas*.

La diferencial total de la restricción está dada por

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial \boldsymbol{\chi}} \mathrm{d}\boldsymbol{\chi} = \boldsymbol{J}_{\phi} \mathrm{d}\boldsymbol{\chi}. \tag{2.2}$$

A la restricción holonómica se le conoce también como restricción integrable. Nótese que las restricciones holonómicas dependen únicamente de la posición $\boldsymbol{\chi}$ y no de la velocidad $\dot{\boldsymbol{\chi}}$. El efecto de las restricciones en la dinámica del sistema se refleja como fuerzas que no efectúan trabajo (fuerzas virtuales) [17].

2.2. Modelado

Un *modelo dinámico* es un conjunto de ecuaciones matemáticas que describen el comportamiento de un sistema. Las ecuaciones que lo constituyen son *ecuaciones diferenciales* que contienen los parámetros del sistema (como coeficientes de fricción, masas, inercias) y cuya solución representa el comportamiento en el tiempo de este. Para modelar un sistema, se necesitan hacer muchas suposiciones y descartar comportamientos y parámetros que en un momento dado pueden ser despreciables en el modelo. Por ejemplo, si modelamos el comportamiento de una bola de billar en caída libre, la fricción del aire se puede considerar despreciable, por lo que se descarta de las ecuaciones dinámicas del sistema, en cambio, si modelamos un avión en su etapa de despegue, la fricción y la dinámica que tiene el aire sobre las alas son de vital importancia y no pueden despreciarse.

Las ecuaciones dinámicas de un sistema se obtienen a partir de las diferentes relaciones constitutivas de sus elementos. Estas relaciones constitutivas, para sistemas físicos, son las leyes físicas que los gobiernan. Por ejemplo, en sistemas mecánicos, pueden ser las Leyes de Newton, para sistemas eléctricos, la Ley de Ohm y las Leyes de Kirchhoff, etc.

2.2.1. Modelado del ambiente virtual

La mayoría de los sistemas mecánicos pueden modelarse con bastante precisión mediante ecuaciones diferenciales de segundo orden de la forma:

$$a_0\frac{\mathrm{d}^2c(t)}{\mathrm{d}t^2}+a_1\frac{\mathrm{d}c(t)}{\mathrm{d}t}+a_2c(t)=b_0\frac{\mathrm{d}^2r(t)}{\mathrm{d}t^2}+b_1\frac{\mathrm{d}r(t)}{\mathrm{d}t}+b_2r(t) \tag{2.3}$$

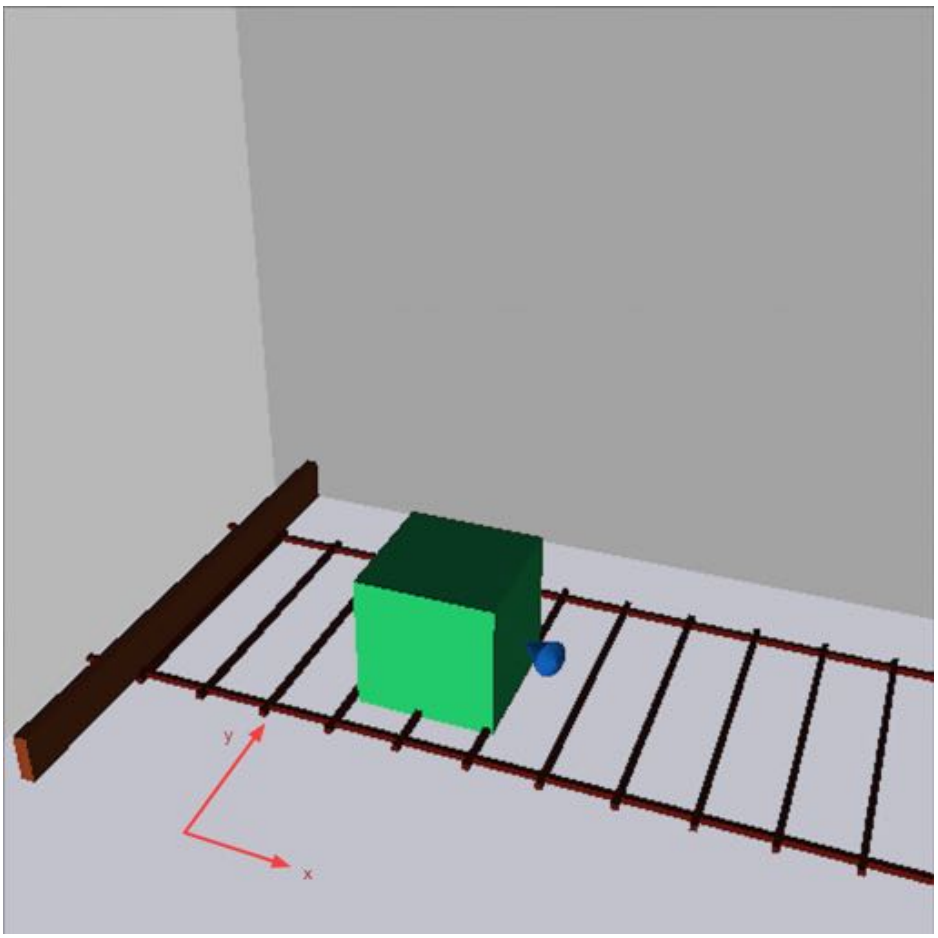


Figura 2.1: Sistema dinámico del ambiente virtual

En la Figura 2.1 se observa el sistema dinámico utilizado en el ambiente virtual que interactúa con la interfaz háptica. Se trata de una masa que está sujeta a cuatro fuerzas: una fuerza inherente a su masa (inercia), una fuerza de fricción, la fuerza del dispositivo háptico (representada por el puntero azul en la figura) y una fuerza derivada de las restricciones holonómicas (originada por los rieles). Puesto que se trata de un solo objeto en el espacio, sus coordenadas generalizadas son[1]

$$\boldsymbol{\chi}=\begin{bmatrix} x \\ y \end{bmatrix} \tag{2.4}$$

La fuerza de fricción se puede simular mediante la siguiente ecuación:

$$\boldsymbol{f}_{\mathrm{f}}(\dot{\boldsymbol{\chi}})=b\dot{\boldsymbol{\chi}}+f_{\mathrm{c}}\cdot\mathrm{sgn}(\dot{\boldsymbol{\chi}})+\boldsymbol{f}_{\mathrm{e}} \tag{2.5}$$

donde $\dot{\boldsymbol{\chi}}$ es la velocidad del objeto, b el coeficiente de fricción viscosa, f_{c} el coeficiente de fricción de Coulomb y $\boldsymbol{f}_{\mathrm{e}}$ la fuerza de fricción estática, que suele representarse mediante

[1]El movimiento del objeto se limita al plano XY para simplificar las ecuaciones.

una zona de histéresis, definida por:

$$\boldsymbol{f}_{\mathrm{e}} = \begin{cases} -\boldsymbol{F} & \text{si } \boldsymbol{F} \leq f_{\mathrm{e}_0} \\ -\boldsymbol{F} \cdot \mathrm{sgn}(\dot{\boldsymbol{\chi}}) & \text{si } \boldsymbol{F} > f_{\mathrm{e}_0} \end{cases} \tag{2.6}$$

con f_{e_0} la fuerza de umbral para iniciar el movimiento. Haciendo un balance de fuerzas, la ecuación que caracteriza el movimiento de esta masa es:

$$\boldsymbol{H}\ddot{\boldsymbol{\chi}} + \boldsymbol{f}_{\mathrm{f}}(\dot{\boldsymbol{\chi}}) = \boldsymbol{F} + \boldsymbol{J}_{\phi}^{\mathrm{T}}\boldsymbol{\lambda} \tag{2.7}$$

con $\boldsymbol{\chi} \in \Re^2$ el vector de coordenadas generalizadas, $\boldsymbol{H} \in \Re^{2\times 2}$ la matriz de inercias del objeto, $\boldsymbol{f}_{\mathrm{f}} \in \Re^2$ el vector de fuerzas de fricción que es función de su velocidad, $\boldsymbol{F} \in \Re^2$ el vector de fuerzas ejercidas por el efector final del dispositivo háptico y $\boldsymbol{J}_{\phi}^{\mathrm{T}}\boldsymbol{\lambda} \in \Re^2$ el vector de fuerzas de reacción debido a las restricciones.

La fuerza de contacto $\boldsymbol{F}$ se modela como un sistema Masa-Resorte-Amortiguador (ver Figura 2.2).

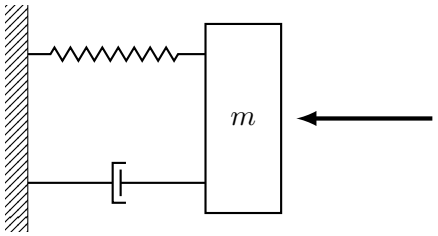


Figura 2.2: Modelado de la fuerza de contacto del efector final

Si se considera a la masa del efector final despreciable, la fuerza de contacto queda definida como:

$$\boldsymbol{F} = k(\boldsymbol{\chi} - \boldsymbol{\chi}_0) + b\dot{\boldsymbol{\chi}} \tag{2.8}$$

con k la constante del resorte, b la constante de amortiguamiento, $\boldsymbol{\chi}$ la posición del efector final y $\boldsymbol{\chi}_0$ el punto de contacto con el objeto. La razón de modelar a la fuerza de contacto como un sistema MRA es debido al algoritmo de colisión elegido para esta tesis (*proxy* virtual).

Modelado de las restricciones

La derivada en el tiempo de (2.1) da como resultado

$$\boldsymbol{J}_{\phi}(\boldsymbol{\chi})\dot{\boldsymbol{\chi}} = 0 \tag{2.9}$$

donde $\boldsymbol{J}_{\phi}(\boldsymbol{\chi}) = \partial\phi/\partial\boldsymbol{\chi}$ es el *Jacobiano de la restricción*. En la ausencia de fricción, las fuerzas de interacción son fuerzas de reacción que se originan cuando el efector final tiende a violar las restricciones, y pueden ser calculadas usando el principio de *trabajo virtual*. Considerando esto, las fuerzas de reacción $\boldsymbol{\tau}$ son [17]:

$$\boldsymbol{\tau} = \boldsymbol{J}_{\phi}^{\mathrm{T}}(\boldsymbol{\chi})\boldsymbol{\lambda} \tag{2.10}$$

con $\boldsymbol{\lambda}$ un vector de $m \times 1$ donde m es el número de restricciones.

El Jacobiano de las restricciones es conocido, por lo tanto el problema se reduce a encontrar el vector $\boldsymbol{\lambda}$ para obtener las fuerzas de reacción debido a las restricciones.

La solución teórica de un sistema sujeto a restricciones se obtiene usualmente mediante el método de los multiplicadores de Lagrange, que conduce a un conjunto de ecuaciones diferenciales y algebráicas con las coordenadas y los multiplicadores de Lagrange ($\boldsymbol{\lambda}$) como

incógnitas. Empero, la solución numérica de dicho conjunto no es un problema sencillo. En [18] se mencionan tres grupos principales de métodos que se han reportado para la solución de este problema.

El método elegido para esta tesis consiste en convertir al sistema de ecuaciones diferenciales y algebraicas en un conjunto de ecuaciones diferenciales ordinarias (ODEs), mediante el cálculo de la segunda derivada con respecto al tiempo de las restricciones.

Considérese el sistema (2.7), sujeto a la restricción

$$\phi_1(\boldsymbol{\chi}) = a - y = 0 \tag{2.11}$$

que representan un movimiento restringido únicamente al eje Y. La segunda derivada de (2.1) es

$$\boldsymbol{J}_\phi \ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{J}}_\phi \dot{\boldsymbol{\chi}} = 0. \tag{2.12}$$

Sustituyendo $\ddot{\boldsymbol{\chi}}$ de (2.7) en (2.12) se obtiene

$$\boldsymbol{J}_\phi \left[\boldsymbol{H}^{-1} \left(\boldsymbol{F} + \boldsymbol{J}_{\boldsymbol{\phi}}^{\mathrm{T}} \boldsymbol{\lambda} - \boldsymbol{f}_{\mathrm{f}}(\dot{\boldsymbol{\chi}}) \right) \right] + \dot{\boldsymbol{J}}_\phi \dot{\boldsymbol{\chi}} = 0. \tag{2.13}$$

Desarrollando términos, se puede despejar y obtener una expresión de $\boldsymbol{\lambda}$:

$$\begin{gathered} \boldsymbol{J}_\phi \boldsymbol{H}^{-1} \boldsymbol{F} + \boldsymbol{J}_\phi \boldsymbol{H}^{-1} \boldsymbol{J}_\phi^{\mathrm{T}} \boldsymbol{\lambda} - \boldsymbol{J}_\phi \boldsymbol{H}^{-1} \boldsymbol{f}(\dot{\boldsymbol{\chi}}) + \dot{\boldsymbol{J}}_\phi \dot{\boldsymbol{\chi}} = 0 \\ \Rightarrow \boldsymbol{\lambda} = \left[\boldsymbol{J}_\phi \boldsymbol{H}^{-1} \boldsymbol{J}_\phi^{\mathrm{T}} \right]^{-1} \left[\boldsymbol{J}_\phi \boldsymbol{H}^{-1} \boldsymbol{f}_{\mathrm{f}}(\dot{\boldsymbol{\chi}}) - \dot{\boldsymbol{J}}_\phi \dot{\boldsymbol{\chi}} - \boldsymbol{J}_\phi \boldsymbol{H}^{-1} \boldsymbol{F} \right]. \end{gathered} \tag{2.14}$$

El Jacobiano de la restricción (2.11) es:

$$\boldsymbol{J}_\phi = \begin{bmatrix} 0 & -1 \end{bmatrix}; \tag{2.15}$$

la matriz $\boldsymbol{H}$, al ser un objeto que no rota, se escribe como

$$\boldsymbol{H} = \begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} = m\boldsymbol{I}; \tag{2.16}$$

$\boldsymbol{f}_{\mathrm{f}}(\dot{\boldsymbol{\chi}})$ se puede escribir como

$$\boldsymbol{f}_{\mathrm{f}} = \begin{bmatrix} f_{\mathrm{x}}(\dot{x}) \\ f_{\mathrm{y}}(\dot{y}) \end{bmatrix} \tag{2.17}$$

y $\boldsymbol{F}$ como

$$\boldsymbol{F} = \begin{bmatrix} F_{\mathrm{x}} \\ F_{\mathrm{y}} \end{bmatrix}. \tag{2.18}$$

Al sustituir términos en (2.14) se tiene que

$$\boldsymbol{\lambda} = \frac{1}{m} \left[\frac{1}{m} f_{\mathrm{y}}(\dot{y}) + \frac{1}{m} F_{\mathrm{y}} \right]. \tag{2.19}$$

Lo que sigue es resolver la Ec. (2.7). El modelo

$$\boldsymbol{H} \ddot{\boldsymbol{\chi}} + \boldsymbol{f}_{\mathrm{f}}(\dot{\boldsymbol{\chi}}) = \boldsymbol{F} + \boldsymbol{J}_\phi^{\mathrm{T}} \boldsymbol{\lambda} \tag{2.20}$$

se puede resolver mediante la resolución del sistema de ecuaciones diferenciales ordinarias

$$\begin{aligned} m\ddot{x} + f_{\mathrm{x}}(\dot{x}) &= F_{\mathrm{x}} \\ m\ddot{y} + f_{\mathrm{y}}(\dot{y}) &= F_{\mathrm{y}} - \tfrac{1}{m} \left[\tfrac{1}{m} f_{\mathrm{y}}(\dot{y}) + \tfrac{1}{m} F_{\mathrm{y}} \right]. \end{aligned} \tag{2.21}$$

utilizando cualquier método numérico (ver Apéndice A).

2.3. Dispositivos hápticos

Robots manipuladores

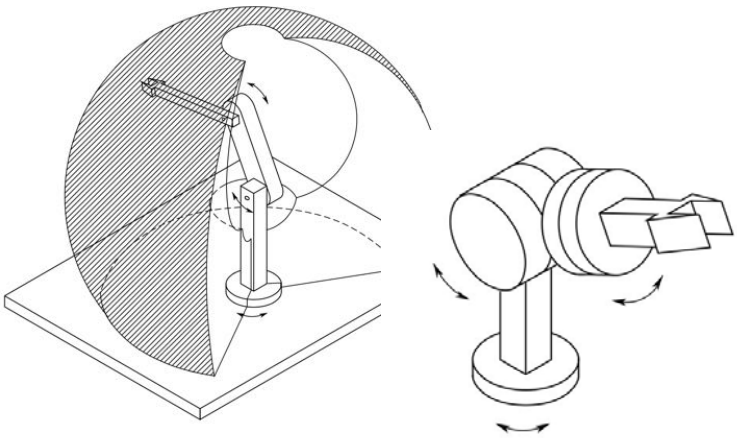
En la actualidad, los robots constituyen la mayoría de dispositivos hápticos en el mercado. La estructura mecánica de un robot manipulador consiste en una secuencia de cuerpos rígidos (*eslabones*) interconectados por medio de *articulaciones*; los robots manipuladores están caracterizados por un *brazo* que se asegura de la movilidad, una *muñeca* que le confiere destreza y un *efector final* que ejecuta la tarea requerida por el robot.

La estructura fundamental de un manipulador es una *cadena cinemática*. Desde un punto de vista topológico, una cadena cinemática se dice que es *abierta* cuando sólo existe una secuencia de eslabones interconectando las dos terminales de la cadena. De manera alternativa, un manipulador tiene una *cadena cinemática cerrada* cuando una secuencia de eslabones forman un lazo.

La movilidad de un manipulador es asegurada por la presencia de articulaciones. Las articulaciones entre dos eslabones consecutivos pueden ser *prismáticas* o de *revolución*. En una cadena cinemática abierta, cada articulación provee a la estructura de un *grado de libertad*. Una articulación prismática crea un movimiento relativo traslacional entre dos eslabones, mientras que la de revolución crea un movimiento relativo rotacional.

Los grados de libertad deben ser distribuidos apropiadamente a lo largo de la estructura mecánica para que pueda ejecutar una tarea dada. En el caso más general de una tarea que consiste en posicionar al efector final en un espacio tridimensional, se requieren 6 grados de libertad: tres para el posicionamiento en el punto deseado y otros tres para la orientación. Si hay más grados de libertad, se dice que el robot es *redundante* desde un punto de vista cinemático.

El *espacio de trabajo* representa la porción del entorno en la que el efector final puede acceder. Su forma y volumen dependen de la estructura del robot. El tipo y secuencia de grados de libertad (articulaciones), comenzando desde la base al efector final, permiten clasificar a los robots manipuladores en: *Cartesianos, cilíndricos, esféricos, SCARA y antropomórficos* [17].



(a) Robot antropomórfico de 3 *DOF*

(b) Muñeca esférica

Figura 2.3: Robot antropomórfico y su espacio de trabajo [17]

El tipo de robot utilizado en esta tesis es un robot antropomórfico de 6 grados de libertad, con los tres grados de posición actuados (Figura 2.3).

2.4. Controladores

En ingeniería, controlar significa decidir sobre el comportamiento de un sistema. De manera más específica se puede ver como manipular ciertas variables de un sistema (físico, biológico o social) para conseguir que ella u otras variables se comporten de la forma deseada, en el momento deseado. En general, se tienen dos tipos de control: en lazo abierto (sin realimentación) y en lazo cerrado (con realimentación), y dos objetivos: regulación (mantener valores constantes) y seguimiento de trayectorias.

Un *Sistema de Control* es una estructura dinámica que está diseñada para operar de una forma prescrita, a pesar de efectos inevitables (perturbaciones) que actúen sobre él. Los componentes básicos de un sistema de control son [19]:

- La planta o el objeto de control
- Actuadores para la planta
- Sensores que midan el punto de operación actual de la planta
- El controlador que maneja a la planta de acuerdo con el objetivo general de control

Existen muchos tipos de controladores, que se clasifican dependiendo de la estructura matemática en la que están basados. Seguramente la familia de controladores más populares es la *Proporcional + Integral + Derivativo (PID)*. Éstos se caracterizan por basar su respuesta en una señal de error. Puesto que la señal de entrada de este controlador es la diferencia entre un valor deseado y un valor real, prácticamente se puede implementar un mismo control a cualquier planta sin tomar en consideración la dinámica de ésta (aunque no es recomendable). Su facilidad de implementación y su baja complejidad -y con esto bajo costo- han hecho de este tipo de control uno de los más populares y más ampliamente usados por la industria.

El uso de estos controladores está definido principalmente para sistemas lineales con una entrada y una salida (*SISO*). En un sistema *multicuerpo*, como un robot manipulador, se tienen más de una entrada y una salida (*MIMO*). Además, la dinámica de cada eslabón es no lineal e influye sobre la de los demás. Es posible aproximar un sistema multicuerpo como un conjunto de sistemas SISO independientes, cuyas dinámicas debidas al acoplamiento entre eslabones son modeladas como una perturbación [20]. Este enfoque es conocido como *Control de Articulaciones Independiente* y simplifica mucho el análisis. Empero, lo mejor es tratar al sistema en conjunto y tomar en cuenta las dinámicas no lineales y los efectos del acoplamiento entre eslabones. Siguiendo esto último, las ecuaciones dinámicas de un robot manipulador se pueden formular en forma matricial [20]:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{u} \tag{2.22}$$

donde $\boldsymbol{H} \in \Re^{n\times n}$ es la matriz de inercias del sistema, $\boldsymbol{g}(\boldsymbol{q}) \in \Re^n$ el vector de gravedad, $\boldsymbol{C} \in \Re^{n\times n}$ la matriz de Coriolis y de fuerzas centrípetas y $\boldsymbol{u}$ el vector de entradas. Nótese que se usan las variables articulares $\boldsymbol{q}$ y no las coordenadas del espacio de operación. Con esto en mente, se pueden reescribir los controladores P, PI, PD y PID en forma vectorial. Por ejemplo, el control PD se escribe como:

$$u = -\boldsymbol{K}_{\mathrm{P}}\tilde{\boldsymbol{q}} - \boldsymbol{K}_{\mathrm{D}}\dot{\boldsymbol{q}} \tag{2.23}$$

donde la tilde representa un error.

2.4.1. Control P

Considérese un sistema representado por la ecuación diferencial:

$$\ddot{y} + k_1\dot{y} + k_2 y = u + d \tag{2.24}$$

donde $y(t)$ es la salida medible, $u(t)$ la entrada de control y d la perturbación (considerada constante). En un sistema mecánico, las variables representan:

- y: posición, $\dot{y}$: velocidad, $\ddot{y}$: aceleración
- k_1: Viscosidad (constante de amortiguamiento), k_2: Constante de Hooke (resorte)
- u: Fuerza externa aplicada sobre el objeto
- d: Términos no considerados en el modelo

En un *Controlador Proporcional (P)*, la señal de control está dada por

$$u = -k_\mathrm{p} y, \tag{2.25}$$

donde k_p es la *constante proporcional* del controlador. Si analizamos el sistema en lazo cerrado en el dominio de Laplace, considerando condiciones iniciales iguales a cero, se tiene:

$$s^2Y(s) + sk_1Y(s) + k_2Y(s) = -k_\mathrm{p}Y(s) + \frac{d}{s} \tag{2.26}$$

Reagrupando términos y despejando $Y(s)$ se tiene que la salida del sistema es:

$$Y(s) = \frac{1}{s^2 + sk_1 + (k_\mathrm{p} + k_2)}\frac{d}{s} \tag{2.27}$$

Una condición necesaria y suficiente para garantizar estabilidad, según el criterio de estabilidad de Hurwitz, es que $k_1 > 0$ y $k_\mathrm{p} + k_2 > 0$. Para el análisis del error en estado permanente ϵ, se recurre al Teorema del Valor Final:

$$\epsilon = y(\infty) = \lim_{s\to 0} sY(s) = \lim_{s\to 0} \frac{d}{s^2 + sk_1 + (k_\mathrm{p} + k_2)} = \frac{d}{k_\mathrm{p} + k_2} \tag{2.28}$$

Con esto es fácil observar que el control proporcional:

- Puede garantizar estabilidad si $k_1 > 0$
- Permite ajustar el error en estado estable ϵ
- Puede ajustar uno de los coeficientes del polinomio característico, y con esto tener cierto control sobre la posición de los polos

2.4.2. Control PI

Considérese el mismo modelo que en la Ec. (2.24). El *control PI* se define como:

$$u = -k_\mathrm{p} y - k_\mathrm{i}\int_0^t y(\tau)\mathrm{d}\tau \tag{2.29}$$

donde k_p es la *constante proporcional* y k_i la *constante integral*. Siguiendo los mismos pasos de análisis, la salida en lazo cerrado del modelo resulta en:

$$Y(s) = \frac{\mathrm{d}}{s^3 + k_1 s^2 + (k_2 + k_\mathrm{p})s + k_\mathrm{i}}. \tag{2.30}$$

Se pueden notar dos cosas: la acción integral en la señal de control permite modificar un coeficiente más del polinomio característico en lazo cerrado, a costa de aumentar el orden del sistema. Según el criterio de Hurwitz, una condición necesaria pero no suficiente para garantizar estabilidad es que $k_1 > 0$, $k_2 + k_\mathrm{p} > 0$ y $k_\mathrm{i} > 0$. Es decir, aunque se tenga control sobre dos de los coeficientes del polinomio característico, no es suficiente para que se pueda garantizar estabilidad en el sistema. Por otro lado, si se analiza el error en estado estable ϵ, se tiene que:

$$\epsilon = y(\infty) = \lim_{s\to 0} sY(s) = \lim_{s\to 0} s\frac{d}{s^3 + k_1 s^2 + (k_2 + k_\mathrm{p})s + k_\mathrm{i}} = 0 \tag{2.31}$$

La ventaja de la acción integral en una señal de control es que elimina el error en estado estable, siempre y cuando el sistema sea estable.

2.4.3. Control PD

Considérese un sistema cuyo modelo está determinado por la ecuación (2.24). El *control PD* se define como:

$$u = -k_\mathrm{p} y - k_\mathrm{d}\dot{y} \tag{2.32}$$

donde k_p es la *constante proporcional* y k_d es la *constante derivativa*. Haciendo un análisis en lazo cerrado, al igual que en las secciones anteriores, se tiene que la salida del sistema es:

$$Y(s) = \frac{1}{s^2 + (k_1 + k_\mathrm{d})s + k_2 + k_\mathrm{p}} \cdot \frac{d}{s} \tag{2.33}$$

Nótese que la acción derivativa permite modificar el coeficiente que faltaba en el control P, con lo que, además de garantizar estabilidad, se pueden ubicar los polos en lazo cerrado del sistema. No obstante, al analizar el error en estado estable ϵ:

$$\epsilon = y(\infty) = \lim_{s\to 0} sY(s) = \lim_{s\to 0} \frac{d}{s^2 + (k_1 + k_\mathrm{d})s + (k_\mathrm{p} + k_2)} = \frac{d}{k_\mathrm{p} + k_2} \tag{2.34}$$

se observa que aún existe un error en estado estable diferente de cero.

Otra característica de la acción derivativa en un controlador es que tiende a amplificar el ruido. Es por eso que hay que tener cuidado a la hora de sintonizar un control PD o PID, comenzando con valores pequeños de k_d.

2.4.4. Control PID

Recapitulando: el *control P* es el más básico de todos; la señal de control, como su nombre lo indica, es proporcional al error o salida medible. Dependiendo del sistema, puede o no garantizar estabilidad y ajustar el error en estado estable. Para mejorar su desempeño, se añaden acciones integral y derivativa. La acción integral elimina el error en estado estable (siempre y cuando el sistema sea estable), pagando el costo de aumentar en un grado el orden del sistema. En cambio, la acción derivativa garantiza estabilidad e incluso permite ubicar los polos en lazo cerrado del sistema, pero donde pierde efectividad es en la eliminación del error en estado estable y la amplificación del ruido. El *controlador PID* combina lo mejor de cada uno de los controladores anteriores:

- Siempre puede garantizar estabilidad con una selección apropiada de $\{k_\mathrm{p}, k_\mathrm{i}, k_\mathrm{d}\}$
- Permite modificar todos los coeficientes del polinomio característico
- Elimina el error en estado estable

La estructura de un controlador PID es:

$$u = -k_{\mathrm{p}} y - k_{\mathrm{i}} \int_0^t y(\tau)\mathrm{d}\tau - k_{\mathrm{d}} \dot{y} \tag{2.35}$$

con $k_{\mathrm{p}}, k_{\mathrm{i}}, k_{\mathrm{d}}$ las constantes proporcional, integral y derivativa, respectivamente. De la Ec. (2.24), la salida del sistema en el dominio de Laplace es:

$$Y(s) = \frac{d}{s^3 + (k_1 + k_{\mathrm{d}})s^2 + (k_2 + k_{\mathrm{p}})s + k_{\mathrm{i}}}. \tag{2.36}$$

Analizando el error en estado estable, se confirma que:

$$\epsilon = y(\infty) = \lim_{s \to 0} sY(s) = \lim_{s \to 0} s \frac{d}{s^3 + (k_1 + k_{\mathrm{d}})s^2 + (k_2 + k_{\mathrm{p}})s + k_{\mathrm{i}}} = 0 \tag{2.37}$$

2.5. Detección de Colisión

La *detección de colisión* es un componente integral de los motores gráficos que están diseñados para proveer animaciones realistas de la interacción de los objetos virtuales. Los *tests de intersección* constituyen la columna vertebral de los algoritmos de detección de colisiones. Sin embargo, la detección de colisiones en escenas grandes requieren métodos eficientes que minimicen el número de tests. En la práctica, es común ver algoritmos que agrupan diferentes objetos virtuales cercanos en un gran volumen, para así reducir el número de cálculos de intersecciones si el objeto que puede colisionar se encuentra fuera de ese volumen; otra práctica común es particionar objetos complejos en formas geométricas simples, como triángulos y planos [21].

2.5.1. *Ray tracing*

Ray tracing o *rayo de intersección* [21] es uno de los métodos más usados para detectar colisiones; además, suele ser un componente fundamental dentro de otros algoritmos de intersección, como los basados en *volúmenes delimitadores (bounding volume)*. La idea principal es calcular el punto de intersección entre un vector (*ray*) y otros objetos geométricos simples como planos, triángulos o lineas. El vector o *rayo* es un vector que parte desde un punto P_0 y atraviesa un objeto donde puede haber una colisión. Una vez que el rayo atraviesa un objeto, se calculan los puntos de intersección P_n entre el rayo y elementos del objeto, como se muestra en la Figura 2.4. Posteriormente se calculan las distancias entre el punto inicial del vector (P_0) y los puntos de intersección y se identifica el punto con menor distancia. Este punto hallado (P_1 en la Figura 2.4) es el que se comparará con la posición del punto P_0 para determinar si existe o no una colisión.

En aplicaciones complejas, P_0 representa cualquier punto de un objeto virtual que está propenso a colisionar con otro. En el caso de esta tesis, P_0 representa la posición del efector final del robot.

Intersección entre un vector y un rayo

Dado un plano, un rayo puede o intersecarlo, o ser paralelo a él. De cualquier forma, ambas condiciones pueden ser determinadas mediante un análisis vectorial [22].

El objetivo es identificar un punto P que se encuentra sobre el plano y sobre el rayo. Sea la ecuación del plano

$$ax + by + cz + d = 0 \tag{2.38}$$

donde

$$\boldsymbol{n} = a\boldsymbol{i} + b\mathbf{j} + c\boldsymbol{k} \tag{2.39}$$

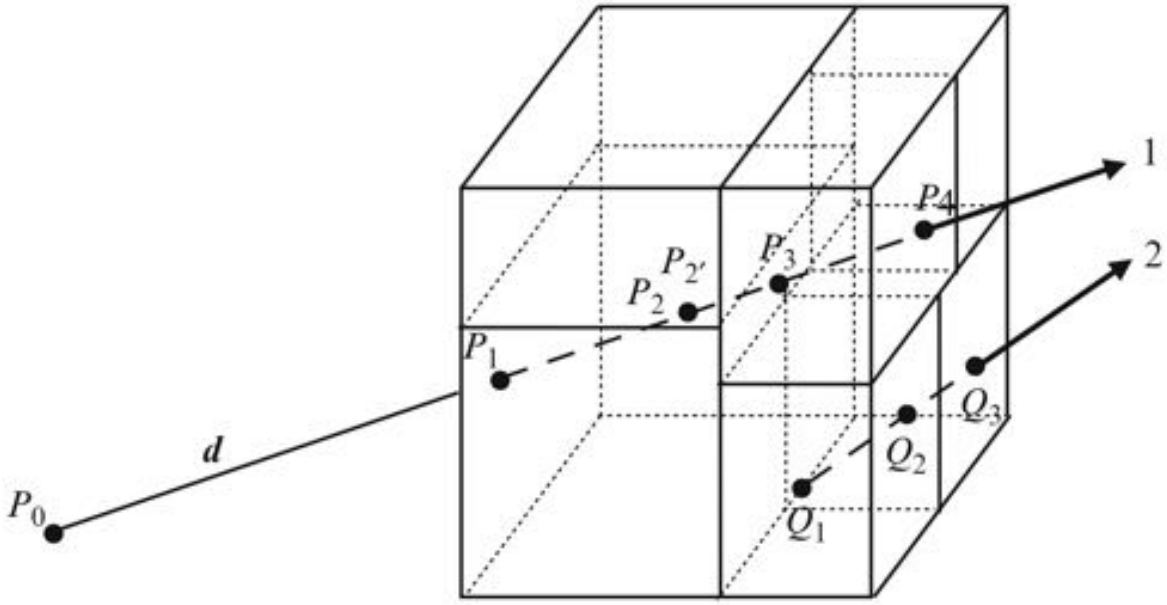


Figura 2.4: Intersección de un rayo con un volumen [21]

es el vector normal al plano. P es un punto en el plano con un vector de posición

$$\boldsymbol{p} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}, \tag{2.40}$$

entonces,

$$\boldsymbol{n} \cdot \boldsymbol{p} + d = 0. \tag{2.41}$$

Sea la ecuación del rayo

$$\boldsymbol{p}_{\mathrm{v}} = \boldsymbol{t} + \lambda \boldsymbol{v} \tag{2.42}$$

donde

$$\boldsymbol{t} = x_{\mathrm{T}}\boldsymbol{i} + y_{\mathrm{T}}\boldsymbol{j} + z_{\mathrm{T}}\boldsymbol{k} \tag{2.43}$$

es el vector de posición del origen del rayo y

$$\boldsymbol{v} = x_{\mathrm{v}}\boldsymbol{i} + y_{\mathrm{v}}\boldsymbol{j} + z_{\mathrm{v}}\boldsymbol{k} \tag{2.44}$$

el vector de dirección del rayo. Así, la línea y el plano se van a intersecar para alguna λ tal que

$$\boldsymbol{n} \cdot (\boldsymbol{t} + \lambda \boldsymbol{v}) + d = \boldsymbol{n} \cdot \boldsymbol{t} + \lambda \boldsymbol{n} \cdot \boldsymbol{v} + d = 0, \tag{2.45}$$

por lo tanto,

$$\lambda = \frac{-(\boldsymbol{n} \cdot \boldsymbol{t} + d)}{\boldsymbol{n} \cdot \boldsymbol{v}} \tag{2.46}$$

para el punto de intersección. El vector de posición para P es $\boldsymbol{p}_{\mathrm{v}} = \boldsymbol{t} + \lambda \boldsymbol{v}$. Si el producto $\boldsymbol{n} \cdot \boldsymbol{v} = 0$, significa que el vector y el plano son paralelos. En la Figura 2.5 se ilustran los distintos vectores involucrados.

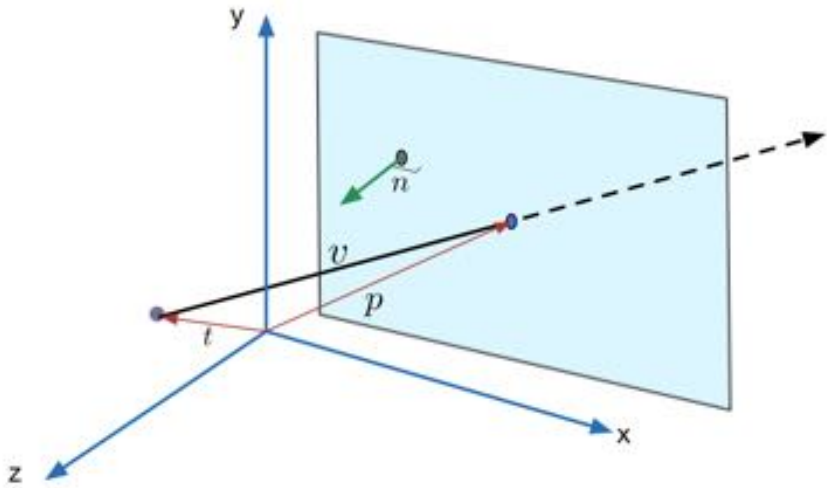


Figura 2.5: Vectores para determinar la intersección de un punto y un plano [22]

2.6. Renderizado háptico

Uno de los problemas más importantes asociados con la háptica es el cálculo de las interacciones entre el dispositivo háptico y el objeto virtual. La técnica de procesamiento de la interacción háptica en un ambiente virtual es llamada *display háptico* o *renderizado háptico* [14]. Su nivel más fundamental consiste en calcular una fuerza que será después ejecutada por un dispositivo háptico [3].

El renderizado háptico puede ser altamente interdisciplinario, combinando áreas como la robótica, física, matemáticas, métodos numéricos, computación gráfica e ingeniería de software. A pesar de esto, con las herramientas que existen hoy en día, el desarrollo de aplicaciones hápticas es muy accesible.

2.6.1. Renderizado de fuerzas

El vector de fuerza es la unidad de salida para un dispositivo háptico. Hay varias maneras de calcular este vector de tal forma que se puedan generar una variedad de sensaciones. Las tres clases de fuerzas que pueden ser simuladas son [3]:

- Dependiente del movimiento
- Dependiente del tiempo
- Una combinación de ambas

Estas fuerzas deben ser generadas a una frecuencia de al menos 1 kHz; de no ser así, la sensación puede no percibirse de una forma natural [3].

Fuerza dependiente del movimiento: Una fuerza que depende del movimiento significa que se calcula de acuerdo con la posición del dispositivo háptico. En general, el cálculo de una fuerza dependiente del movimiento puede pertenecer a una de las siguientes clases [3]

- **Resorte.** La fuerza de un resorte es probablemente el cálculo de fuerza más común en un renderizado háptico. Puede ser calculado al aplicar la Ley de Hooke ($F = k\Delta x$). El resorte se coloca con uno de los extremos anclados a un punto del objeto virtual y el otro en la posición del efector final.

- **Amortiguador.** El amortiguador es otra analogía común en el renderizado háptico. Su uso principal es para reducir las vibraciones ocasionadas por el resorte. En general, la fuerza calculada es proporcional a la velocidad del efector final. Normalmente, $F = -bv$, con b la constante de amortiguamiento y v la velocidad del efector final.

- **Fricción.** Hay varios tipos de fricción que pueden ser simulados con un dispositivo háptico. Éstos incluyen:

  - Fricción de Coulomb. Únicamente se opone al movimiento con una fuerza constante. Se calcula con $F = c \cdot \text{sgn}(v)$ donde c es una constante y $\text{sgn}(v)$ es el signo (dirección) de la velocidad.
  - Fricción viscosa. La fuerza de fricción es proporcional a la velocidad, siendo su ecuación $F = -b \cdot v$.
  - Fricción estática y dinámica. La fuerza de fricción siempre se opone al movimiento lateral a lo largo de la superficie y es proporcional a la fuerza normal a ella.

- **Fuerza inercial.** La inercia es una fuerza asociada con el movimiento de una masa. Si se conoce la trayectoria de un objeto (por ejemplo, la solución de las ecuaciones de movimiento) uno puede calcular de manera sencilla la fuerza que sentiría durante ese movimiento usando la Segunda Ley de Newton $F = m \cdot a$.

Fuerza dependiente del tiempo. Significa que el valor de la fuerza es función del tiempo. Algunos ejemplos son:

- **Constante.** Es una fuerza de magnitud y dirección constante. Generalmente se usa como compensación por gravedad.

- **Periódica.** Es una fuerza resultado de aplicar un patrón que se repite con el tiempo. Puede ser una sinusoide, diente de sierra o una onda cuadrada. La fuerza periódica se describe en términos de una frecuencia (que debe ser menor a la mitad de la frecuencia de renderizado del dispositivo) y una magnitud.

- **Impulsos.** Consiste en un vector de fuerza que es aplicado de manera instantánea (durante un periodo muy corto de tiempo).

Método del *proxy* virtual

Básicamente hay dos clases de métodos estándar que son implementados en APIs[2] de alto nivel para robots de 3 grados de libertad: métodos basados en penalización y el método del *proxy virtual* [14]. Debe hacerse notar que, a pesar de haber varios artículos concernientes al renderizado háptico para robots de 6 grados de libertad, no existe una metodología 'estándar" ampliamente usada en la práctica.

En el método del *proxy* virtual hay un punto virtual, el *God-object*, que no es capaz de penetrar en objetos sólidos. La posición de este punto es actualizada en cada ciclo del renderizado háptico con la posición del efector final. Si el efector final penetra un objeto sólido, el movimiento del *God-Object* es restringido a la superficie (no puede penetrar objetos sólidos), generando una fuerza que puede ser calculada simulando un resorte ideal sin masa con uno de los extremos del resorte anclado al *God-object* y el otro al efector final (ver Figura 2.6). Con la ley de Hooke, la fuerza generada es $F = -K(x_{ef} - x_{god-object})$, que finalmente será reproducida por el dispositivo háptico.

[2]Interfaz de Programación de Aplicaciones

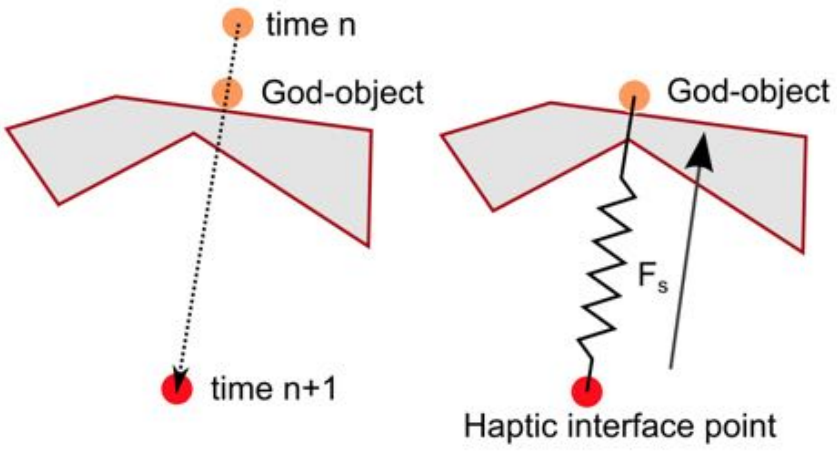


Figura 2.6: Método del *proxy* virtual [14]

2.7. Interfaz de Programación de Aplicaciones (APIs)

Una *Interfaz de Programación de Aplicaciones*, abreviada como *API* por sus siglas en inglés, es un conjunto de funciones, subrutinas, métodos (en el sentido de la POO[3]) y definiciones que ofrece cierta biblioteca (conjunto de librerías) para ser usadas por otro software como una capa de abstracción. Geomagic Touch incluye su propio API para el desarrollo de aplicaciones hápticas: *OpenHaptics*.

2.7.1. Abstracción de capas de *OpenHaptics*

Hay varios métodos para implementar una interfaz háptica en alguna aplicación, que van desde el acceso más *crudo* en los controladores del dispositivo hasta interfaces gráficas de alto nivel. Las librerías de desarrollo de *OpenHaptics* incluyen *QuickHaptics* micro API, *Haptic Device API* (*HDAPI*) y *Haptic Library API* (*HLAPI*) (Figura 2.7) [3].

QuickHaptics es un micro API que hace fácil y rápido escribir nuevas aplicaciones hápticas o añadirlas a aplicaciones existentes. Con funciones integradas de geometría y renderizado de fuerza, y al establecer una serie de parámetros predeterminados, hace posible desarrollar escenarios hápticos con un mínimo de esfuerzo. No obstante, el catálogo de fuerzas disponibles a ser renderizadas es limitado.

Las librerías de HDAPI proveen acceso de bajo nivel a los drivers del dispositivo háptico. Permite a los programadores renderizar fuerzas de manera directa, acceder a parámetros del ciclo de ejecución y utilidades para realizar un *debug* de la aplicación. No obstante, no tiene capacidades gráficas integradas. Se deben usar estas librerías si se desea implementar una ley de control personalizada para el control del robot o si se desean utilizar vectores de fuerza personalizados para simular sensaciones más complejas.

Las librerías de HLAPI son un intermedio entre las dos anteriores. Contienen funciones de alto nivel para el renderizado háptico y además son compatibles con HDAPI, por lo que se pueden usar las dos de manera combinada. Su punto fuerte es la integración con *OpenGL*, un conjunto de librerías gráficas de código abierto. *OpenHaptics* está escrito en C++.

[3]Programación Orientada a Objetos

Figura 2.7: Abstracción de capas de OpenHaptics [3]

Capítulo 3

Programación del dispositivo háptico

Para poner en marcha el dispositivo háptico Geomagic Touch, se utilizaron las librerías de *OpenHaptics*, en particular HDAPI para el control del robot y HLAPI para la parte gráfica. Como se mencionó en el Capítulo 2, se trata de dos APIs distintas que, sin embargo, pueden usarse en forma conjunta. A continuación se presenta la estructura general de una interfaz háptica con display gráfico. También se incluye una aplicación de seguimiento de trayectorias como primer acercamiento al desarrollo de una interfaz háptica y que sirve de base para desarrollos posteriores.

3.1. HDAPI

HDAPI consiste de dos componentes principales: el *dispositivo* y el *planificador* [3]. La abstracción del dispositivo permite utilizar cualquiera de los dispositivos hápticos soportados (para una lista detallada, ver el archivo *Readme*) con las librerías. Las *funciones de retrollamada (callbacks)* permiten al programador ingresar comandos que van a ser ejecutados por el *hilo del servo*. El hilo del servo es el hilo de ejecución encargado de establecer la comunicación con el dispositivo háptico. Se ejecuta de forma independiente de, por ejemplo, un hilo gráfico con el fin de asegurar el correcto funcionamiento del dispositivo háptico a pesar de cualquier retardo o error que pueda afectar el desempeño de las demás partes de un programa, como son cálculos numéricos, display gráfico o una interfaz de usuario.

Las funciones típicas de HDAPI son inicializar el dispositivo, controlar el planificador y ejecutar comandos en el robot. En una aplicación básica, los pasos a seguir para poner en marcha al robot son:

1. Inicializar el dispositivo
2. Crear una función *callback* que adquiera los parámetros del robot y escriba valores en los actuadores
3. Habilitar el renderizado de fuerzas
4. Inicializar el planificador
5. Limpiar los parámetros del dispositivo y datos del planificador al cerrar la aplicación

Además de los pasos anteriores, es necesario escribir funciones que aseguren la integridad del equipo, revisando constantemente picos de fuerza, sobrevelocidad en los eslabones,

parámetros de temperatura en el motor, etc. Adicionalmente, se pueden configurar parámetros como la frecuencia del hilo del servo, aunque se recomienda que no sea menor a 1000 Hz para asegurar una realimentación de fuerza estable y una sensación más *natural*.

Todas las acciones llevadas a cabo por el dispositivo háptico, en cada ciclo de ejecución del hilo del servo, deben estar contenidas dentro de un *frame háptico*. El frame háptico debe estar implementado, a su vez, dentro de una función *callback* principal, la cual será la encargada de la lectura y escritura de los parámetros del robot. Los límites de un frame háptico se definen con los comandos *hdBeginFrame()* y *hdEndFrame()*. Limitadas por estas dos líneas de código, deben estar todas las instrucciones para obtener y establecer parámetros del robot (*hdGet() y hdSet()*). Puesto que C++ es un lenguaje que permite la programación modular, es posible acceder a los parámetros del robot desde cualquier parte del código; no obstante, para evitar un mal funcionamiento, cuando se manda a llamar a una función *hdGet()* fuera del frame háptico, lo que se obtiene no es la lectura en *tiempo real* del parámetro, sino una copia del último valor obtenido dentro del frame. En el caso de la escritura de fuerzas sucede algo similar, con la diferencia de que cada vez que se manda a llamar a la función *hdSet()*, las fuerzas resultantes se van acumulando para que, al final del frame háptico, se escriba una sola fuerza y no una sucesión de estas. La estructura general[1] de un programa desarrollado en HDAPI es:

```
1  INICIALIZAR DISPOSITIVO HAPTICO hdInitDevice
2  HABILITAR EL RENDERIZADO DE FUERZAS hdEnable(HD_FORCE_OUTPUT)
3  PROGRAMAR EL CALLBACK PRINCIPAL hdScheduleAsynchronous
4  INICIALIZAR EL PLANIFICADOR hdStartScheduler
5  WHILE(APLICACION NO FINALIZADA)
6          INICIAR FRAME HAPTICO hdBeginFrame
7          OBTENER POSICION DEL ROBOT hdGet(HD_CURRENT_POSITION)
8          FOR n Objetos Virtuales
9                  IF hay colision
10                         CALCULAR FUERZA DE REACCION
11                 ENDIF
12         ENDFOR
13         ESCRIBIR FUERZA RESULTANTE hdSet(HD_CURRENT_FORCE)
14         FINALIZAR FRAME HAPTICO hdEndFrame()
15 ENDWHILE
16 DETENER EL PLANIFICADOR hdStopScheduler()
17 DESHABILITAR DISPOSITIVO HAPTICO hdDisableDevice()
```

3.2. HLAPI

HLAPI es un API de alto nivel escrito en C para el renderizado háptico que incluye una parte gráfica desarrollada con *OpenGL*. Con HLAPI el programador puede especificar estructuras primitivas (puntos, líneas y triángulos) y ligarlas a una interfaz háptica [3]. En otras palabras, permite añadir gráficos a través de *OpenGL* a la interfaz háptica.

HLAPI es capaz de renderizar gráficos usando todas las capacidades que ofrece *OpenGL*. Además, su principal ventaja es que puede *reutilizar* código de otras aplicaciones, lo que hace de esta librería una poderosa herramienta para añadir interfaces hápticas a aplicaciones gráficas existentes. La estructura básica de un programa en HLAPI es la siguiente:

[1] El código completo se encuentra en el CD adjunto.

```
1 CONFIGURACION DE OpenGL
2 CONFIGURACION DE HLAPI
3 MAPEO DE COORDENADAS DEL DISPOSITIVO A COORDENADAS DE OpenGL
4 WHILE(TRUE)
5         PROCESAR EVENTOS
6         RENDERIZADO GRAFICO
7         RENDERIZADO HAPTICO
8 ENDWHILE
```

A diferencia del renderizado háptico, el renderizado gráfico no necesita ejecutarse a una frecuencia tan alta. Para observar fluidez en la animación, el hilo gráfico de la aplicación debe ejecutarse a una frecuencia aproximada de 60 Hz. Con las librerías de HLAPI no hace falta crear explícitamente un hilo de ejecución extra para la parte gráfica, ya que éstas se encargan de hacerlo. Aún así, se recomienda crear un hilo de ejecución adicional para la detección de colisiones, si es que consume demasiados recursos.

Al tratarse de dos hilos de ejecución diferentes (el del robot y el de la parte gráfica), es muy importante la forma en que se manipulan los datos, especialmente porque los hilos corren a velocidades muy distintas. Hay que definir bien qué acciones se ejecutan en cada hilo, de lo contrario, el resultado puede no ser el esperado. Generalmente el cálculo de fuerzas, si no se realiza en el hilo del servo, se lleva a cabo en un hilo diferente al de display gráfico. De cualquier forma, es indispensable que una interfaz háptica con display gráfico sea una aplicación *multi-hilo*.

3.3. Seguimiento de trayectorias

El seguimiento de trayectorias es uno de los objetivos de control. En este caso, se trata de dirigir al efector final a través de una secuencia de puntos en el espacio.

3.3.1. Cálculo de la trayectoria

La generación de una trayectoria consiste en calcular una secuencia de puntos que sirvan como referencia al efector final. Puesto que se trata de un sistema discreto, para observar *suavidad* en la trayectoria se necesitan generar la mayor cantidad de puntos posibles. *Trayectoria* no es lo mismo que *camino*: un camino es únicamente la descripción geométrica del movimiento. Por otra parte, una trayectoria es un camino en el cual se incluye una descripción temporal [17].

Las trayectorias se pueden especificar, para el caso de los robots, en términos de variables articulares o variables del espacio de trabajo -normalmente cartesianas-. Las librerías de *OpenHaptics* nos permiten trabajar directamente en el espacio de trabajo, de tal forma que la trayectoria programada para este ejemplo se puede describir con las ecuaciones paramétricas de una circunferencia en el espacio:

$$\begin{aligned} x &= r\cos(\omega t) \\ y &= r\sin(\omega t) \\ z &= 40 \end{aligned} \tag{3.1}$$

donde r es el radio, ω la frecuencia angular y t el tiempo. El cálculo de los puntos se realiza en el *hilo del servo* en tiempo real para asegurar un movimiento fluido del robot.

3.3.2. Display gráfico

En la función *callback* principal, se guarda el estado completo del robot en una estructura de datos fija, accesible a través de *apuntadores* (direcciones de memoria). Por medio de esta estructura, una función de animación (que se ejecuta a 60 Hz en el hilo gráfico) puede acceder a la información del efector final. Con las librerías de HDAPI se puede guardar directamente, en un arreglo de 16 elementos, la matriz de transformación del efector, que contiene su posición y orientación. En el hilo gráfico se multiplica esta matriz de transformación por la matriz de transformación del objeto -en nuestro caso, un cono- que va a representar gráficamente al efector final y posteriormente se realiza una transformación de coordenadas, de tal manera que el efector final y su espacio de trabajo quedan representados en un ambiente virtual.

Para dibujar la trayectoria, simplemente se dibujan líneas entre la sucesión de puntos por los que pasa el efector final.

3.3.3. Controlador

El objetivo del controlador es calcular las fuerzas necesarias en el robot de tal forma que el efector final siga la trayectoria deseada. Una vez más, *OpenHaptics* ayuda al poder definir las fuerzas directamente en coordenadas del espacio de trabajo, y no variables articulares. Las funciones incluidas en *OpenHaptics* hacen uso del método de control del *Jacobiano inverso* (ver Apéndice B): en él, se comparan la posición del efector final en el espacio operacional $\boldsymbol{\chi}_e$ con el punto deseado de la trayectoria $\boldsymbol{\chi}_d$ y la diferencia $\Delta\boldsymbol{\chi}$ se transforma a variables articulares $\Delta\boldsymbol{q}$ mediante el inverso del Jacobiano del manipulador. Una vez en variables articulares, se pueden calcular las fuerzas generalizadas a través de una matriz de ganancias [17] (ver Figura 3.1). En esta aplicación, se implementó un controlador PID lineal y variaciones de este para ejemplificar las acciones integral y derivativa. La señal de control es de la forma:

$$\boldsymbol{u} = -k_{\mathrm{p}}\boldsymbol{e} - k_{\mathrm{i}}\int_0^t \boldsymbol{e}(\tau)\mathrm{d}\tau - k_{\mathrm{d}}\dot{\boldsymbol{e}} \tag{3.2}$$

con

$$\boldsymbol{e} = \boldsymbol{\chi}_{\mathrm{d}} - \boldsymbol{\chi}_{\mathrm{ef}} \tag{3.3}$$

la señal de error. La integración del error se realizó utilizando la aproximación de Euler

$$\boldsymbol{e}(t_0 + \Delta t) = \boldsymbol{e}(t_0) + f(x_0, t_0)\Delta t \tag{3.4}$$

con $f(x_0, t_0)$ el valor actual del error. Adicionalmente, se implementó una ventana de seguridad, que no es más que una función saturación de la forma:

$$\boldsymbol{e}(t_0 + \Delta t) = \begin{cases} -\boldsymbol{e}_{sat} & \text{si } \boldsymbol{e} < -\boldsymbol{e}_{sat} \\ \boldsymbol{e}_{sat} & \text{si } \boldsymbol{e} > \boldsymbol{e}_{sat}. \end{cases} \tag{3.5}$$

Para calcular la acción derivativa, también se utilizó la aproximación de Euler, que aproxima la derivada mediante

$$\dot{\boldsymbol{e}} = \frac{[\boldsymbol{e}(t_0 + \Delta t) - \boldsymbol{e}(t_0)]}{\Delta t}. \tag{3.6}$$

En ambos casos, se consideró $\Delta t = 0.001$. La sintonización del controlador PID condujo a las siguientes constantes:

- $k_{\mathrm{p}} = 0.15$

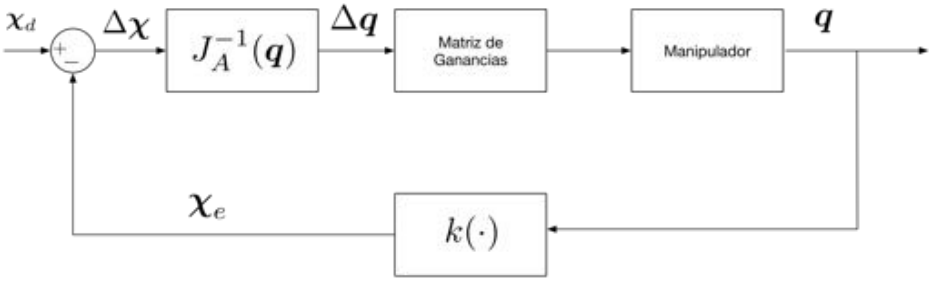


Figura 3.1: Diagrama de bloques del Control del Jacobiano Inverso

- $k_{\mathrm{i}} = 0.5$
- $k_{\mathrm{d}} = 0.001$

A continuación se compara el desempeño de los controladores P, PI, PD y PID. En cada caso, se aplicó una perturbación de 0.8 N en el eje Y cada 21 segundos aproximadamente.

Control P: para la trayectoria descrita en (3.1), con una velocidad angular de $\omega = 0.3$ rad/s, un radio $r = 40$ mm y haciendo $k_{\mathrm{i}} = 0$ y $k_{\mathrm{d}} = 0$ se obtuvieron los errores de la Figura 3.2.

Control PD: para las mismas condiciones que en el control P, haciendo únicamente $k_{\mathrm{i}} = 0$, se obtuvieron los errores de la Figura 3.3.

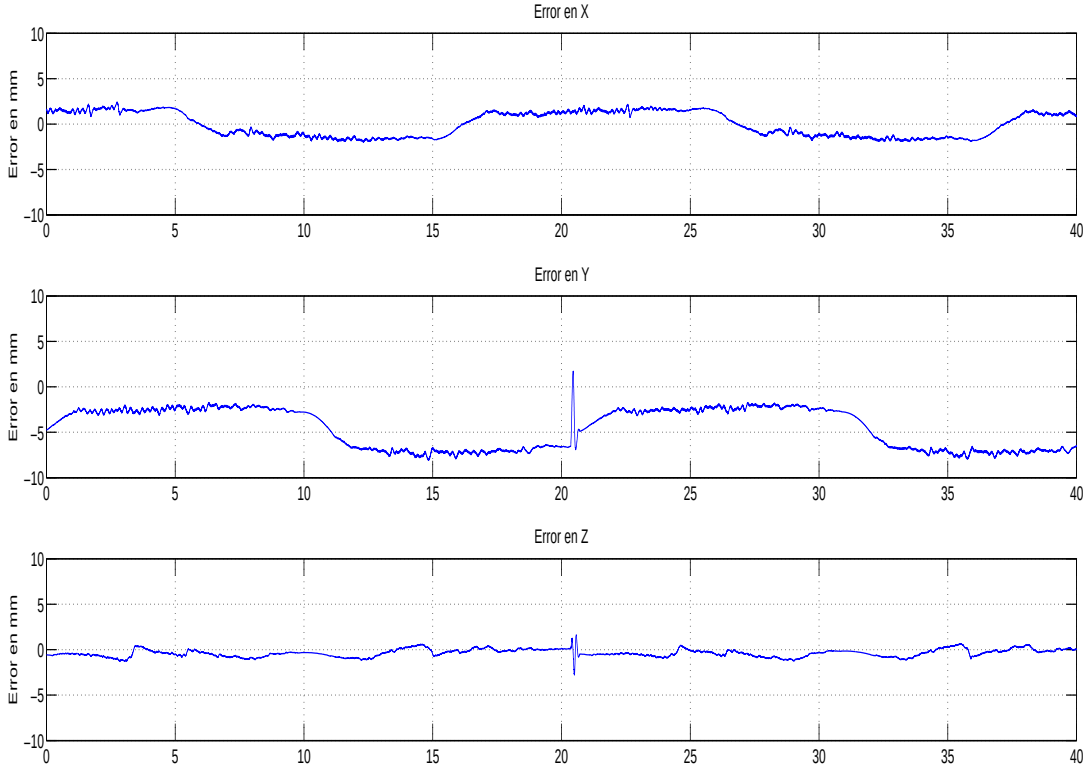
Control PI: en el control PI, se elimina la acción derivativa ($k_d = 0$). Con las mismas condiciones anteriores, se obtuvieron los errores de la Figura 3.4.

Control PID: en el control PID, se toman en cuenta las tres acciones: Proporcional, Integral y Derivativa. Usando los valores para las constantes especificados anteriormente, se obtuvieron los errores de la Figura 3.5.

Conclusiones

En general, los controladores de mejor desempeño fueron el PI y el PID. Los controles P y PD tuvieron un error en estado estable de 4 mm en promedio para el eje Y. Se observa de las gráficas la acción integral que elimina el error en estado estable, que además se puede apreciar en el trazado de las trayectorias. En los controles P y PD, por ejemplo, el error se aprecia a simple vista en el dibujo de la trayectoria, que evidentemente no tiene forma de circunferencia. En cuanto a los controles PI y PID, el desempeño es ligeramente mejor en el PID, que pudo controlar de mejor manera el efecto de la perturbación en el eje Z.

Pese a ser un sistema no lineal, el uso de un controlador lineal se justifica gracias a que OpenHaptics realiza las operaciones necesarias para transformar un torque resultante en el espacio de operación, en un vector de torques para cada uno de los actuadores, a través del Jacobiano del robot. Con esto se puede ver al robot como un sistema SISO -desde el punto de vista de programación-.



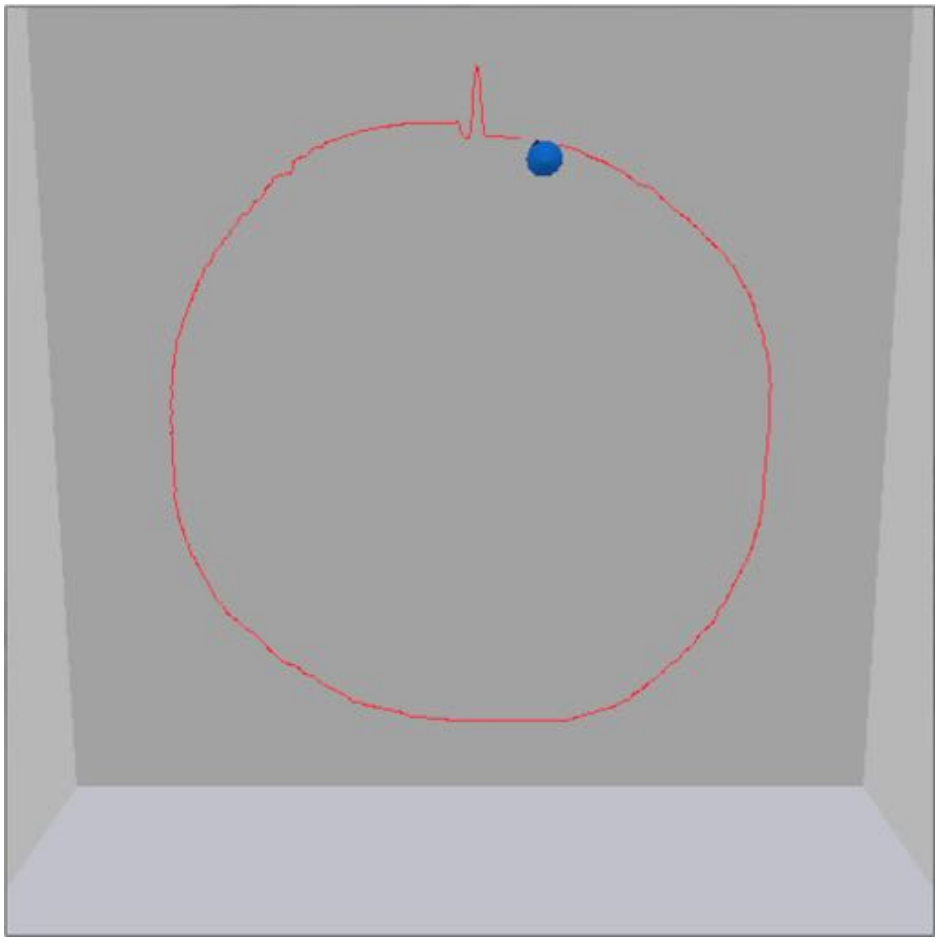
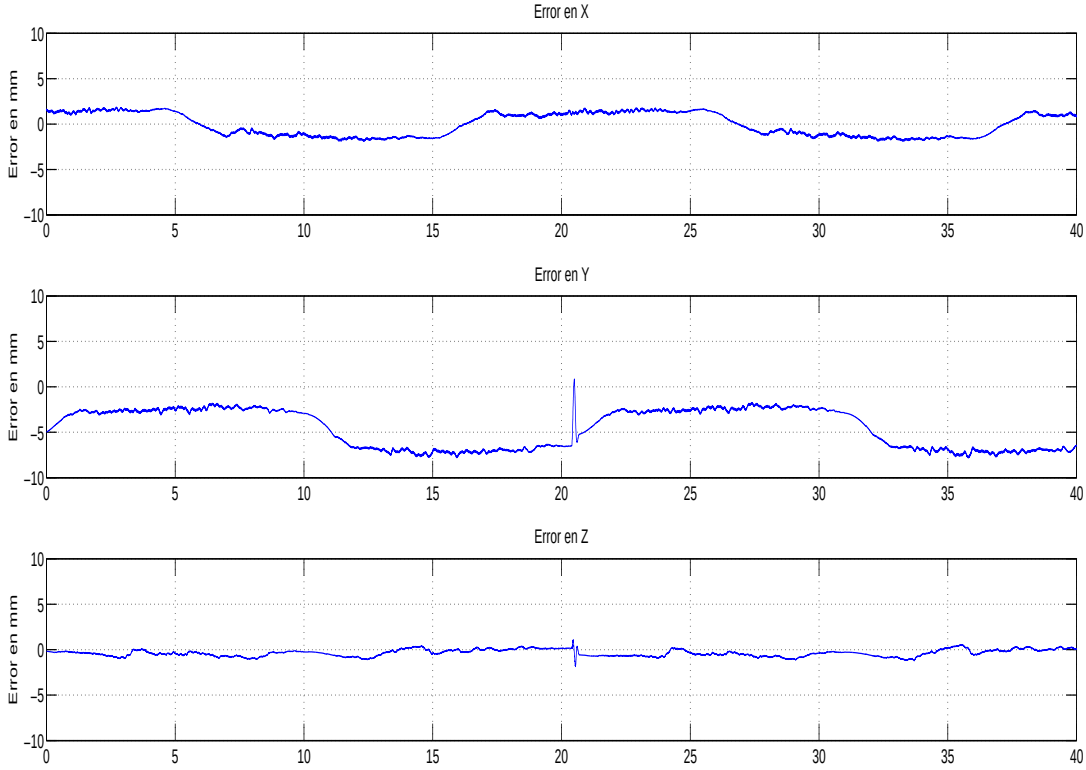


Figura 3.2: Gráfica de errores para el control P



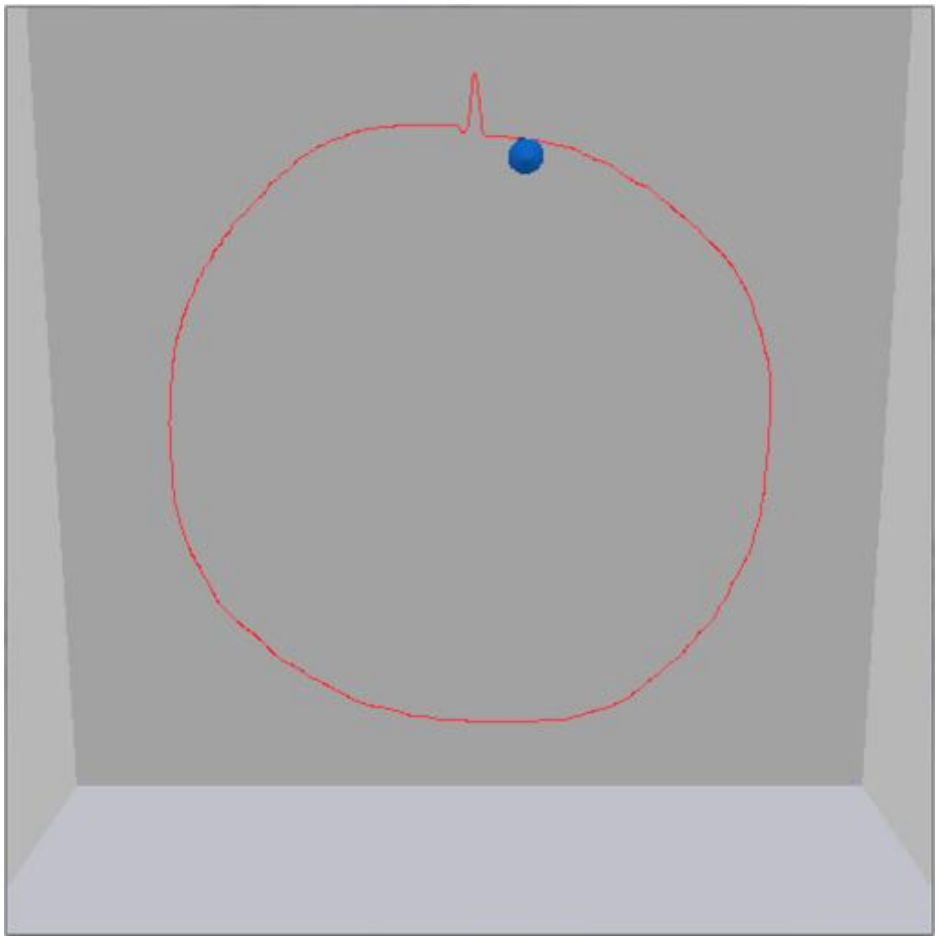
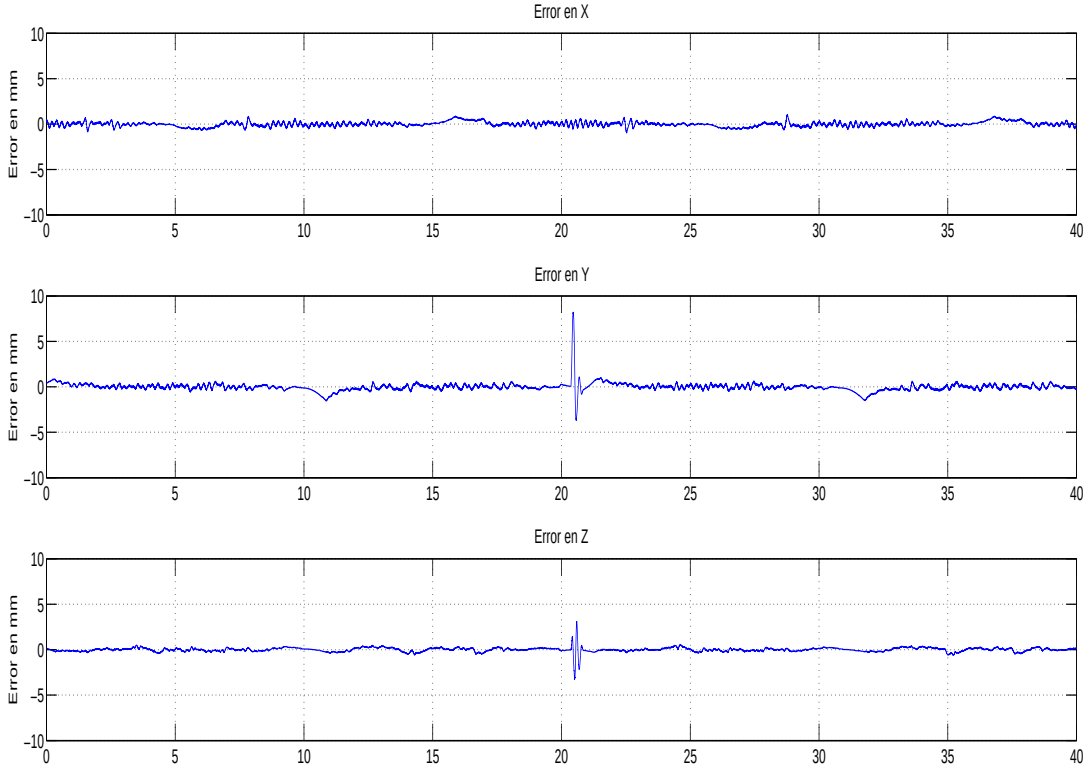


Figura 3.3: Gráfica de errores para el control PD



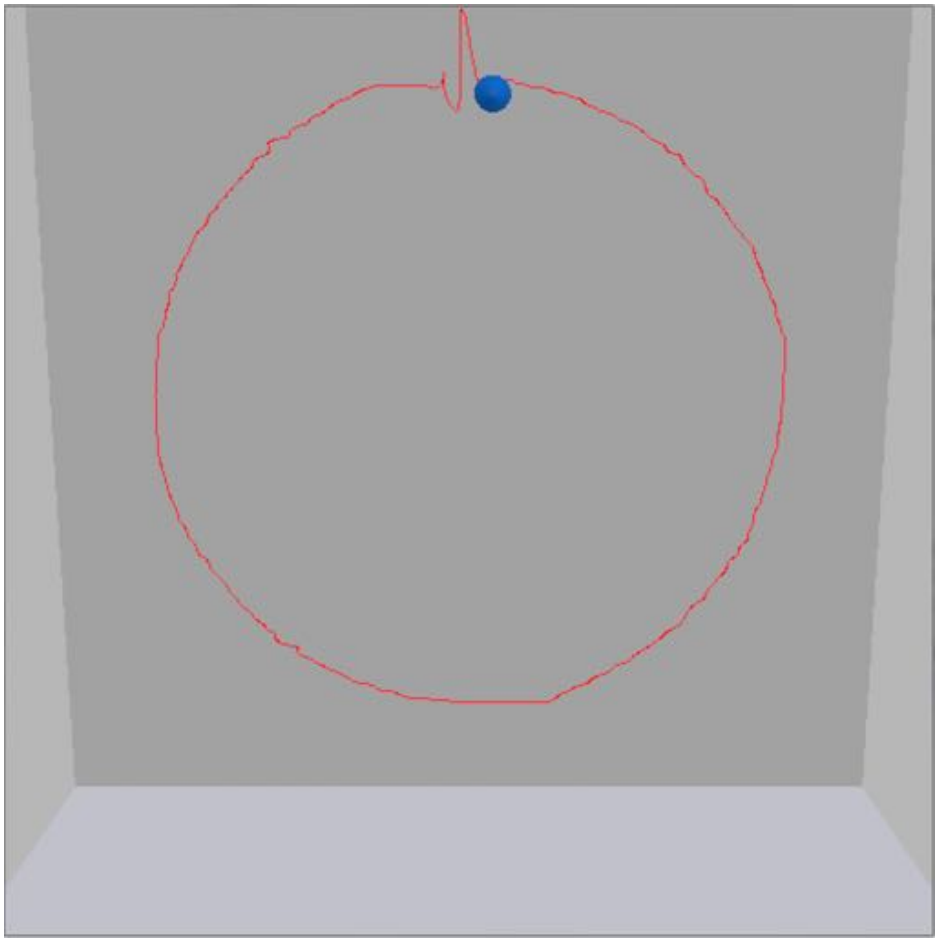
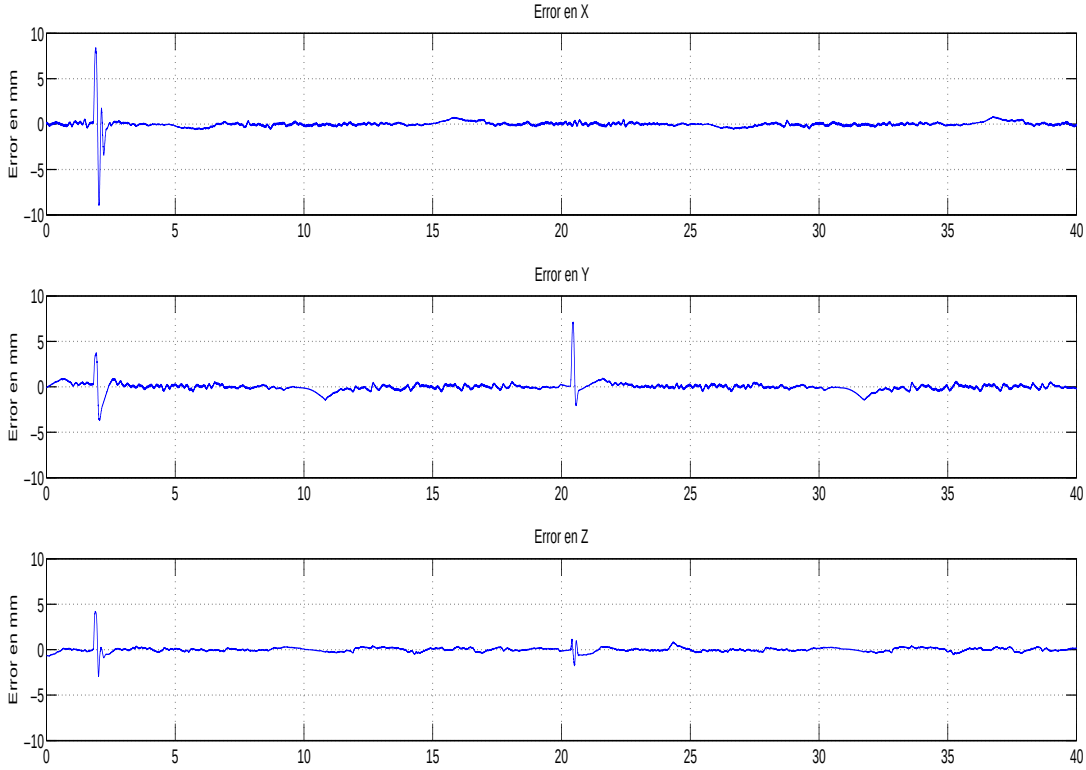


Figura 3.4: Gráfica de errores para el control PI



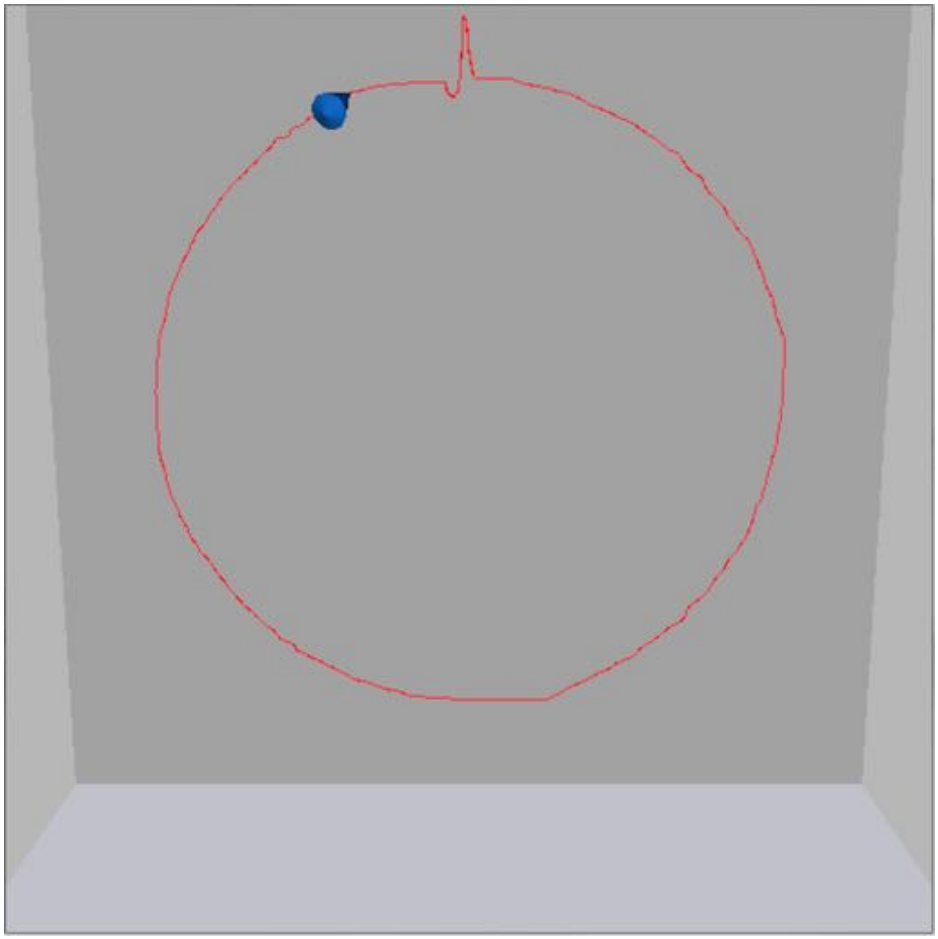


Figura 3.5: Gráfica de errores para el control PID

Capítulo 4

Interacción con un ambiente virtual

Para que un dispositivo háptico pueda interactuar con un ambiente virtual, se deben resolver tres cuestiones principales:

1. El modelado y display gráfico del ambiente virtual
2. El control del dispositivo háptico
3. La interacción entre ambos

En el Capítulo 2 se desarrolló el modelo dinámico del objeto virtual sujeto a restricciones y su solución. Adicionalmente, en el Capítulo 3, se mostró la estructura general de una interfaz háptica con gráficos y se ejemplificó mediante una aplicación de seguimiento de trayectorias. Con esto se tienen resueltas las dos primeras cuestiones. Para la tercera cuestión, se recurre al uso de *algoritmos de colisión*.

4.1. Estructura de la aplicación

OpenGL es un conjunto de librerias de computación gráfica. A diferencia de otras librerías, como *XNA* de *Microsoft*, no almacena información de los objetos que se dibujan: una vez que realiza el display gráfico, la información del objeto se vuelve inaccesible, lo que vuelve más complicada la detección de colisiones. Es así que el primer paso es crear estructuras de datos que almacenen la información del objeto virtual y que sean accesibles desde cualquier parte del programa. Junto con las variables del objeto, se deben crear métodos que las actualicen después de realizar una transformación al objeto. Esto se logra a través de la creación de *Clases*, que en programación representan un nuevo tipo de dato.

Se realizó una clase *Box*, que contiene todos los métodos y variables que son útiles para implementar el algoritmo de colisión. Contiene un método *InitializeBox()* que dibuja la caja utilizando *OpenGL* e inicializa sus parámetros y un método *UpdateComponents()* que actualiza los parámetros del objeto después de una transformación (trasladar o rotar). Entre sus parámetros están los planos que la conforman (recordemos que en computación gráfica todo se dibuja a partir de triángulos o planos), los vectores normales a estos, un tamaño y su posición.

Plane, Point y *Vector* son clases adicionales que definen la estructura de cada uno de estos elementos geométricos, así como las operaciones que pueden realizar entre ellos. Son utilizadas por la clase *Box* para caracterizar al objeto.

Para implementar el algoritmo de colisión, se escribieron dos clases: *Intersection* y *HapticRenderer*. La primera se encarga de calcular los puntos de intersección entre un *rayo* que

va desde el efector final hasta el centro del objeto virtual (ver Capítulo 2) y especificar a qué plano pertenece. Posteriormente, *HapticRenderer* utiliza esta información para determinar si hay colisión y, si es así, calcular la fuerza de reacción debido a ésta. La forma en que *HapticRenderer* determina si hay una colisión entre el efector final y el objeto virtual es:

1. Determinar la distancia entre el efector final y el punto de intersección en el plano
2. Si la distancia es menor a una distancia de umbral d_0, se considera que hay colisión y se pone en marcha el método del *proxy* virtual para generar la fuerza de reacción.

Posteriormente, si el *proxy* virtual generó alguna fuerza diferente de cero, se resuelven las ecuaciones diferenciales del objeto y se actualiza su posición.

Es natural entonces mandar llamar a las funciones de las clases *HapticRenderer* e *Intersection* desde el hilo del servo y a las funciones de la clase *Box* desde ambos hilos. Para comunicar los hilos, se utiliza la infraestructura de *OpenHaptics*. En el hilo del servo (en la función *callback* principal) se resuelven las ecuaciones diferenciales (2.21) utilizando el método de Runge-Kutta de 4°orden y se actualiza la posición del objeto virtual (ver Figura 4.1). Desde el hilo gráfico se puede acceder a la información de la clase *Box* a través de *pointers*. Con la nueva posición, el objeto se redibuja en cada ciclo (a 60 Hz).

De manera general, el pseudocódigo para la interacción es el siguiente:

```
1  //A ejecutarse en el hilo del servo
2  Definir una distancia de umbral d_0
3  WHILE(TRUE)
4    Generar un vector entre el efector final y el centro del objeto
5    Calcular los puntos de interseccion
6    Calcular la distancia entre el efector final y las intersecciones
7    IF distancia < d_0 THEN
8      Ejecutar metodo de proxy virtual
9      IF fuerza generada F != 0 THEN
10       WHILE(Velocidad del objeto != 0 OR F!=0)
11         Resolver ODEs
12         Actualizar la posicion del objeto virtual
13       ENDWHILE
14     ENDIF
15   ENDIF
16 ENDWHILE
```

Naturalmente, el cálculo de las fuerzas debido a las restricciones se hace dentro de la solución de las ecuaciones diferenciales.

4.2. Resultados experimentales

Utilizando el reloj del CPU, se determinó la frecuencia del hilo del servo en 860.75 Hz y la del hilo gráfico en 60.24 Hz. Aunque en [3] especifican una frecuencia mínima de 1000 Hz, la percepción táctil en la interacción fue satisfactoria, percibiéndose de manera natural. El experimento consistió en aplicar una fuerza con el dispositivo háptico en una de las caras del objeto virtual (ver Figura 4.2). Para observar el efecto de las restricciones holonómicas, se cambió la orientación del efector de tal manera que el objeto *sintiera* una fuerza aplicada en el eje Y y tratara de violar la restricción. El objeto virtual se definió con una masa $m = 0.4$ kg, un coeficiente de fricción viscosa $b = 0.175$ Nm/s, un coeficiente de fricción de Coulomb de $f_c = 1.7$ y un coeficiente de fricción estática $f_{e_0} = 0.75$.

En la Figura 4.3 se presentan los valores del ángulo relativo entre el efector final y la superficie de contacto, el valor de la fuerza[1] en el eje Y y los valores de $\boldsymbol{\lambda}$. Se puede apreciar

[1]Todas las fuerzas mencionadas son calculadas y no resultado de mediciones de fuerza ejercidas por el dispositivo háptico.

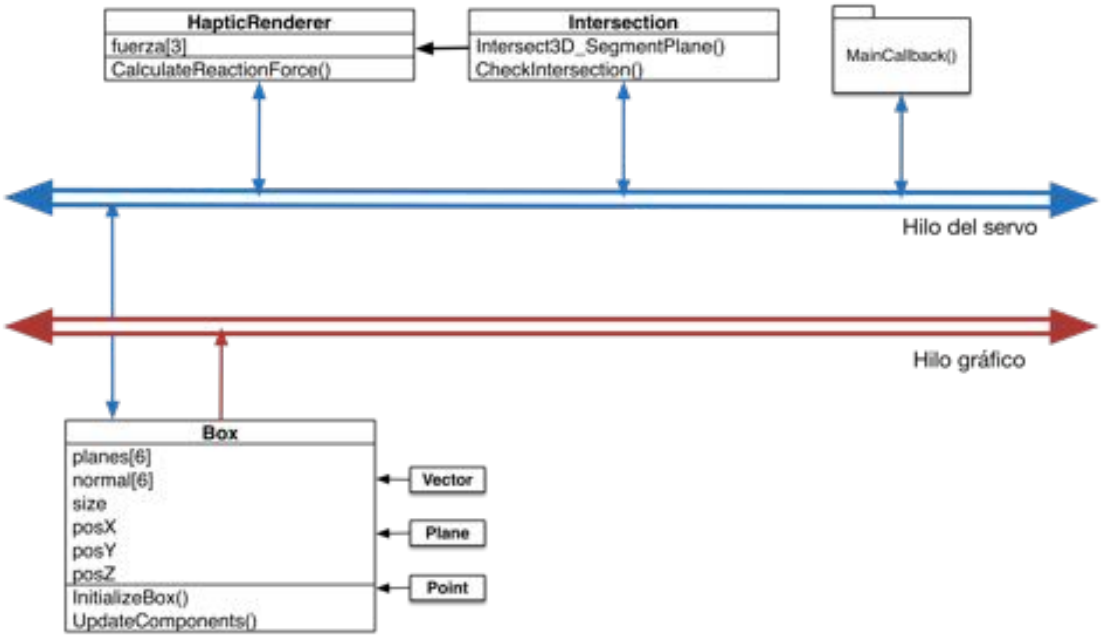


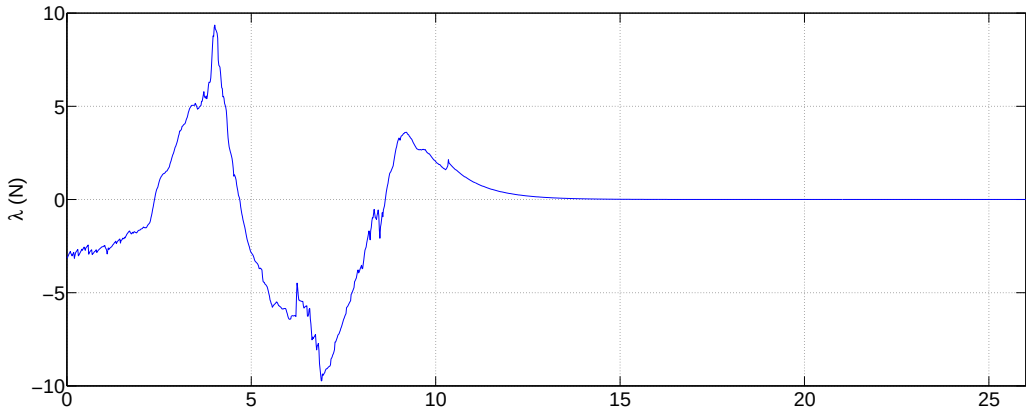
Figura 4.1: Comunicación entre clases e hilos de programación

Figura 4.2: Interacción del operador con el ambiente virtual

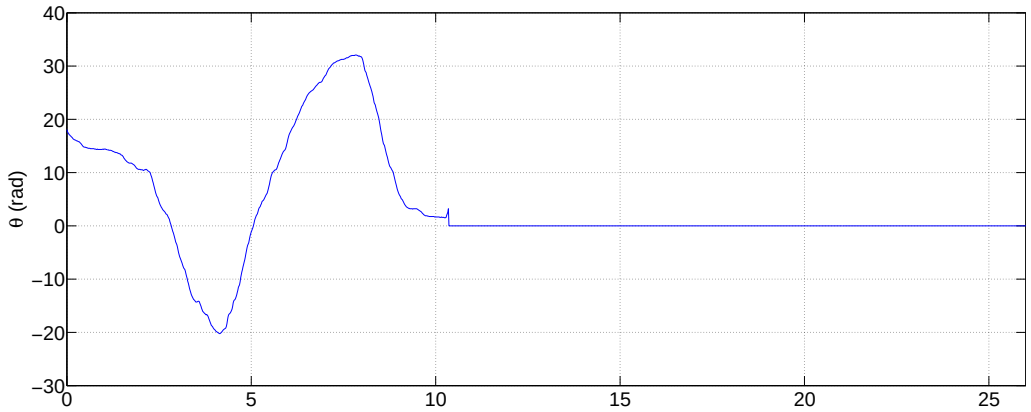
una relación directa entre el ángulo relativo y la magnitud de la fuerza de reacción debido a la restricción. Para efectos prácticos, cuando el efector final perdía contacto con el objeto virtual, se establecía el valor de θ en cero. Como se puede apreciar de la figura, la fuerza debido a la restricción persiste como resultado del movimiento del objeto, aunque no por mucho tiempo.

En la Figura 4.4 se observan la fuerza ejercida por el efector final en el objeto virtual, la posición y la velocidad del objeto virtual. Se observa una disminución exponencial de la velocidad al momento de dejar de aplicarse una fuerza sobre el objeto, como habría de esperarse.

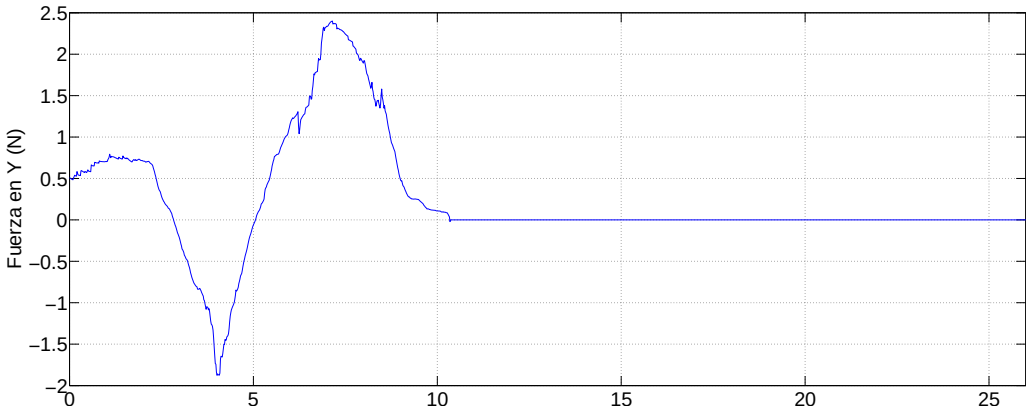
Otro punto interesante se encuentra en $t \approx 4$ s. De la Gráfica 4.4 (a) se observa un pico de fuerza de alrededor de 5 N con una orientación de aproximadamente 20° que produce una fuerza en el eje Y de casi 2 N. Si observamos la fuerza de reacción debido a la restricción, vemos que llega casi a los 10 N, mucho mayor que la fuerza aplicada por el efector final. Esto se explica con la dinámica del objeto, que se encuentra en movimiento y donde su inercia se suma a la fuerza aplicada para tratar de violar la restricción. Una vez que deja de aplicarse fuerza sobre el objeto, la fuerza debido a la restricción disminuye drásticamente, hasta llegar a ser muy cercana a cero un momento después.



(a) Fuerza debida a la restricción

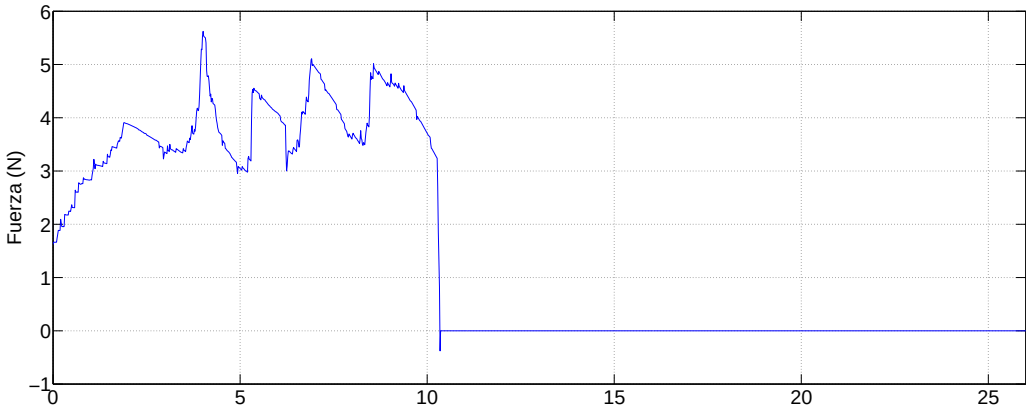


(b) Ángulo θ relativo entre el efector final y la superficie de contacto

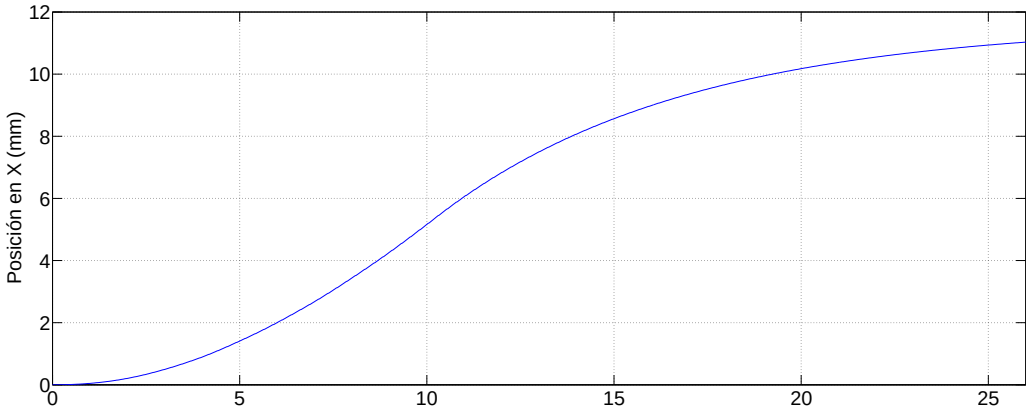


(c) Fuerza ejercida en el eje Y

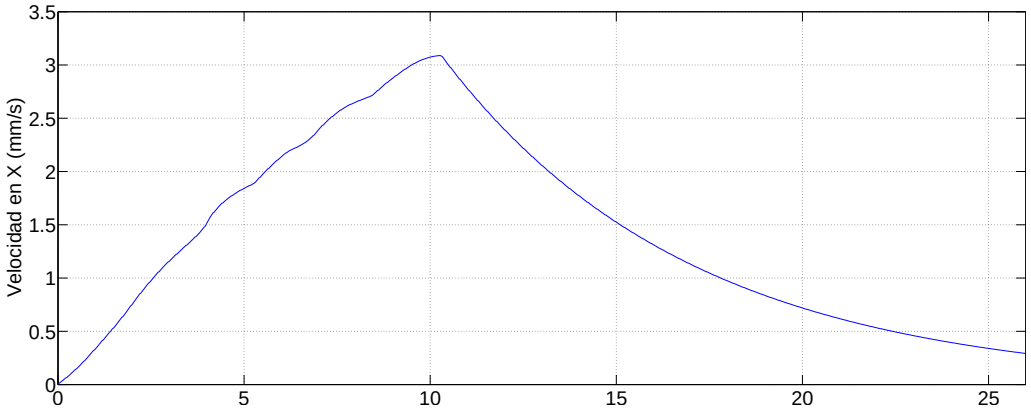
Figura 4.3: Fuerza debido a la restricción, ángulo relativo del efector final y fuerza ejercida en el eje Y



(a) Fuerza ejercida por el efector final



(b) Posición del objeto



(c) Velocidad del objeto

Figura 4.4: Fuerza del efector final, posición y velocidad del objeto virtual

Capítulo 5

Sistema de teleoperación

El objetivo de un sistema de teleoperación es realizar tareas en un ambiente o contexto diferente al que nos encontramos. Generalmente, se trata de ambientes peligrosos o no aptos para un humano, como zonas con altos niveles de radioactividad o espacios de difícil acceso. Sin embargo, cada vez es más común ver sistemas de teleoperación con el propósito de *trasladarnos* a otro lugar. Tal es el caso de cirugías a distancia, donde el médico puede estar a kilómetros del paciente. Otra razón de utilizar sistemas de teleoperación es para escalar las capacidades humanas, *e.g.*, cargar objetos de gran tamaño y peso o trabajar en zonas muy pequeñas. No obstante, sea cual sea la aplicación de un sistema de teleoperación, este va a consistir de 5 sistemas interactivos [6, 23]:

- Operador
- Dispositivo maestro
- Canal de comunicación
- Dispositivo esclavo
- Entorno

Tomando como ejemplo al robot *da Vinci* [13], el cirujano es el operador que manipula la consola, que es el *maestro*; las herramientas del robot son el *esclavo* que manipula los órganos del paciente. La tarea del sistema de teleoperación es transmitir y ejecutar los movimientos realizados por el operador, de un modo que se sienta natural. Un sistema de teleoperación ideal se comporta de manera que se vuelve *invisible* al operador y al entorno [6, 23]. En otras palabras, el operador *siente* directamente el entorno. Esta tarea es imposible para la tecnología actual, por razones que pueden resumirse a la alta complejidad del cuerpo humano: su sistema sensorial tiene un ancho de banda muy amplio y puede percibir cambios microscópicos en la textura, dureza, fuerza o movimiento. Otra limitante importante es el canal de comunicación, el cual introduce retardos entre el movimiento del operador y el del sistema esclavo.

En la Figura 5.1 se da una ejemplo intuitivo de un sistema de teleoperación. La tarea a realizar es agarrar un objeto, lo cual podemos hacer fácilmente con nuestra mano. Otra forma de tomar el objeto es a través de una herramienta, como unas pinzas. En un nivel superior, la herramienta y el objeto no están situados en el mismo entorno que nosotros, por lo que hacemos uso de tres subsistemas interconectados por algún medio de comunicación. Dichos subsistemas son:

- Un sistema maestro
- Un controlador

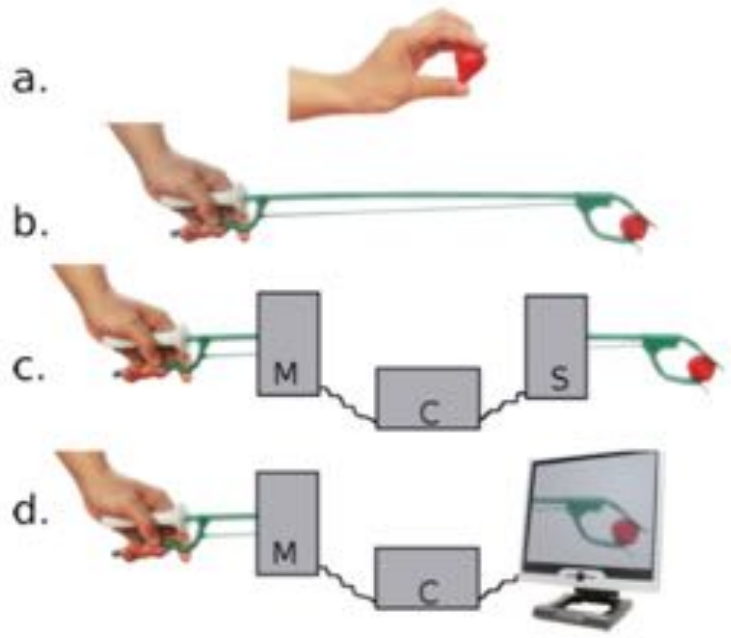


Figura 5.1: Diferentes formas de realizar una tarea [6]

- Un sistema esclavo

Se tiene un cuarto caso, donde el sistema maestro-esclavo se utiliza para manipular objetos en ambientes virtuales; en este caso, el efector final y el objeto en el entorno del sistema esclavo son modelados virtualmente.

5.1. Esquemas de Control

Se han desarrollado varios esquemas de control para lidiar con los distintos problemas que se presentan en un sistema de teleoperación. Cada uno de ellos tiene características diferentes y dan buenos resultados para algunos aspectos del sistema de teleoperación, pero difícilmente pueden dar buenos resultados para todos.

En [6] se mencionan los tres esquemas de teleoparación más comunes. Estos son:

- Control Posición Posición (PP)
- Control Posición Fuerza (FP)
- Control de 4 Canales (4Ch)

El **control PP** es un esquema sencillo donde la única información intercambiada entre maestro y esclavo es la posición, y como el nombre indica las fuerzas son estimadas a partir de errores de posición. Entre las ventajas de utilizar este esquema de control están su simplicidad, la ausencia de sensores de fuerza y respuesta rápida en términos de tiempo de cálculo. Entre sus desventajas están la inexactitud y una sensación poco realista para el operador.

La posición del robot maestro $\boldsymbol{\chi}_{\mathrm{m}}$ es usada como trayectoria de referencia para el esclavo, y la posición del esclavo $\boldsymbol{\chi}_{\mathrm{s}}$ es usada para generar fuerzas que son reflejadas al operador. Este esquema usa controladores PD tanto en el maestro como en el esclavo que actúan como una especie de resorte con constante $\boldsymbol{k}$ y un amortiguador con constante $\boldsymbol{b}$.

El **control FP** es un esquema de control más intuitivo, consecuencia de que las fuerzas generadas son mediciones de fuerzas reales. Del lado del esclavo, el esquema corresponde con el anterior en el sentido de que el control PD_s trata de seguir la trayectoria dada por el maestro. La diferencia está en que las fuerzas reflejadas de esclavo a maestro son fuerzas reales medidas del contacto entre el esclavo y una superficie, y las fuerzas generadas por el robot maestro únicamente son escaladas por una constante $\boldsymbol{p}$. Es claro entonces que este esquema requiere instalar algún sensor de fuerza en el efector final del robot esclavo.

El **control 4ch** intercambia fuerzas y posiciones tanto del lado del maestro como del esclavo. El nombre denota los cuatro canales de comunicación intercambiando valores de fuerza y posición en ambas direcciones. Requiere sensores de fuerza y posición en ambos lados del sistema de teleoperación.

5.2. Implementación de un sistema de teleoperación mediante un control PP

Se implementó un sistema de teleoperación conformado por dos dispositivos hápticos Geomagic Touch. El esquema de control elegido es el de *Posición Posición*, ya que ninguno de los dos dispositivos cuenta con un sensor de fuerza instalado.

En la Figura 5.2 se muestra de manera intuitiva la estructura básica del sistema de teleoperación. Tal como se ve en la figura el resultado de utilizar dos controladores PD tanto para el maestro como para el esclavo da la sensación de tener un resorte amortiguado entre ambos efectores finales. En la práctica permite percibir la dinámica del robot esclavo, tal como su inercia en movimientos rápidos o cuando se encuentra dentro de una restricción.

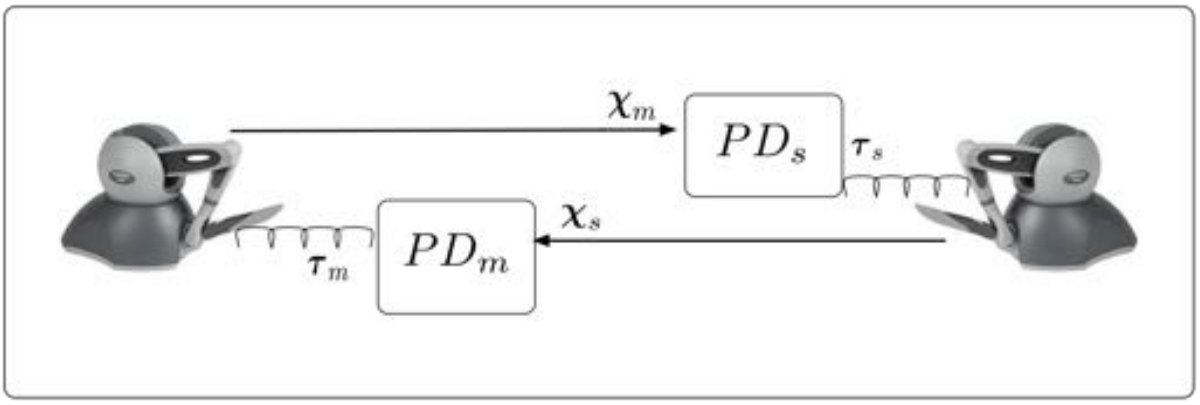


Figura 5.2: Diagrama del sistema de teleoperación

Del lado del robot esclavo, la estructura de programación es muy similar a la del seguimiento de trayectorias en el Capítulo 3. En cuanto al robot maestro, en primera instancia pareciera que también trata de seguir la trayectoria del robot esclavo, puesto que se trata de un *resorte virtual*. En realidad el objetivo no es ese, sino generar fuerzas de reacción cuando exista algún error de posición. En otras palabras, cuando el robot esclavo se encuentre con una restricción, *e.g.* una pared, el robot maestro va a estimar la fuerza de reacción a partir de los errores de posición, similar al método del *proxy* virtual presentado en el Capítulo 2.

5.2.1. Resultados

Los experimentos realizados consisten en que el operador genere una trayectoria y el robot esclavo la siga. La primera trayectoria es una serie de cuadrados en el espacio, tal y como se puede observar en la Figura 5.3. Los resultados del seguimiento se observan en la Figura 5.5. La segunda trayectoria es una serie de curvas generadas en el plano YZ, como se ve en la Figura 5.4; los resultados del seguimiento se encuentran en la Figura 5.6. Para la sintonización de los controladores, se utilizaron $k_{\mathrm{p}} = 0.15$ y $k_{\mathrm{d}} = 0.005$ en ambos casos.

El mejor resultado se obtuvo en la primer trayectoria, donde se observa un seguimiento excelente en las coordenadas X y Y. En la coordenada Z, sin embargo, los resultados no son tan buenos. De igual forma, para el segundo experimento los peores resultados de seguimiento se obtuvieron en la coordenada Z. Al comparar las curvas generadas, es posible notar que las curvas en la coordenada Z no son tan *suaves* como las demás. Esas pequeñas oscilaciones en el seguimiento seguramente son generadas debido a los efectos de la gravedad, que no son compensados por el controlador.

En general, el desempeño del sistema de teleoperación es bueno: el error es mínimo en la mayoría de los casos y no se observan retardos en el seguimiento, gracias en parte a la poca complejidad del sistema de control. El utilizar dos dispositivos iguales ayuda en la disminución de los cálculos ya que no se requieren realizar transformaciones para mapear el movimiento del robot maestro al esclavo.

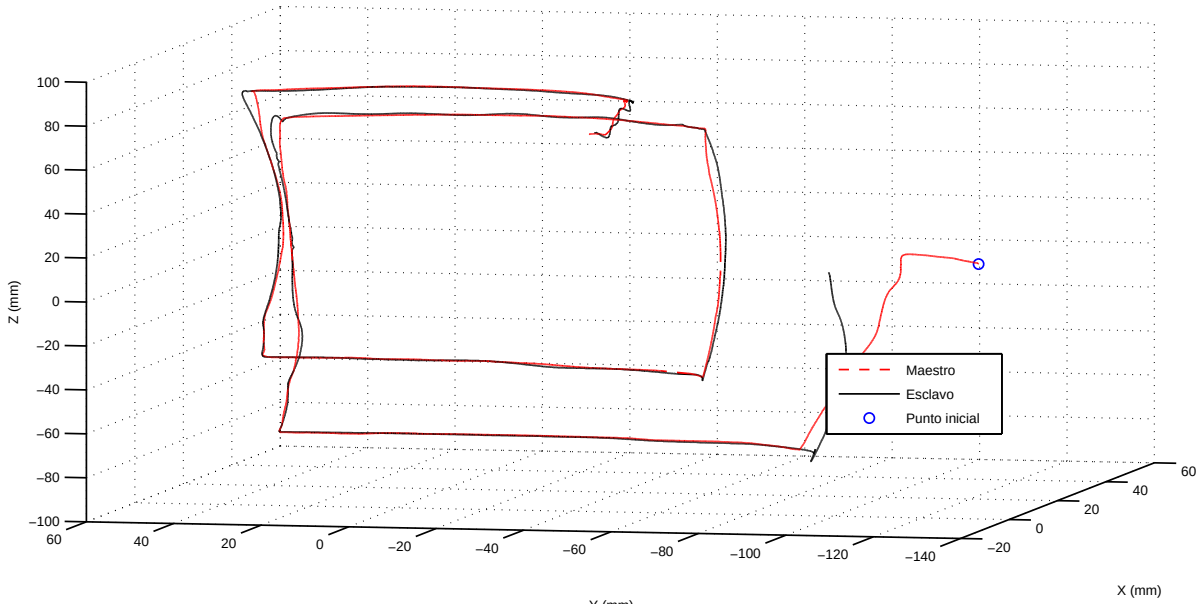


Figura 5.3: Dibujo de la trayectoria 1 del sistema maestro y esclavo

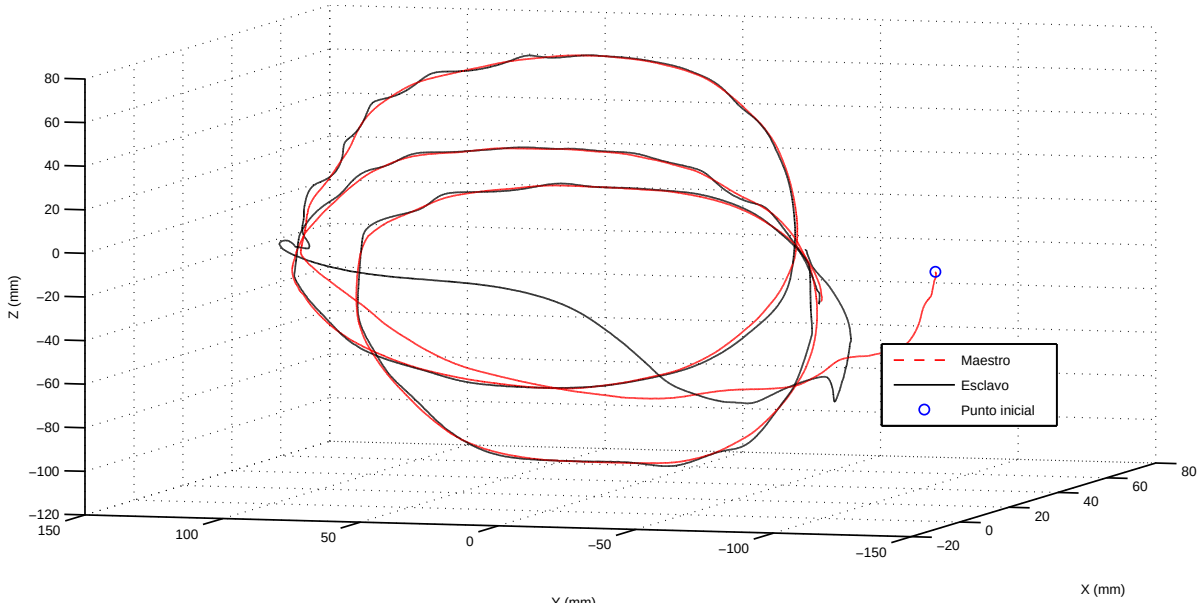
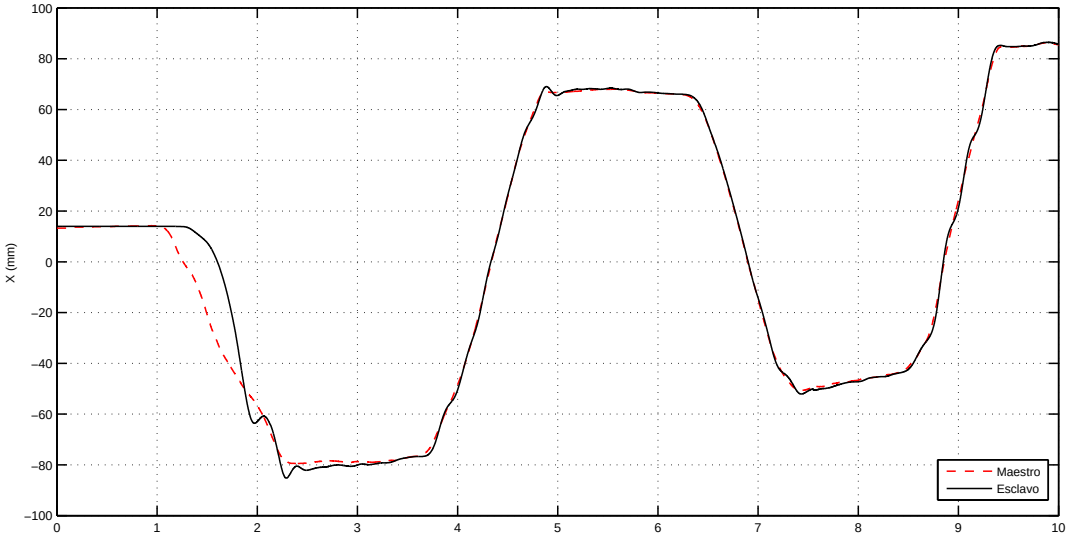
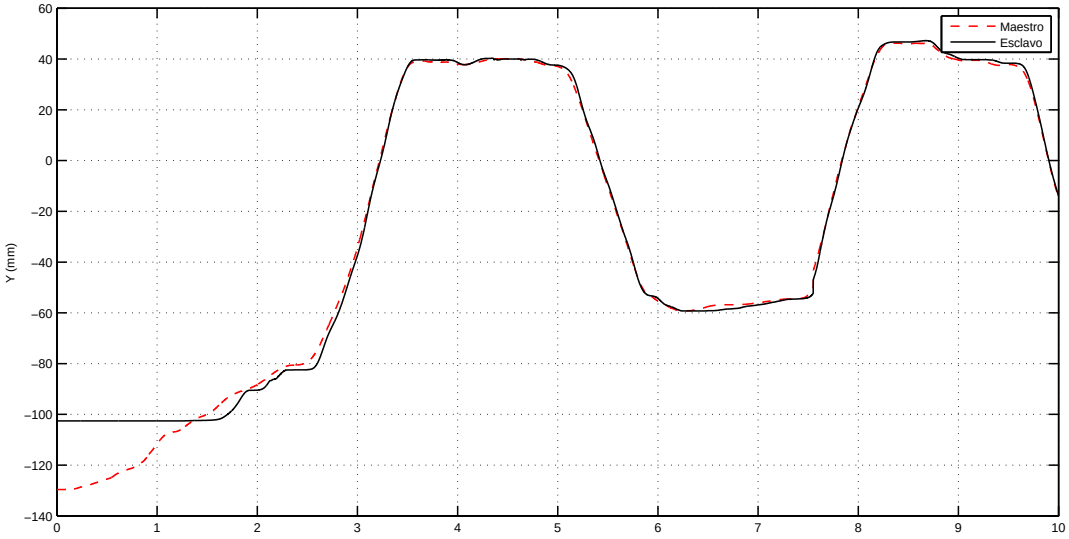


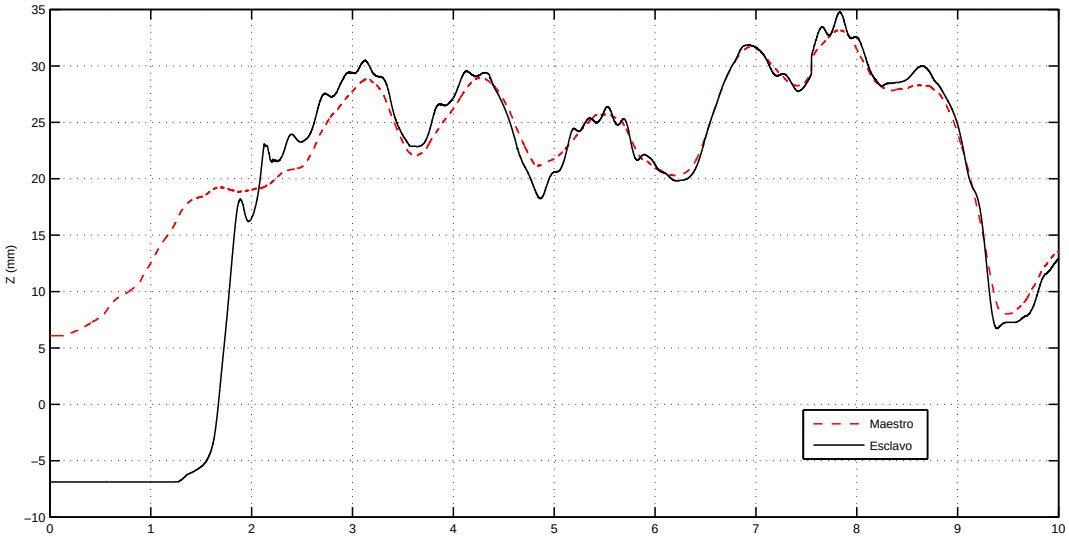
Figura 5.4: Dibujo de la trayectoria 2 del sistema maestro y esclavo



(a) Seguimiento en X

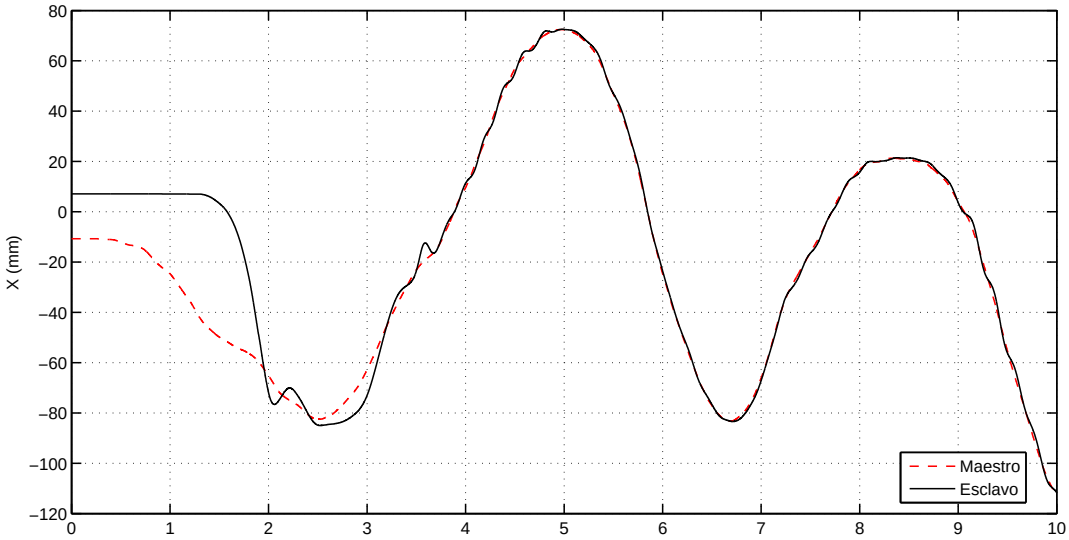


(b) Seguimiento en Y

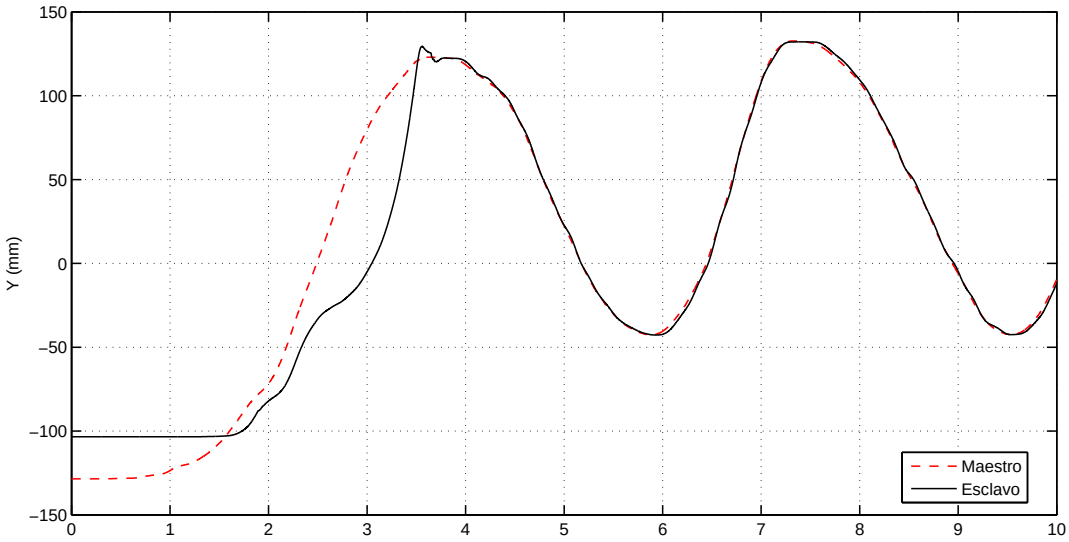


(c) Seguimiento en Z

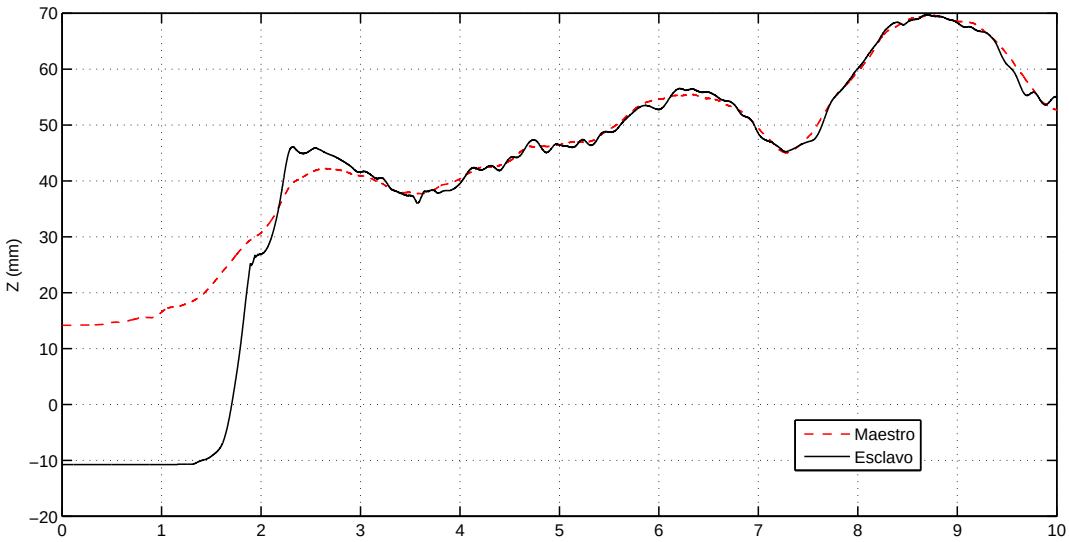
Figura 5.5: Seguimiento de trayectoria 1 en sistema de teleoperación



(a) Seguimiento en X



(b) Seguimiento en Y



(c) Seguimiento en Z

Figura 5.6: Seguimiento de trayectoria 2 en sistema de teleoperación

5.3. Canal de Comunicación

Uno de los problemas presentes en todo sistema de teleoperación es el de la comunicación. El objetivo de un sistema de teleoperación es ejecutar tareas en un lugar distinto al que nos encontramos, por lo que es necesario que la información viaje de un sitio a otro. Dependiendo de la distancia es el tiempo que tarda una señal en trasladarse. Al retardo temporal dentro de una red se le conoce como *latencia*, y es producida por la demora en la propagación y transmisión de paquetes de información. Otros factores que influyen en la latencia de una red son:

- El tamaño de los paquetes transmitidos
- El tamaño de los *buffers* dentro de los equipos de conectividad

En redes celulares, por ejemplo, la latencia es de 90 ms aproximadamente. Esto podría parecer poco, sin embargo, si consideramos una frecuencia de operación de una interfaz háptica de 1000 Hz, la información tarda al menos 180 ms (1/5 parte de la frecuencia) en completar un ciclo al viajar del sistema maestro al esclavo y de regreso. Si además se añade el tiempo en que se tarda uno de los sistemas en procesar la información, se pueden tener retardos de 0.5 s en la acción, lo cual ya es notable y deja de ser *natural* para el operador.

Otros puntos a considerar son la seguridad e integridad en la comunicación, así como la facilidad de implementación. A continuación se propone un canal de comunicación flexible que permita la conexión de las diferentes partes de un sistema de teleoperación.

5.3.1. Named Pipes

Las *named pipes* o *tuberías nombradas* son una vía de comunicación entre un servidor y uno o más clientes. Pueden establecer comunicación bidireccional y tienen un comportamiento *FIFO* (*First In First Out*). Todas las instancias de una tubería nombrada comparten el mismo nombre de tubería, pero cada instancia tiene sus propios *buffers* y puede proveer de conductos separados para la comunicación cliente/servidor. El uso de instancias habilita a múltiples clientes para utilizar el mismo nombre de tubería simultaneamente. Esto es particularmente útil cuando se requiere trabajar con varios canales de comunicación, como en el caso del control 4ch (ver Figura 5.7). También permite conectar sistemas *multirobot*, *e.g.* un sistema con dos robots maestros con dos robots esclavos de manera simultánea.

Las tuberías nombradas están diseñadas para conectar cualquier tipo de procesos, incluso si no están relacionados. Cualquier proceso puede actuar como cliente y servidor a la vez, haciendo posible la comunicación *P2P* (*Peer to Peer*), cuya característica principal es que permite el intercambio directo de información entre los ordenadores interconectados. Además, las tuberías nombradas pueden proveer un canal de comunicación entre procesos en la misma computadora o entre computadoras a lo largo de una red.

La función principal es *CreateNamedPipe()*:

```
1  Pipe = CreateNamedPipe(
2             lpszPipename,              // Nombre de la tuberia
3             PIPE_ACCESS_DUPLEX,        // Acceso de lectura/escritura
4             PIPE_TYPE_MESSAGE |        // Tipo de mensaje
5             PIPE_READMODE_MESSAGE |    // Modo de lectura de mensaje
6             PIPE_WAIT,                 // Modo de bloqueo
7             PIPE_UNLIMITED_INSTANCES,  // Instancias maximas
8             BUFSIZE,                   // Buffer de salida
9             BUFSIZE,                   // Buffer de entrada
10            0,                         // Timeout
11            NULL);                     // Atributo de seguridad
```

La estructura general de un canal de comunicación es:

```
Definir el tamano del buffer
Inicializar nombre de la tuberia tipo \\\\.\\pipe\\mytuberia
//Inicializacion del loop principal
WHILE(true)
  CreateNamedPipe(...)
  IF hay un error en la creacion THEN
    Salir
  ENDIF

  Esperar la conexion de un cliente (ConenctNamedPipe(...))
  IF cliente conectado THEN
    Crear un hilo de ejecucion
  ENDIF
ENDWHILE

//Ademas se declara la funcion que se ejecuta cada vez
//que se crea un hilo de ejecucion

InstanceThread(...)
  WHILE(este leyendo datos)
    Leer la solicitud del cliente (ReadFile(...))
    Procesar el mensaje entrante (GetAnswerToRequest(...))
    Escribir una respuesta (WriteFile(...))
  ENDWHILE
  Limpia la tuberia
END InstanceThread
```

Para que la comunicación sea efectiva y no interfiera con otros procesos, *i.e.* cálculos numéricos o renderizado gráfico o háptico, es necesaria la programación de hilos de programación independientes para cada canal de comunicación.

5.3.2. Named Pipes vs. Sockets TCP/IP

En un entorno con una red de área local (LAN) rápida, los Sockets TCP/IP y las tuberías nombradas (named pipes) son semejantes en términos de rendimiento. Sin embargo, las diferencias en rendimiento se vuelven notables con redes más lentas, tal como redes WAN (*Wide Area Networks*) o redes *dial-up*, debido a las diferencias entre los mecanismos de intercomunicación de procesos entre partes.

En general, se prefieren los sockets en redes lentas, ya sea LAN, WAN o dial-up, mientras que las tuberías nombradas pueden ser mejor opción donde la velocidad de la red no es un problema, ya que ofrece más funcionalidades, facilidad de uso y opciones de configuración [24].

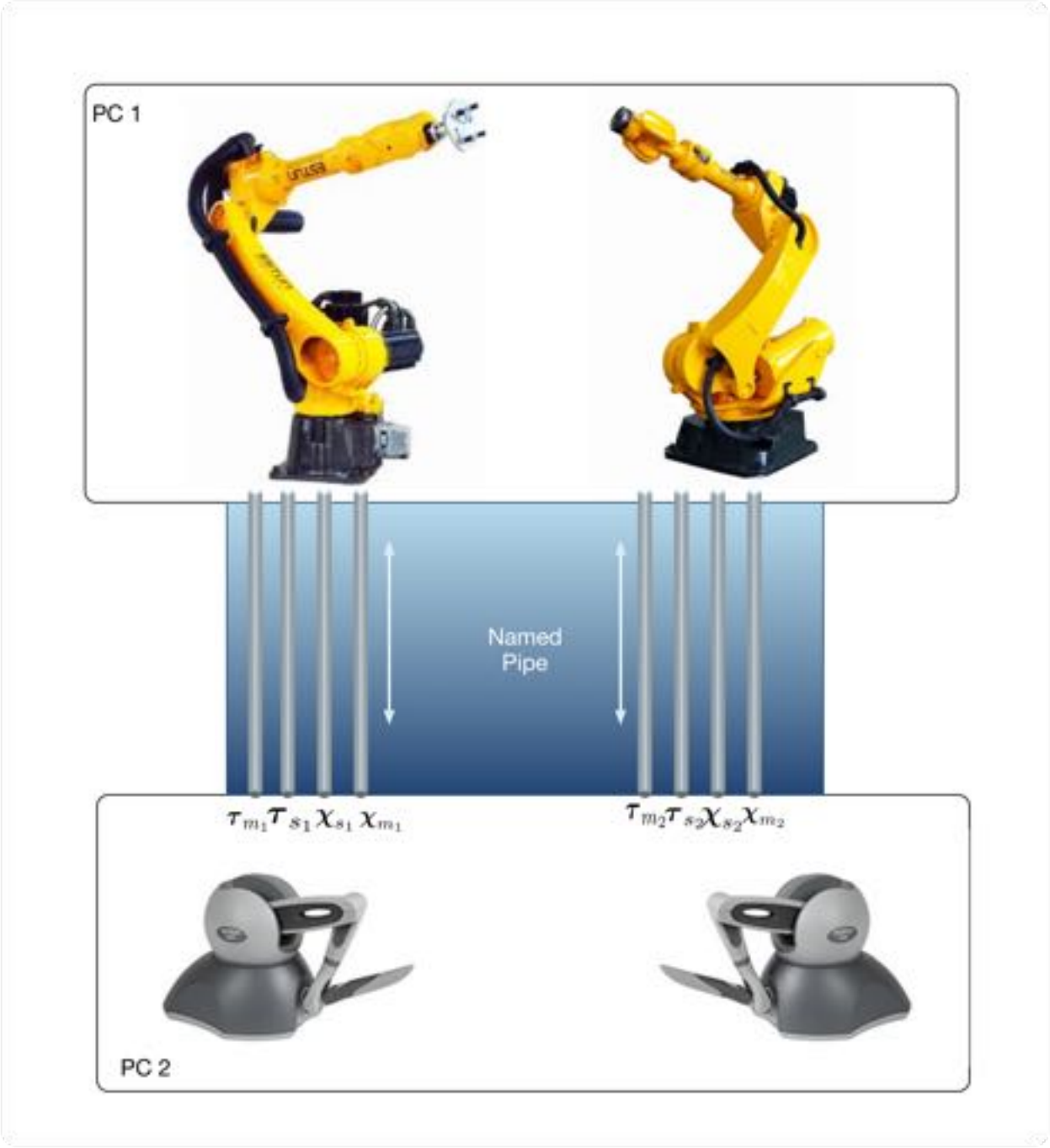


Figura 5.7: Uso de *named pipes* en un control 4ch

Capítulo 6

Conclusiones y trabajo futuro

Este trabajo abordó el problema de la interacción de un humano con ambientes virtuales a través de interfaces hápticas. La tesis se puede dividir en dos:

- Manipulación de ambientes virtuales
- Manipulación de ambientes remotos a través de sistemas de teleoperación

En la **interacción con un ambiente virtual**, se desarrolló un modelo matemático que simula el comportamiento dinámico de un objeto que se desplaza a lo largo de rieles; puesto que solo se puede desplazar en una dirección, se dice que el movimiento del objeto está *restringido*. La restricción que se modeló es del tipo *holonómica* y se representa mediante ecuaciones algebraicas de las variables que describen el movimiento del objeto.

Del lado de la interfaz háptica, se puso en marcha el dispositivo háptico Geomagic Touch, un robot de 6 grados de libertad con 3 de ellos actuados. Este pequeño robot está pensado para interfaces hápticas y sistemas de teleoperación por la dinámica casi *invisible* de sus elementos: no presentan mucha fricción, los eslabones son ligeros y pueden generar fuerzas que simulan distintas sensaciones. Se utilizaron las librerías de *OpenHaptics*, en especial HDAPI y HLAPI para el control del robot y el display gráfico, respectivamente, y las librerías de *OpenGL* para dibujar el ambiente virtual. Para probar el robot, se desarrolló una estructura de programación que permite probar varias leyes de control y visualizarlas de forma gráfica en tiempo real. Para ejemplificar esto, se hicieron pruebas utilizando controladores lineales (P, PI, PD y PID), con resultados razonables excepto cuando el robot tiene que contrarrestar el efecto de la gravedad en los eslabones, es decir cuando efectúa movimiento en el eje coordenado Z.

Una vez obtenido el modelo dinámico del sistema virtual y la estructura de programación para utilizar el dispositivo háptico, se desarrolló un simulador donde ambos interactúan. Para establecer la interacción entre ambos sistemas se propuso un algoritmo de colisión basado en el método de *ray tracing*, un método posicional ampliamente usado en motores gráficos. Para resolver la ecuación diferencial del modelo se utilizó el método numérico de Runge-Kutta de 4° orden.

Los resultados obtenidos de la interacción del ambiente virtual con el dispositivo háptico fueron satisfactorios. Por un lado el operador sintió la interacción de forma natural, percibiendo las diferentes fuerzas que actúan sobre el sistema como fricción estática, la masa del objeto y las fuerzas de reacción debido a la restricción. Desde el punto de vista numérico, los resultados no se pueden comparar con otras simulaciones ya que el sistema desarrollado es en sí misma una simulación. Empero, las gráficas de fuerza, posición, velocidad y fuerza de reacción debido a la restricción concuerdan con las esperadas.

A pesar de que en [3] se establece una frecuencia mínima de operación de 1000 Hz para *sentir* de forma natural al ambiente virtual, la experiencia al trabajar con una frecuencia de 860 Hz fue muy buena. Empero, cabe resaltar algunos detalles: la presencia de

discontinuidades, *e.g.* en el límite entre dos paredes del entorno virtual, a veces ocasiona que el efector final atraviese la superficie regresando de manera brusca. Por otro lado, ya que el método de renderizado de fuerzas depende de la penetración del efector final en el objeto virtual para simular la fuerza de contacto, cuando se aplica mucha fuerza sobre el objeto virtual se puede ver al efector final penetrar en el objeto y no mantenerse sobre la superficie como habría de esperarse.

En el caso del **sistema de teleoperación**, se utilizó el esquema de control *Posición Posición*. Se implementó un control PD tanto en el robot maestro como en el esclavo para controlar su movimiento. Los resultados fueron muy buenos para el movimiento en las coordenadas X y Y, sin embargo, en la coordenada Z se observa un movimiento menos suave que en las otras coordenadas, debido al efecto de la gravedad sobre los elementos y a que el esquema de control usado no toma en cuenta la dinámica del robot ni efectos de la gravedad.

Finalmente se presentó un esquema de comunicación para sistemas de teleoperación basado en *tuberías nombradas* o *named pipes*. Este esquema permite, entre otras cosas, establecer una comunicación direca (*P2P*) entre dos procesos mediante múltiples canales de comunicación. Adicionalmente, es flexible en el sentido de que un mismo proceso puede actuar como cliente o servidor en la comunicación y permite la comunicación de múltiples clientes con un mismo servidor. Esto es particularmente útil si se desea implementar un sistema de teleoperación utilizando esquemas que requieran varios canales, como el *FP* o el *4ch* mencionados en el Capítulo 5.

Hay muchas ventanas de oportunidad en el desarrollo de interfaces hápticas. En general, se pueden mejorar los siguientes aspectos:

- Modelado
- Esquemas de control y renderizado háptico
- Algoritmos de colisión
- Sistemas de teleoperación usando esquemas de control *FP* o *4ch*

En cuanto al modelado, se puede incrementar la complejidad del modelo dinámico del ambiente virtual al considerar un mayor número de restricciones holonómicas. Adicionalmente, se pueden considerar restricciones *no holonómicas* y aumentar el número de objetos y la complejidad de estos. Hecho esto, será necesario diseñar algoritmos de detección de colisiones más complejos y más eficientes, así como métodos numéricos con menores errores de cálculo y mayor eficiencia. Otro punto interesante es el modelado de objetos deformables, como tejidos u objetos que puedan romperse.

El diseño del controlador es uno de los puntos más importantes donde se pueden implementar mejoras. Tratándose de sistemas no lineales, el siguiente paso es implementar controladores no lineales tanto para el control del robot como para el renderizado háptico. Con un buen esquema de control, es posible renderizar sensaciones más complejas que la de una simple colisión, *e.g.* simular texturas u objetos deformables. Los esquemas de control posteriores deben incorporar la dinámica del robot y las dinámicas del entorno, y no tratarlas como si fueran perturbaciones constantes.

Trabajos posteriores sobre sistemas de teleoperación deberán aumentar la complejidad de los esquemas de control. En la ausencia de sensores de fuerza, se pueden incorporar observadores de fuerza y utilizar controladores más complejos que el PD para la estimación de la fuerza de reacción. Sin embargo, lo recomendable es usar esquemas que utilicen mediciones reales de la fuerza de contacto del esclavo con el entorno; los esquemas de control *FP* y *4ch* se caracterizan por incluir alguna medición de fuerza del lado del robot esclavo.

Por último, en trabajos futuros se puede explorar la generación de interfaces hápticas con sistemas distintos a los electromecánicos. Un ejemplo de esto es el reciente desarrollo de Disney Research sobre la generación de interfaces hápticas utilizando torbellinos de aire [16]. Esto abre las puertas a un mundo poco explorado en la interacción hombre-máquina.

Apéndice A

Solución de ODEs

Ingenieros y científicos han buscado la ayuda de *máquinas de cálculo* de cualquier tipo para solucionar ecuaciones matemáticas que modelen sistemas dinámicos. La máquina de cálculo más usada hoy en día es la computadora. De manera general, la eficacia de los resultados obtenidos con una computadora dependen de dos condiciones: la correcta formulación del problema y la capacidad de cómputo (*i.e.* de cálculos) del equipo.

El modelo matemático de un sistema es expresado en términos de ecuaciones diferenciales, y en nuestro caso, de ecuaciones diferenciales ordinarias (ODEs). La solución numérica de estas ecuaciones es implementada mediante la integración de las primeras derivadas; cuando se tienen derivadas de orden superior, una práctica común es realizar cambios de variables y transformar un sistema de orden n en un conjunto de n sistemas de primer orden.

A.1. Método de Euler

En 1768, Leonhard Euler, uno de los matemáticos más prominentes de la historia, publicó el primer método numérico para la resolución de ecuaciones diferenciales de primer orden de la forma:

$$\frac{dx}{dt} = f(x,t), \qquad x(t_0) = x_0. \tag{A.1}$$

El método propuesto por Euler [25] está basado en una aproximación de diferencias finitas de la definición de derivada, formulada por otro gran matemático contemporáneo de Euler, Brook Taylor. De esta forma, Euler aproximó la fórmula de Taylor

$$\frac{dx}{dt} = \lim_{\Delta t \to 0} \frac{[x(t_0 + \Delta t) - x(t_0)]}{\Delta t} \tag{A.2}$$

con la expresión

$$\frac{dx}{dt} \approx \frac{[x(t_0 + \Delta t) - x(t_0)]}{\Delta t}. \tag{A.3}$$

Es decir, su aproximación es mediante la eliminación del límite que hace Δt tender a cero. A partir de esta aproximación, el valor estimado de la función x en el tiempo $t_0 + \Delta t$ es

$$x(t_0 + \Delta t) \approx x(t_0) + \left.\frac{dx}{dt}\right|_{t_0} \Delta t. \tag{A.4}$$

Sustituyendo dx/dt de la Ec. (A.1) se obtiene la solución aproximada

$$x(t_0 + \Delta t) \approx x(t_0) + f(x_0, t_0)\Delta t. \tag{A.5}$$

En la Figura A.1 se muestra una representación gráfica del método de Euler. Este método asume que la pendiente de $x(t)$ permanece constante con su valor en $t = t_0$ para todo

el intervalo de integración. Esto es equivalente a utilizar el primer término de la Serie de Taylor, por lo que a este método se le considera de primer orden.

La aproximación (A.5) provee de un algoritmo recursivo para el cálculo de $x(t_0 + k\Delta t)$, con $k = 1, 2, ..., N$, es decir, de la solución de la ODE. El procedimiento consiste en hacer el cálculo, desde el tiempo inicial $t = t_0$ hasta el tiempo final $t_f = t_0 + N\Delta t$ con un paso constante para Δt. Las ecuaciones a resolver son:

$$\begin{aligned} x(t_0 + \Delta t) &= x_0 + f(x, t_0)\Delta t, \\ x(t_0 + 2\Delta t) &= x(t_0 + \Delta t) + f[x, (t_0 + \Delta t)]\Delta t, \\ &\vdots \\ x(t_0 + N\Delta t) &= x[t_0 + (N-1)\Delta t] + f[x, [t_0 + (N-1)\Delta t]]\Delta t \end{aligned} \tag{A.6}$$

En general, para el caso de un sistema de orden n representado por un conjunto de n ODE de primer orden, la ecuación (A.1) puede ser rescrita como sigue:

$$\frac{d\boldsymbol{q_i}}{dt} = f_i(\boldsymbol{q}, t), \quad i = 1, 2, ..., n \tag{A.7}$$

Donde $\boldsymbol{q_i}$ es un vector columna.

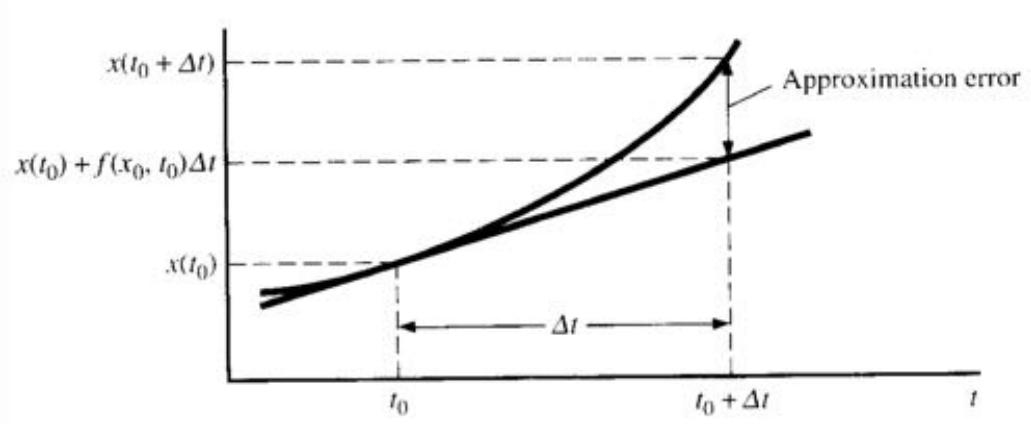


Figura A.1: Método de Euler [25]

A.2. Método de Runge-Kutta

En el método de Euler, una derivada contínua se aproxima a partir del término de primer orden de la serie de Taylor de la función. Como se observa en la Figura A.1, existe un error de aproximación que en algunos casos puede ser muy grande. La exactitud de la aproximación de la primera derivada y, eventualmente, de la estimación de $x(t_0 + \Delta t)$ pueden ser mejorados con la inclusión de términos de orden superior de la serie de Taylor, y es la idea esencial detrás de una gran cantidad de métodos de integración numérica. Por otro lado, el *truco* del método de Runge-Kutta, consiste en aproximar las derivadas de orden superior por expresiones de diferencia finita y así no tener que calcularlos de la ecuación diferencial original [25]. Las aproximaciones son calculadas a partir de datos obtenidos de pasos tentativos realizados desde t_0 hasta $t_0 + \Delta t$. El número de pasos usados en la estimación de $x(t_0 + \Delta t)$ determinan el orden del método de Runge-Kutta [25].

En la versión más común de este método, se realizan 4 pasos tentativos dentro de cada intervalo de tiempo, y los valores sucesivos de la variable dependiente son calculados de la siguiente manera:

$$x(t_0 + \Delta t) = x(t_0) + \left(\frac{\Delta t}{6}\right)(k_1 + 2k_2 + 2k_3 + k_4) \tag{A.8}$$

donde

$$\begin{aligned} k_1 &= f[x(t_0), t_0], \\ k_2 &= f\left[\left(x(t_0) + \Delta t \frac{k_1}{2}\right), \left(t_0 + \frac{\Delta t}{2}\right)\right], \\ k_3 &= f\left[\left(x(t_0) + \Delta t \frac{k_2}{2}\right), \left(t_0 + \frac{\Delta t}{2}\right)\right], \\ k_4 &= f[x(t_0) + \Delta t k_3), (t_0 + \Delta t)] \end{aligned} \tag{A.9}$$

Para sistemas de orden superior, un conjunto de n ecuaciones se resuelve, una en cada paso de tiempo de la solución numérica.

Apéndice B

Cinemática inversa mediante el Jacobiano geométrico

B.1. Jacobiano geométrico

Considere un robot manipulador de n-grados de libertad. La ecuación de la cinemática directa se puede escribir mediante una matriz de transformación de la forma:

$$\boldsymbol{T}_e(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{R}_e(\boldsymbol{q}) & \boldsymbol{p}_e(\boldsymbol{q}) \\ \\ \boldsymbol{0}^T & 1 \end{bmatrix} \tag{B.1}$$

donde $\boldsymbol{q} = [q_1 ... q_n]$ es el vector de variables articulares, $\boldsymbol{R}_e(\boldsymbol{q}) \in \Re^{3\times 3}$ la matriz de rotación del robot y $\boldsymbol{p}_e(\boldsymbol{q})$ el vector de posición del efector final. La relación entre la velocidad lineal y angular del efector final con las velocidades articulares se define como [17]:

$$\dot{\boldsymbol{p}}_e = \boldsymbol{J}_P(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{B.2}$$

$$\boldsymbol{\omega}_e = \boldsymbol{J}_O(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{B.3}$$

donde $\boldsymbol{J}_P$ es la matriz de $(3\times n)$ que relaciona la contribución de las velocidades articulares $\dot{\boldsymbol{q}}$ con la velocidad lineal del efector final $\dot{\boldsymbol{p}}_e$ y $\boldsymbol{J}_O$ es la matriz de $(3 \times n)$ que relaciona la contribución de las velocidades articulares con la velocidad angular del efector final $\boldsymbol{\omega}_e$. En forma compacta:

$$\boldsymbol{v}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{B.4}$$

que representa la *ecuación cinemática diferencial* del robot manipulador. La matriz $\boldsymbol{J}$ de $(6 \times n)$ es el *Jacobiano geométrico* del manipulador

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_P \\ \boldsymbol{J}_O \end{bmatrix}. \tag{B.5}$$

En [17] se define el cálculo del Jacobiano geométrico de la siguiente forma:

$$\begin{bmatrix} \boldsymbol{J}_{Pi} \\ \boldsymbol{J}_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & \text{para articulaciones prismáticas} \\ \begin{bmatrix} z_{i-1} \times (\boldsymbol{p}_e - \boldsymbol{p}_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{para articulaciones de revolución.} \end{cases} \tag{B.6}$$

B.2. Cinemática diferencial inversa

La ecuación cinemática diferencial representa un mapeo lineal entre la velocidad en el espacio articular y el espacio operacional del robot. De la Ec. (B.4) se pueden obtener las velocidades articulares mediante la inversión del Jacobiano

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{\phi}^{-1}(\boldsymbol{q})\boldsymbol{v}_e. \tag{B.7}$$

Si se conoce la posición inicial $\boldsymbol{q}_0$ del robot, se pueden calcular las posiciones articulares integrando las velociades, es decir

$$\boldsymbol{q} = \int_0^t \dot{\boldsymbol{q}}(\varsigma)d\varsigma + \boldsymbol{q}(0). \tag{B.8}$$

La integración se puede realizar en forma discreta utilizando métodos numéricos. El método más simple consiste en el método de integración de Euler (ver Apéndice A); dado un intervalo de integración Δt, si las posiciones y velocidades articulares son conocidas en el tiempo t_k, las posiciones en tiempo t_{k+1} se calculan como

$$\boldsymbol{q}(t_{k+1}) = \boldsymbol{q}(t_k) + \dot{\boldsymbol{q}}(t_k)\Delta t. \tag{B.9}$$

Sustituyendo (B.7) en (B.9) se obtiene

$$\boldsymbol{q}(t_{k+1}) = \boldsymbol{q}(t_k) + \boldsymbol{J}_{\phi}^{-1}(\boldsymbol{q}(t_k))\boldsymbol{v}_e(t_k).\Delta t, \tag{B.10}$$

que permite mapear las posiciones articulares del robot manipulador a partir del Jacobiano geométrico y la velocidad lineal del efector final.

Bibliografía

[1] A. Whitehead. *Science and the Modern World*. The New American Library, 1925.

[2] J. Fox, D. Arena, and J. Bailenson. Virtual reality: A survival guide for the social scientist. *Journal of Media Psychology*, 2009.

[3] *OpenHaptics Toolkit Programmer's Guide*. Sensable, 3.1 edition.

[4] K. Garcia. Control de fuerza sobre superficies virtuales. Master's thesis, UNAM, 2014.

[5] A. Bertelsen, J. Melo, et al. A review of surgical robots for spinal interventions. *The International Journal of Medical Robots and Computer Assited Surgery*, 2012.

[6] A. Nygaard. High-level control system for remote controlled surgical robots. Master's thesis, Norwegian University of Science and Technology, 2008.

[7] D.A. Rosenbaum. *Human Motor Control*. Academic Press, 1990.

[8] M.V. Ottermo. *Virtual Palpation Gripper*. PhD thesis, Norwegian University of Science and Technology, 2006.

[9] S. Skoglund. Anatomical and physiological studies of knee joint in the cat. *Acta physiologica Scandinavica. Supplementum*, 1956.

[10] *Tactual Perception - A Source Book*. Cambridge University Press, 1982.

[11] NASA-CIRSSE International Conference on Intelligent Robotic Systems for Space Exploration, editor. *Finger force and touch feedback issues in dextrous telemanipulation*, 1992.

[12] *Force and Touch Feedback for Virtual Reality*. John Wiley and Sons, 1996.

[13] URL `http://www.davincisurgery.com/es/`.

[14] *Overview of current developments in haptics APIs*, 2011. The 15th Central European Seminar on Computer Graphics.

[15] H. Jin, L. Wang, et al. Design and control strategy of robotic spinal surgical system. In *IEEE/ICME International Conference on Complex Medical Engineering*, 2011.

[16] R. Sodhi, I. Poupyrev, et al. Aireal: Interactive tactile experiences in free air. Technical report, Disney Research, 2013.

[17] B. Siciliano, L. Sciavicco, et al. *Robotics. Modeliing, Planning and Control*. Springer, 2010.

[18] E. Bayo and J. Garcia de Jalon. A modified lagrangian formulation for the dynamic analysis of constrained mechanical systems. *Computer Methods in Applied Mechanics and Engineering*, 1988.

[19] E. Hendricks, O. Jannerup, and P.H. Sorensen. *Linear Systems Control: Deterministic and Stochastic Methods*. Springer, 2008.

[20] M. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley and Sons, 2006.

[21] R. Mukundan. *Advanced Methods in Computer Graphics*. Springer, 2012.

[22] J. Vince. *Mathematics for Computer Graphics*. Springer, 2006.

[23] D.A. Lawrence. Stability and transparency in bilateral teleoperation. *IEEE Transactions on Robotics and Automation*, 1993.

[24] URL `https://technet.microsoft.com/en-us/library/aa178138(v=sql.80).aspx`.

[25] B.T. Kulakowski, John F. Gardner, and J.L. Shearer. *Dynamic Modeling and Control of Engineering Systems*. Cambridge University Press, 2007.