CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE MECATRÓNICA

Teleoperación de un robot diferencial (2,0) con fuerza de retroalimentación de ganancia variable

Tesis que presenta:

Héctor Mendoza Hernández

Para obtener el grado de:

Maestro en Ciencias

En la especialidad de:

Ingeniería Eléctrica

Directores de la Tesis:

Dr. Rafael Castro Linares

Dr. Jaime Álvarez Gallegos

México, Distrito Federal. Febrero 2018

II

Agradecimentos

A mi familia: *Por el apoyo constante que me brindaron durante mis estudios de maestría.*

A mis amigos: *Porque siempre estuvieron conmigo para motivarme a conseguir mis metas personales, y que en ningún momento dudaron de mí.*

A mis compañeros de generación: *Por permitirme trabajar en conjunto con ellos, y de esa manera conseguir un mejor aprendizaje de muchos de los temas vistos.*

A mis asesores: *Porque siempre tuvieron tiempo de compartir su conocimiento conmigo y responder a mis dudas, pero sobretodo por su paciencia.*

A los miembros del jurado: *Por sus comentarios y las aportaciones que hicieron a este trabajo de tesis.*

A los doctores de la sección de mecatrónica: *Por sus horas de trabajo dedicadas a que los alumnos de mi generación pudieran desarrollarse como maestros.*

A los empleados de la sección de mecatrónica: *Porque siempre que solicité su ayuda lo hicieron de una manera amable y cortés.*

Al Conacyt: *Por el apoyo económico a través del proyecto CB-2015-01/254329.*

Al Auxiliar M. en C. Igor Morett Valenzuela: *Por el préstamo de equipo especializado en este proyecto.*

Resumen

Este trabajo se enfoca en la utilización de dos estrategias de control aplicadas a un robot móvil (2,0). La primera estrategia se trata de un control discontinuo por modos deslizantes. La segunda estrategia se trata de un esquema de teleoperación con fuerza de retroalimentación de ganancia variable. Dicho esquema de teleoperación nos permite sentir a través de un mando háptico una fuerza que varía dependiendo de la distancia entre el robot y algún obstáculo así como de la velocidad de aproximación. La estrategia de control discontinuo puede trabajar con o sin la ayuda del esquema de teleoperación.

El cálculo del control se lleva a cabo en una computadora de escritorio y las señales de control se envían de manera inalámbrica a través de un servidor de internet. Se induce un retardo constante en la comunicación para simular la existencia de una distancia considerable entre el robot y la computadora. El robot fue modelado en tiempo discreto, tomando en cuenta el retardo constante en las señales de control.

La estrategia de control utilizada le permite al robot seguir una trayectoria deseada evadiendo un obstáculo.

Abstract

This work focuses on the use of two control strategies applied to a mobile robot (2,0). The first strategy is a discontinuous control by sliding modes. The second strategy is a teleoperation scheme with variable gain feedback force. This teleoperation scheme allows us to feel through a haptic device a force that varies depending on the distance between the robot and some obstacle as well as the approaching speed. The discontinuous control strategy can work with or without the help of the teleoperation scheme.

The computation of the control strategy is carried out on a desktop computer and the control signals are sent wirelessly through an internet server. A constant delay in communication is induced to simulate the existence of a considerable distance between the robot and the computer. The robot was modeled in discrete time, taking into account the constant delay in the control signals.

The control strategy used allows the robot to follow a desired trajectory evading an obstacle.

Contenido

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

La constante búsqueda del ser humano de delegar labores peligrosas a los robots ha llevado a una extensa investigación del tema. Un tipo particular de robots son aquellos que se pueden desplazar libremente en un marco de referencia, o sea los robots móviles, los cuales se utilizan principalmente en la exploración de terrenos de difícil acceso o peligrosos para los humanos. El cálculo del control de tales robots puede llevarse a cabo en una computadora a bordo del robot o dicha computadora puede encontrarse a una distancia considerable del robot y comunicarse con él de manera inalámbrica. Entre los principales objetivos de los diversos esquemas de control se encuentran; el seguimiento de una trayectoria, alcanzar una posición deseada y/o la evasión de obstáculos. Cuando se envían las señales de control al robot desde una posición remota, es inherente la presecia de retardos, dichos retardos afectan de manera negativa al comportamiento del sistema y en algunos casos vuelven inestable a la planta. Por otra parte, existe también la posibilidad de controlar de manera manual a los robots móviles a través de mandos hápticos, los cuales nos permiten sentir fuerzas asociadas a la distancia entre el robot y algún obstáculo para evitar alguna colisión, es a lo que comúnmente se le conoce como teleoperación, donde se pueden ocupar diversos sensores para mejorar la calidad de la teleoperación, como cámaras por ejemplo. Sin embargo, la presencia de retardos, aún cuando se opera de manera manual el robot puede provocar colisiones, puesto que la presencia de un obstáculo podría verse o sentirse ya cuando es muy tarde, por lo tanto es necesario predecir de alguna manera la trayectoria del robot y las posibles colisiones.

1.1. Planteamiento del problema.

En este trabajo en particular se busca que un robot móvil diferencial (2,0) sea capaz de alcanzar una posición deseada, además de evitar un obstáculo mientras cumple la tarea principal. Para lograr esto se utilizará una combinación entre el control automático y la teleoperación. Donde se consideran retardos constantes inducidos artificialmente en la comunicación, el hecho de que la magnitud del retardo sea constante e invariante en el tiempo es para facilitar su tratamiento en el esquema de control. A continuación se muestran algunos de los problemas a resolver:

- Seguimiento de una trayectoria deseada.
- Compensación del retardo de comunicación.
- Evasión de obstáculos.
- Implementación de una estrategia de control discontinuo.
- Implementación de un esquema de teleoperación de fuerza de retroalimentación de ganancia variable.
- Trabajo cooperativo entre el control automático y la teleoperación.

1.2. Objetivo de la tesis.

Objetivos generales:

- Llevar a un robot diferencial de una posición inicial a una posición final deseada.
  - Evadiendo obstáculos.
  - Incluyendo retardos en las señales de control.
- Lograr el seguimiento de una trayectoria deseada.
  - Evadiendo obstáculos.
  - Incluyendo retardos en las señales de control.

Objetivos particulares:

- Diseñar un controlador discontinuo en tiempo discreto para un robot (2,0).
  - Que permita al robot alcanzar una posición cartesiana.
  - Que permita al robot seguir una trayectoria.
  - Que permita al robot evadir obstáculos
  - Incluyendo retardos en las señales de control.
- Implementar el esquema de teleoperación de fuerza de retroalimentación de ganancia variable.
- Combinar el control automático con la teleoperación.
- Utilizar la visión artificial a través una cámara a bordo del robot.

1.3. Antecedentes

En esta sección se citaran trabajos relacionados con la teleoperación de robots móviles, el modelado en tiempo discreto, estrategias de control discontinuo y algunos otros temas que fueron relevantes para desarrollar el trabajo presente.

En [1] se describen las operaciones de un brazo robótico de cinco grados de libertad localizado en la superficie del planeta Marte. Una de las ventajas de que el control sea a distancia es que existe la posibilidad de intervenir el funcionamiento del robot cuando se presente un imprevisto, pero esta ventaja viene con el inherente problema del retardo de comunicación entre la computadora que calcula el control y el robot. Algunos autores han propuesto distintos métodos para aminorar el efecto de este retardo. Uno de los caminos seguidos es tratar el sistema en tiempo discreto.

Una de las principales tareas de los robots móviles es la de poder seguir una trayectoria, con el fin de explorar alguna zona. Este trabajo se enfoca en el seguimiento de una trayectoria cuando existe retardo entre el envío y la recepción de las señales de control. Este tema ha sido tratado para diferentes robots con distintos esquemas de control. En [2] se utiliza el modelo en tiempo discreto de un robot móvil (2,0) tomando en cuenta el retardo en las señales de control, y junto con un control discontinuo para compensar el efecto del retardo en el funcionamiento del sistema. En [3] se obtiene el modelo discreto exacto de un robot móvil (2,0), linealizándolo para después usar controles auxiliares del tipo proporcional, con los que se logra el seguimiento de una trayectoria. En [4] se obtiene el modelo discreto extendido para un robot omnidireccional. Ahí se utiliza dicho modelo para sintetizar leyes de control con el fin de realizar el seguimiento de una trayectoria. Por otra parte, algunas veces se necesita que los robots naveguen libremente a fin de alcanzar una meta. Esto significa que la trayectoria no puede ser planificada. En la mayoría de las ocasiones el robot se topa con obstáculos que debe evadir. Es necesario tener leyes de control robustas para poder trabajar con los problemas del retardo, los obstáculos y las posibles singularidades del sistema. En [5] se utiliza el modelo en tiempo discreto extendido de un robot móvil (2,0), junto con el método de potenciales artificiales para lograr la evasión de obstáculos y regulación de posición, usando un esquema de control discontinuo discreto. En [6] se propone un control discreto robusto, de tipo discontinuo para controlar un robot manipulador de unión flexible en simulación. Este último esquema produce un error de seguimiento pequeño en un número pequeño de periodos de muestreo.

Por otro lado, cuando el operador se encuentra lejos necesita la ayuda de distintos dispositivos a fin de poder tener una mejor perspectiva del terreno en el que se encuentra el robot. Esto se logra con cámaras, sensores de proximidad y mandos hápticos. Este tema ha sido abordado ampliamente por distintos autores. En [7] se estudia el comportamiento de un robot móvil (2,0) con diferentes estrategias de teleoperación;

posición-posición, posición-velocidad e híbrido. En [8] se propone un esquema de teleoperación asistido, utilizando un modelo que predice el comportamiento del robot para evitar colisiones. En [9] se presenta un esquema de teleoperación háptica con fuerza de retroalimentación de ganancia variable. Ahí se comprueba la eficacia de este método en comparación al anterior que se basa en una ganancia constante.

1.4. Contribución de la tesis.

Este trabajo tiene entre sus principales aportaciones la implementación de una estrategia de control discontinuo en tiempo discreto que nunca se había utilizado en los robots móviles. Otra diferencia con respecto a trabajos anteriores es la aplicación de la estrategia de teleoperación de fuerza de retroalimentación de ganancia variable que no se había ocupado en sistemas con retardo constante en la comunicación. En la parte tecnológica se utilizó un teléfono celular para adquirir imagen en tiempo real y de esa manera facilitar la navegación del robot cuando se encontraba en el modo de teleoperación. Todo el sistema consta de dos computadoras, un teléfono celular, un robot móvil (2,0), un mando háptico, el código de control y un operador humano. Con todas estas partes funcionando en conjunto fue posible lograr que el robot alcanzara una posición deseada compensando el retardo de la comunicación, además de evadir un obstáculo.

1.5. Organización de la tesis.

En el capítulo 2 se presentan las bases para comprender el comportamiento de un robot móvil (2,0), empezando por el modelo cinemático en tiempo continuo con presencia de retardo en las señales de control, despues se muestra una forma de controlar el robot automáticamente y finalmente se muestra un esquema particular de teleoperación.

En el capítulo 3 se describe la síntesis de la estrategia de control discontinuo usado, partiendo de un modelo en tiempo discreto del robot móvil (2,0) con retardos en la entrada. También se muestran las dos formas utilizadas en este trabajo para generar las trayectorias deseadas; usando un robot virtual o usando el método de potenciales artificiales.

En el capítulo 4 se describe a profundidad la estrategia de teleoperación con fuerza de retroalimentación de ganancia variable. La cual depende de la distancia que existe entre el robot y el obstáculo así como de la velocidad de aproximamiento. Se describe

también la manera en la que interactuan la teleoperación y el control automático.

En el capítulo 5 se dan detalles técnicos sobre la plataforma experimental. Mostrando ambas configuraciones, cuando únicamente se utiliza el control automático y cuando se ocupa la combinación del control automático con la teleoperación. Posteriormente se describe cada elemento que compone la plataforma experimental, como lo son: el robot, el mando háptico, la cámara, etc.

En el capítulo 6 se exponen los resultados de los 5 experimentos distintos que se llevaron a cabo. Cada uno con sus propias características, ya sea por el esquema de control o por las limitaciones técnicas. Siendo los relacionados con el seguimiento de una trayectoria punto a punto los más satisfactorios en cuanto a desempeño.

Por último, en el capítulo 7 se dan algunas conclusiones sobre las limitaciones y alcances del presente trabajo. Dejando en claro cuales serían los aspectos a mejorar y el rumbo a seguir en futuras investigaciones. Donde no sólo se debe mejorar el esquema de control sino la plataforma experimental también.

Capítulo 2

Operación de un robot móvil (2,0)

2.1. Robots móviles.

La robótica móvil se encarga de estudiar la clase de robots que son capaces de desplazarse. Un tipo particular de robots móviles son aquellos que pueden conseguir un movimiento autónomo mediante el uso de sensores y actuadores controlados por un sistema computarizado a bordo del robot o en una posición remota. Existen dos principales maneras de lograr que un robot sea capaz de desplazarse, una de ellas es mediante el uso de extremidades o patas que emulan el comportamiento de diferentes animales o insectos. Los robots móviles con patas tienen la ventaja de poder desplazarse a través de terrenos difíciles, sin embargo se necesita mucha energía para impulsar sus varios actuadores, las ecuaciones que describen su movimiento son complejas e igual de complejos son los esquemas de control. En la figura 2.1 se muestra uno de los robots bípedos más famosos.

Figura 2.1: Robot bípedo ASIMO desarrollado por Honda Motor Company.

La otra manera de lograr el movimiento de los robots no está inspirado en la naturaleza sino que es meramente un invento del ser humano, la rueda. Los robots móviles que utilizan ruedas son especialmente populares, ya que como ha sido comprobado desde siglos atrás, los vehículos que implementan ruedas son capaces de desplazarse con facilidad sobre terrenos lisos principalmente pero también pueden adaptarse para sortear terrenos difíciles. Este tipo de robot móvil también resulta fácil de controlar

por lo sencillo de su descripción matemática, lo que los ha vuelto desde hace mucho tiempo objeto de una amplia investigación científica. En la figura 2.2 se muestra un robot móvil impulsado por ruedas que fue enviado al planeta Marte.

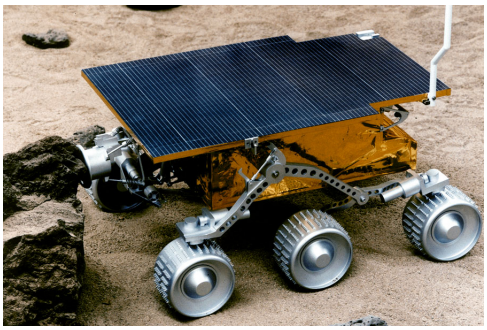


Figura 2.2: Sonda espacial Mars Pathfinder desarrolada por la Nasa.

Dentro de los robots móviles con ruedas, existen subcategorias, una de ellas son aquellos robots con sólo tres ruedas. Además, existen 5 tipos básicos de robots móviles de 3 ruedas los cuales se pueden diferenciar por el tipo de ruedas que ocupan así como por la disposición de dichas ruedas. En la figura 2.3 se muestran los 5 tipos diferentes de robots móviles de 3 ruedas, donde δM es el grado de maniobrabilidad, δm es el grado de movilidad y δs es el grado de direccionabilidad.

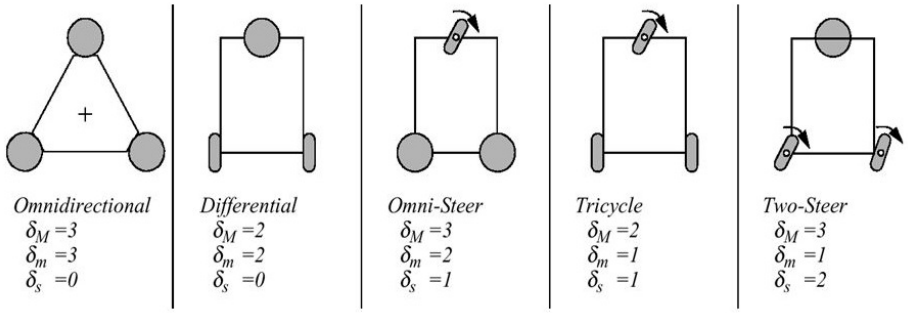


Figura 2.3: Tipos básicos de robots de 3 ruedas.

Para el estudio cinemático se asume que los robots son cuerpos rígidos con llantas no deformables las cuales se mantienen verticales y rotan alrededor de un eje horizontal. Se pueden considerar dos tipos de llantas, las ruedas convencionales y las ruedas suecas. Para las ruedas convencionales, se debe cumplir que el contacto entre la llanta y el suelo es de tal forma que no existe deslizamiento y puede rodar correctamente, esto significa que la velocidad del punto de contacto es igual a cero. Las ruedas suecas presentan un grado de libertad adicional ya que pueden moverse de forma lateral.

2.1.1. Robot móvil (2,0)

El robot móvil considerado, tipo diferencial (2,0) o uniciclo, cuenta con dos ruedas fijas alineadas. Dichas ruedas pueden girar de manera independiente a distintas velocidades, permitiendo que el robot avance cuando ambas ruedas giran a la misma velocidad y en el mismo sentido de giro, el robot puede girar hacia la derecha si ambas ruedas giran en el mismo sentido pero la magnitud de la velocidad de la rueda izquierda es mayor que la de la rueda derecha, de manera similar se puede lograr el giro a la izquierda, mientras para girar sobre sí mismo es necesario que las ruedas giren es sentido contrario con la misma velocidad. Además el robot cuenta con una tercera rueda tipo omnidireccional que sirve como un tercer punto de apoyo y brinda estabilidad. La cinemática del punto medio del eje del robot con retardo en la entrada está dada por [10]

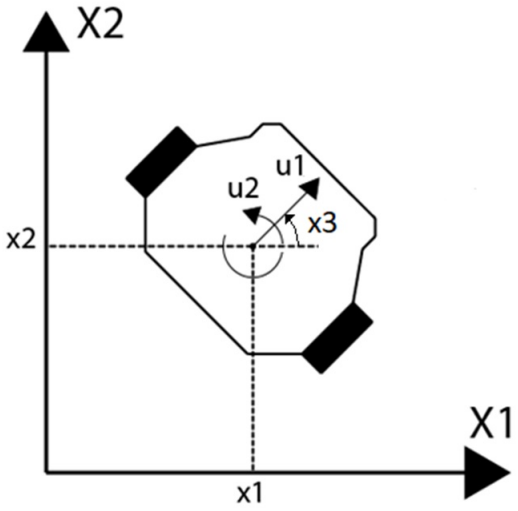


Figura 2.4: Robot móvil (2,0).

$$\begin{aligned} \dot{x}_1 &= u_1(t-\lambda)\cos x_3(t) \\ \dot{x}_2 &= u_1(t-\lambda)\sin x_3(t) \\ \dot{x}_3 &= u_2(t-\lambda) \end{aligned} \tag{2.1}$$

donde u_1 es la velocidad lineal del punto medio del eje imaginario que une ambas ruedas, u_2 la velocidad angular, x_1 es la coordenada horizontal, x_2 es la coordenada vertical, x_3 es el ángulo de orientación del robot con respecto al eje horizontal, t es el tiempo y λ es el retardo, el cual se considera constante y multiplo del periodo de muestreo. Se supone también que las ruedas del robot giran sin resbalar. Las velocidades u_1 y u_2 están relacionadas con las velocidades de rotación de cada una de las

ruedas por las siguientes ecuaciones:

$$u_1 = \frac{(w_d + w_i)r}{2}$$
$$u_2 = \frac{(w_d - w_i)r}{2l}) \tag{2.2}$$

donde w_d es la velocidad angular de la rueda derecha, w_i es la velocidad angular de la rueda izquierda, r es el radio de las ruedas, y $2l$ es la distancia perpendicular entre las ruedas impulsoras. La restricción no holónoma limita el movimiento del robot. Es decir, se puede alcanzar cualquier posición en el plano, pero la trayectoria está restringida por la ecuación

$$\dot{x}_2 \cos(x_3) - \dot{x}_1 \sin(x_3) = 0 \tag{2.3}$$

donde $\dot{x}_1$ y $\dot{x}_2$ son las velocidades en el eje horizontal y en el eje vertical.

2.2. Control de un robot móvil (2,0).

Los sistemas o plantas a controlar pueden representarse en tiempo discreto, esto quiere decir que las variables adquieren valor sólo en cada periodo de muestreo T, por lo tanto no son continuas. Los instantes de muestreo se denotan por kT o $t_k(k = 0, 1, 2...)$. Las ecuaciones que describen el comportamiento de un sistema pueden ser representadas en el espacio de estados en tiempo discreto mediante ecuaciones de diferencias. Para sistemas lineales o no lineales de tiempo discreto la ecuación de estado se puede escribir de la siguiente manera

$$x(k+1) = f[x(k), u(k), k]. \tag{2.4}$$

y la ecuación de salida como

$$y(k) = g[x(k), u(k), k] \tag{2.5}$$

donde $x(k)$ es el vector de estados y $u(k)$ es el vector de entradas.

Una de los métodos que existen para sintetizar leyes de control para un robot móvil (2,0) es partir del modelo cinemático en tiempo continuo dado por:

$$\begin{aligned}\dot{x}_1 &= u_1(t)\cos x_3(t)\\ \dot{x}_2 &= u_1(t)\sin x_3(t)\\ \dot{x}_3 &= u_2(t)\end{aligned} \tag{2.6}$$

Después, siguiendo el procedimiento descrito en [3], es posible pasar el sistema continuo (2.6) a su representación en tiempo discreto (2.7).

$$\begin{aligned}x_1^+ &= x_1 + Tu_1\psi(u_2)\cos(\gamma),\\ x_2^+ &= x_2 + Tu_1\psi(u_2)\sin(\gamma),\\ x_3^+ &= x_3 + Tu_2.\end{aligned} \tag{2.7}$$

Donde se usa la notación

$$x = x(kT) \quad y \quad x^+ = x(kT+T)). \tag{2.8}$$

con

$$\psi(u_2) = \frac{sin(\frac{T}{2}u_2)}{(\frac{T}{2}u_2)} \quad y \quad \gamma(x_3,u_2) = x_3 + \frac{T}{2}u_2. \tag{2.9}$$

A partir de las ecuaciones (2.7) se proponen las siguientes leyes de control [3]

$$\begin{aligned}u_1 &= \frac{1}{T\psi(u_2)}[(v_1-x_1)cos(\gamma(x_3,u_2)) + (v_2-x_2)sin(\gamma(x_3,u_2))],\\ u_2 &= \frac{1}{T}(v_3-x_3).\end{aligned} \tag{2.10}$$

con la finalidad de hacer el seguimiento de una trayectoria, o bien alcanzar una posición deseada en el plano. Las ecuaciones (2.10) llevan al sistema (2.7) a la siguiente forma

$$x_1^+ = v_1, \quad x_2^+ = v_2 \quad y \quad x_3^+ = v_3. \tag{2.11}$$

Donde v_1, v_2 y v_3 son controles auxiliares que pueden ser desde un controlador proporcional hasta un control más complicado. Se puede notar que existe una singularidad cuando $\psi(u_2) = 0$, este punto se explica con mayor detalle en el capítulo 3.

2.2.1. Teleoperación de un robot móvil (2,0).

La teleoperación significa en principio, como su nombre lo dice, operar una máquina, mecanismo, robot o sistema a distancia. Siendo esta definición muy ambigua.

Sin embargo la teleoperación tiene tantas variantes como aplicaciones, así que sería complicado dar una definición más exacta. Por lo tanto, para comprender un poco mejor la mayoría de los aspectos que comprenden la teleoperación, es necesario dar más definiciones. En la figura 2.5 se muestra un robot teleoperado que se utilizaba principalmente para levantar grandes cargas en la industria de la forja y la fundición.

- Operador: Un operador humano es la persona que monitorea la máquina controlada y toma las acciones de control necesarias.
- Esclavo: Es la máquina (o robot) controlada.
- Manipulación mecánica: Las señales de control son transmitidas mecánicamente o hidráulicamente al esclavo. La retroalimentación visual puede ser directa o proporcionada por un monitor.
- Control remoto: El operador tiene contacto visual la mayor parte del tiempo con el objetivo a controlar. Las señales de control son enviadas eléctricamente por cables o por ondas de radio.
- Telepresencia: Si una cantidad suficiente de información es recibida por el operador a través de los diversos sensores dispuestos en el sitio de trabajo del esclavo, entonces el operador puede sentir que está presente en el sitio de trabajo.
- Presencia virtual: Es similar a la telepresencia pero en este caso el operador siente estar en un ambiente de trabajo virtual.
- Realidad aumentada: Es una combinación de la información recibida del mundo real y de la realidad virtual.

Figura 2.5: Se muestra una máquina MAN-MATE, ejemplo de la teleoperación directa.

Para ayudar a comprender mejor el término de teleoperación, podemos definir los tres tipos principales de esquemas de teleoperación que existen:

- Teleoperación directa: El operador controla los actuadores del esclavo mediante señales de control directas y recibe retroalimentación en tiempo real. Esto sólo es posible cuando los retardos en el lazo de control son mínimos.
- Teleoperación coordinada: El operador controla los actuadores pero ahora existe algún lazo de control interno en el esclavo. Sin embargo el esclavo no es autónomo, sino que dicho lazo de control interno sirve para controlar aspectos que el operador es incapaz de mantener bajo control debido a la existencia de los retardos en la comunicación.
- Control supervisorio: La mayor parte del control lo realiza el esclavo, mientras que el operador principalmente supervisa las acciones de robot y da ordenes de alto nivel.

Para teleoperar un robot móvil (2,0) se puede usar un mando háptico, el cual nos permite asociar de manera proporcional las velocidades del robot con la posición del mando. Lo que nos permite variar las velocidades de traslación y rotación. Este tipo de mandos nos da la oportunidad de generar fuerzas que comúnmente se asocian a la distancia que existe entre el robot y algún obstáculo, [9], [11]. En la figura 2.6 se puede ver un esquema de un mando háptico de dos grados de libertad.

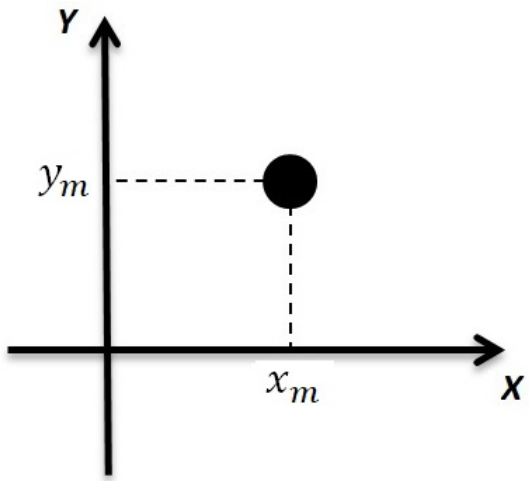


Figura 2.6: Esquema de un mando háptico.

Donde x_m es la coordenada horizontal del mando y y_m es la coordenada vertical. Entonces las velocidades quedan de la siguiente manera:

$$\begin{aligned} u_1 &= k_{xm} x_m, \\ u_2 &= k_{ym} y_m. \end{aligned} \tag{2.12}$$

Donde k_{xm} y k_{ym} son constantes positivas, por lo tanto, mientras mayor sea el valor

de las coordenadas, mayor será la magnitud de las velocidades.

Uno de los modelos más sencillos para generar fuerzas que nos ayuden a evitar la colisión con obstáculos es el que simula un resorte lineal, que depende de la distancia entre el obstáculo y el robot.

Sea r_0 la distancia del robot al obstáculo en la cual queremos que empiece a funcionar la retroalimentación de fuerza, y sea r la distancia del robot al obstáculo, entonces:

$$\Delta = r_0 - r. \tag{2.13}$$

Nótese que Δ puede ser menor que cero, sin embargo, cuando se encuentra dentro de ese conjunto de valores no afecta al esquema de teleoperación, como se verá más adelante. En la figura 2.7 se muestra un esquema de la distancia entre el robot y un obstáculo, donde x_{1obs} y x_{2obs} son las coordenadas del obstáculo. Las fuerzas de retroalimentación se definen como:

$$\begin{aligned} F_1 &= \begin{cases} k_{x1}\Delta + f_x, & para \quad r \leq r_0 \\ 0, & para \quad r > r_0 \end{cases} \\ F_2 &= \begin{cases} k_{y1}\Delta + f_y, & para \quad r \leq r_0 \\ 0, & para \quad r > r_0 \end{cases} \end{aligned} \tag{2.14}$$

Donde los componentes $k_{x1}\Delta$ y $k_{y1}\Delta$ son fuerzas que se oponen al incremento de las coordenadas x_m y y_m, por lo tanto se disminuye también la velocidad del robot, permitiendo al usuario cambiar la trayectoria para evitar una colisión. Mientras f_x y f_y son fuerzas que simulan a un amortiguador y ayudan al mando a disipar la energía, esto se hace para evitar posibles daños, pues si se llega soltar el mando mientras $r \leq r_0$ se podría dañar dicho dispositivo. Las fuerzas que simulan el amortiguador están definidas de la siguiente manera:

$$\begin{aligned} f_x &= c_{x2}\frac{dx_m}{dt}, \\ f_y &= c_{y2}\frac{dy_m}{dt}. \end{aligned} \tag{2.15}$$

Donde k_{x1}, k_{y1}, c_{x2} y c_{y2} son constantes menores que cero, entonces las fuerzas totales que se aplican al mando háptico quedan definidas de la siguiente manera:

$$\begin{aligned} F_{txm} &= F_1 + F_{1op}, \\ F_{tym} &= F_2 + F_{2op}. \end{aligned} \tag{2.16}$$

Donde F_{1op} y F_{2op} son las fuerzas aplicadas por el operador humano.

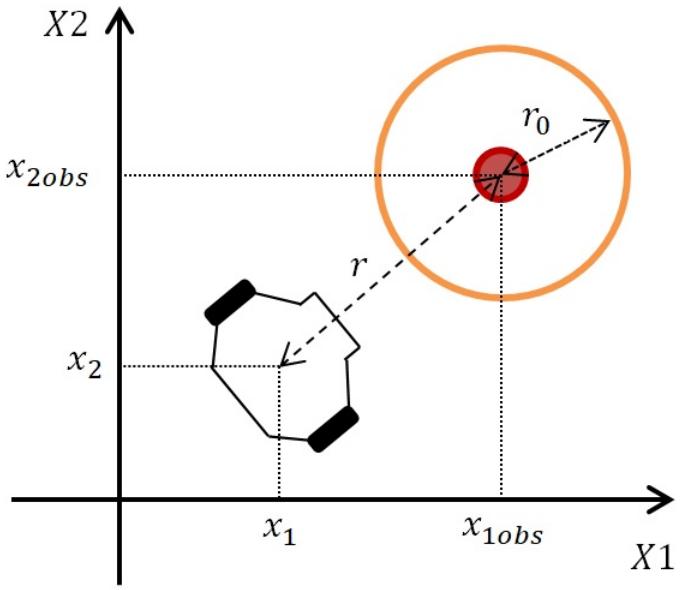


Figura 2.7: Robot y un obstáculo (en rojo).

Capítulo 3

Esquema de control discontinuo para un robot móvil (2,0)

En este capítulo se presenta un esquema de control con el proposito de que el robot alcance una posición final deseada, evadiendo un obstáculo fijo en la trayectoria seguida, también incluyendo retardos en las señales de control. Cabe señalar que el objetivo principal es alcanzar la posición deseada y evitar la colisión con el obstáculo.

A partir del modelo cinemático en tiempo discreto con retardos en la entrada para un robot móvil (2,0), se muestran dos formas distintas que se utilizaron para generar las trayectorias deseadas. Posteriormente, con base en los pasos anteriores se sintetiza el esquema de control. Finalmente se lleva a cabo un estudio sobre el error de seguimiento.

Antes de todo el desarrollo es necesario establecer algunas consideraciones con respecto al retardo en la comunicación. Estas consideraciones ayudan a simplificar el tratamiento del retardo [12], y son las siguientes:

- El tiempo de muestreo $T > 0$ es constante.
- Se define un intervalo de tiempo t_k como $t_k = \{t \in [kT, kT + T)\}$, donde $k = 0, 1, 2...$
- Se supone que las señales de control son constantes cada instante de muestreo, es decir $u(T) = u(kT), \forall t \in [kT, kT + T)$.
- El retardo en el tiempo t satisface $\lambda = \alpha T$,para algún entero $\alpha > 0$.

3.1. Modelo en tiempo discreto con retardos en la entrada.

Teniendo en cuenta las consideraciones anteriores para el retardo y partiendo del modelo (2.1) es posible llegar al modelo en tiempo discreto con retardos en la entrada

[13] siguiente:

$$\begin{aligned} x_1^+ &= x_1 + Tu_1^{-\alpha}\psi(u_2^{-\alpha})cos(\gamma(x_3, u_2^{-\alpha})), \\ x_2^+ &= x_2 + Tu_1^{-\alpha}\psi(u_2^{-\alpha})cos(\gamma(x_3, u_2^{-\alpha})), \\ x_3^+ &= x_3 + Tu_2^{-\alpha}, \end{aligned} \tag{3.1}$$

donde se usa la notación

$$\xi = \xi(kT), \xi^{[\pm]} = \xi(kT \pm T) \quad y \quad \xi^{\pm\alpha} = \xi(kT \pm \alpha T), \tag{3.2}$$

con

$$\psi(u_2^{-\alpha}) = \frac{sin(\frac{T}{2}u_2^{-\alpha})}{(\frac{T}{2}u_2^{-\alpha})}, \gamma(x_3, u_2^{-\alpha}) = x_3 + \frac{T}{2}u_2^{-\alpha}. \tag{3.3}$$

Una vez teniendo el modelo en tiempo discreto, se puede abordar el tratamiento de los retardos con mayor facilidad.

3.1.1. Modelo extendido en tiempo discreto.

Aunque el modelo del robot móvil (2,0) descrito por las ecuaciones (3.1) es un modelo exacto, en el sentido de que la discretización se obtiene a partir de la integración exacta de las ecuaciones (2.1), puede observarse que requiere de predicciones de variables futuras en el diseño de esquemas de control. Por esta razón, se considera un modelo discreto en el tiempo extendido equivalente. Con base en la consideración de que el retardo es un multiplo entero positivo del periodo de muestreo, se definen los nuevos estados

$$\begin{aligned} z_1(kT) &= u_1(kT - \lambda) = u_1^{-\alpha}, \\ z_2(kT) &= u_1(kT - \lambda + T) = u_1^{-\alpha+1}, \\ &\vdots \qquad\qquad \vdots \\ z_\alpha(kT) &= u_2(kT - \lambda + (\lambda - 1)T) = u_1^{-1}. \\ w_1(kT) &= u_2(kT - \lambda) = u_2^{-\alpha}, \\ w_2(kT) &= u_2(kT - \lambda + T) = u_2^{-\alpha+1}, \\ &\vdots \qquad\qquad \vdots \\ w_\alpha(kT) &= u_2(kT - \lambda + (\lambda - 1)T) = u_2^{-1}. \end{aligned} \tag{3.4}$$

De esta manera, se puede reescribir el modelo dado por (3.1) como

$$
\begin{aligned}
x_1^+ &= x_1 + Tz_1\psi(w_1)cos(\gamma(x_3, w_1)),\\
x_2^+ &= x_2 + Tz_1\psi(w_1)sin(\gamma(x_3, w_1)),\\
x_3^+ &= x_3 + Tw_1,\\
z_1^+ &= z_2,\\
z_2^+ &= z_3,\\
&\vdots \quad \vdots\\
z_\alpha^+ &= u_1,\\
w_1^+ &= w_2,\\
w_2^+ &= w_3,\\
&\vdots \quad \vdots\\
w_\alpha^+ &= u_2.
\end{aligned}
\tag{3.5}
$$

El modelo discreto extendido (3.5), de dimensión $3+2\alpha$, representa el modelo discreto exacto equivalente libre de retardos del modelo original en tiempo continuo del robot móvil (2,0) con retardo en las entradas. Nótese, en particular, que en el modelo (3.5) las entradas u_1 y u_2 aparecen en forma explícita hasta el $\alpha + 1$ adelanto de x_1 y x_2, más precisamente

$$
\begin{aligned}
x_1^{\alpha+1} &= x_1^\alpha + Tu_1\psi(u_2)cos(\gamma(x_3^\alpha, u_2)),\\
x_2^{\alpha+1} &= x_2^\alpha + Tu_1\psi(u_2)sin(\gamma(x_3^\alpha, u_2)),\\
x_3^{\alpha+1} &= x_3^\alpha + Tu_2.
\end{aligned}
\tag{3.6}
$$

Este hecho permite asegurar que la retroalimentación

$$
\begin{aligned}
u_1 &= \frac{1}{T\psi(u_2)}[(v_1 - x_1^\alpha)cos(\gamma(x_3^\alpha, u_2)) + (v_2 - x_2^\alpha)sin(\gamma(x_3^\alpha, u_2))],\\
u_2 &= \frac{1}{T}(v_3 - x_3^\alpha).
\end{aligned}
\tag{3.7}
$$

transforma al sistema de tal manera que la dinámica de las variables x_1, x_2 y x_3 esté dada por [5]

$$
x_1^{\alpha+1} = v_1, x_2^{\alpha+1} = v_2, x_3^{\alpha+1} = v_3, \tag{3.8}
$$

donde v_1, v_2 y v_3 son nuevas señales de entrada que se escogen para lograr que x_1, x_2 y x_3 sigan trayectorias deseadas o alcancen un punto de interés, tanto en posición como en orientación, como se propone en la sección siguiente. Es importante notar que la retroalimentación (3.7) tiene una singularidad cuando $\psi(u_2) = 0$. Ya que $\psi(u_2)$ tiende a 1 cuando u_2 tiende a 0, la singularidad puede solamente aparecer cuando $u_2 = \pm\frac{2}{T}l\pi$, para $l = 1, 2, \ldots$, esto es para valores altos de la velocidad angular u_2, lo que, a su vez, depende del periodo de muestreo T.

3.2. Generación de trayectorias deseadas.

En este trabajo se utilizaron dos maneras distintas para generar las trayectorias deseadas. La primera manera fue utilizando un robot virtual y la segunda fue usando el método de potenciales artificiales. Para el robot virtual se utilizó el modelo cinematico en tiempo discreto que se muestra a continuación:

$$\begin{aligned} x_{1d}^{+} &= x_{1_d} + Tu_{1d}\psi(u_{2d})\cos(\gamma(x_{3d}, u_{2_d})), \\ x_{2d}^{+} &= x_{2_d} + Tu_{1d}\psi(u_{2d})\sin(\gamma(x_{3d}, u_{2_d})), \\ x_{3d}^{+} &= x_{3_d} + Tu_{2d}, \end{aligned} \tag{3.9}$$

donde x_{1d},x_{2d} y x_{3d} son los estados deseados. La trayectoria generada depende directamente de las funciones que asignemos a las velocidades u_{1d} y u_{2d}, por ejemplo si queremos generar un círculo, bastará que las dos velocidades sean constantes. Se eligió esta forma de generar las trayectorias para asegurar que el robot fuera capaz de llevar a cabo la trayectoria deseada en lazo abierto.

3.2.1. Usando el método de potenciales artificiales.

Este método consiste en la creación de un campo potencial artificial en el cual la meta constituye un polo atractivo y los obstáculos representan polos repulsivos. El robot navega siguiendo el negativo del gradiente de la función potencial hasta alcanzar el mínimo global, es decir, la meta.

Sea $U(x)$ la función del potencial artificial, además $U(x) : R^n \to R$, tal que

$$U(x) = U_{att}(x) + U_{rep}(x), \tag{3.10}$$

donde x es la posición del robot, $U_{att}(x)$ es el potencial atractivo asociado a la meta, mientras que $U_{rep}(x)$ es el potencial repulsivo asociado a algún obstáculo. El potencial atractivo está dado por

$$U_{att}(x) = \frac{1}{2}\varepsilon d^2(x, x_g), \tag{3.11}$$

donde ε es un factor de escalamiento constante y positivo, $d(x, x_g)$ es una función cuya primera derivada es continua y tiene un mínimo global igual a cero en $x = x_g$, y x_g es un vector que contiene las coordenadas de la meta. Se define $d(x, x_g) = \|x - x_g\|$, es decir la distancia euclidiana entre el robot y la meta. Por lo tanto, la fuerza atractiva toma la siguiente forma

$$F_{att} = \begin{bmatrix} f_{1att} \\ f_{2att} \end{bmatrix} = -\nabla U_{att}(x) = -\Big[\nabla \frac{1}{2}\varepsilon\big[(x_1 - x_{1g})^2 + (x_2 - x_{2g})^2\big]\Big]^T = \begin{bmatrix} -\varepsilon(x_1 - x_{1g}) \\ -\varepsilon(x_2 - x_{2g}) \end{bmatrix}, \tag{3.12}$$

donde ∇ indica la derivada parcial del potencial atractivo respecto a la posición del robot móvil. El potencial repulsivo comúnmente se define como:

$$U_{rep}(x) = \begin{cases} \frac{1}{2}\eta(\frac{1}{p(x,x_{obs})} - \frac{1}{p_0})^2, & para \quad p(x, x_{obs}) \leq p_0, \\ 0, & para \quad p(x, x_{obs}) > p_0, \end{cases} \tag{3.13}$$

donde η es un factor de escala positivo, $p(x, x_{obs})$ es la distancia entre el robot y algún obstáculo, p_0 es la distancia a la cual se quiere empezar el campo repulsivo, y x_{obs} es un vector que contiene las coordenadas del obstáculo. Por lo tanto, la fuerza repulsiva cuando $p(x, x_{obs}) \leq p_0$ toma la forma

$$F_{rep} = -\nabla U_{rep}(x) = \eta(\frac{1}{p(x, x_{obs}} - \frac{1}{p_0})(\frac{1}{p^3(x, x_{obs})}(x - x_{obs})). \tag{3.14}$$

Entonces las componentes de la fuerza de repulsión son

$$\begin{aligned} F_{1rep}(x) &= \begin{cases} \eta(\frac{1}{p(x,x_{obs})} - \frac{1}{p_0})(\frac{1}{p^3(x,x_{obs})})(x_1 - x_{obs}), & para \quad p(x, x_{obs}) \leq p_0 \\ 0, & para \quad p(x, x_{obs}) > p_0, \end{cases} \\ F_{2rep}(x) &= \begin{cases} \eta(\frac{1}{p(x,x_{obs})} - \frac{1}{p_0})(\frac{1}{p^3(x,x_{obs})})(x_2 - x_{obs}), & para \quad p(x, x_{obs}) \leq p_0 \\ 0, & para \quad p(x, x_{obs}) > p_0. \end{cases} \end{aligned} \tag{3.15}$$

Finalmente, la fuerza total queda definida como

$$F_t = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = F_{att}(x) + F_{rep}(x). \tag{3.16}$$

Es importante, notar que para el cálculo del control de un robot móvil (2,0) se requiere de las posiciones deseadas x_{1d}, x_{2d}, los adelantos deseados para la orientación y también las posiciones siguientes deseadas, las cuales no se obtienen directamente al usar el método de potenciales artificiales sino que se debe definir una trayectoria que el robot sea capaz de seguir. Entonces para obtener $x_{id}^{+}, \ldots, x_{id}^{\alpha+1}$, $i = 1, 2$, se calcula la línea recta que debería seguir el robot con el gradiente obtenido en la posición actual, recalculando el gradiente y la línea recta cada periodo de muestreo. Los adelantos requeridos para x_{1d} y x_{2d} están entonces dados por

$$x_{id}^{n+1} = x_{id}^{n} + TV_d \frac{f_i}{\sqrt{f_1^2 + f_2^2}}, \tag{3.17}$$

con $i = 1, 2$, y $n = 0, 1, \ldots, \alpha$. Las posiciones deseadas actuales x_{1d} y x_{2d} se obtienen bajo la suposición de que son iguales a las posiciones actuales esto es $x_{1d} = x_1$ y $x_{2d} = x_2$. Nótese que la expresión (3.17) tiene una singularidad cuando ambas fuerzas son iguales a cero, esto sucede cuando $d(x, x_g) = 0$ y $p(x, x_{obs}) > p_0$; en la práctica se impone un valor mínimo a la distancia entre el robot y la meta para evitar que sea cero. Ya que no existe restricción alguna para la orientación deseada, se utiliza la aproximación (3.18), donde primero se calculan las velocidades traslacionales deseadas en X_1 y X_2, a partir de dichas velocidades se obtiene el ángulo de la velocidad tangencial deseada del robot, el cual coincide con la orientación deseada. Sin embargo también se puede utilizar la orientación deseada como

$$x_{3d} = tan^{-1}\left[\frac{\frac{x_{1d}^{+} - x_{1d}}{T}}{\frac{x_{2d}^{+} - x_{2d}}{T}}\right]. \tag{3.18}$$

La orientación deseada adelantada se calcula entonces como

$$x_{3d}^{n+1} = x_{3d}^{n} + Ttan^{-1}\left[\frac{\frac{x_{1d}^{+} - x_{1d}}{T}}{\frac{x_{2d}^{+} - x_{2d}}{T}}\right], \quad n = 0, 1, 2, \ldots, \alpha, \tag{3.19}$$

V_d es la velocidad dada por

$$V_d = k_v\left[d(x, x_g) - \frac{k_0}{p(x, x_{obs})}\right], \tag{3.20}$$

donde k_v es una constante positiva que determina la velocidad con la que se desplaza el robot hacia la meta, mientras que k_0 es una constante positiva que disminuye la velocidad del robot cuando éste se aproxima al obstáculo, para prevenir una posible colisión. Para evitar que V_d tome valores negativos, se selecciona el valor de la constante k_0 de tal manera que el cociente $k_0/p(x, x_{obs}) < d(x, xobs)$. En la figura 3.1 se muestra un esquema del robot, un obstáculo y la meta, cabe mencionar que $p_0 = r_0$ y $p = r$.

3.3. Síntesis del esquema de control discontinuo.

De acuerdo a [6] se propone un controlador por modos deslizantes que hace que el robot siga trayectorias deseadas, x_{id}, $i = 1, 2$. Se consideran entonces los errores de seguimientos definidos como

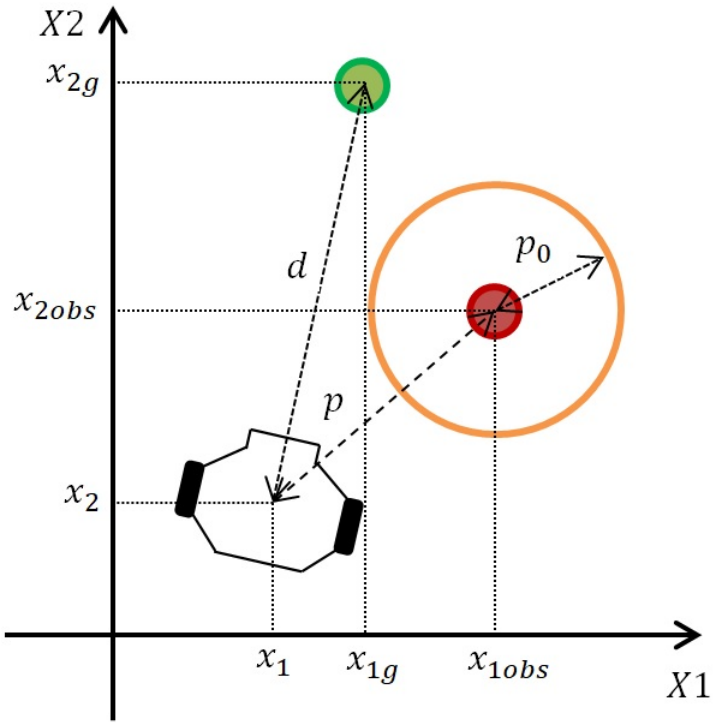


Figura 3.1: Robot, obstáculo y meta.

$$\begin{aligned} e_i &= (x_i - x_{id}), \\ e_i^+ &= (x_i^+ - x_{id}^+), \\ &\vdots \\ e_i^{\alpha+1} &= (x_i^{\alpha+1} - x_{id}^{\alpha+1}) = v_i - x_{id}^{\alpha+1}, \end{aligned} \tag{3.21}$$

con $i = 1, 2, 3$. Se consideran también las siguientes funciones coordenadas que describen superficies deslizantes:

$$\sigma_i = -k_{i-1}e_i^- - k_{i0}e_i - k_{i1}e_i^+ - \cdots - k_{i\alpha-1}e_i^{\alpha-1} + e_i^\alpha, \quad i = 1, 2, 3. \tag{3.22}$$

De esta manera, si la evolución de σ_i se restringe a cero, las dinámicas de los errores de seguimiento están caracterizadas por las ecuaciones

$$-k_{i-1}e_i^- - k_{i0}e_i - k_{i1}e_i^+ - \cdots - k_{i\alpha-1}e_i^{\alpha-1} + e_i^\alpha = 0, \quad i = 1, 2, 3. \tag{3.23}$$

donde los coeficientes reales $k_{i-1}, k_{i0}, \ldots, k_{i\alpha-1}$, se escogen de tal manera que las soluciones de (3.24) tiendan asintóticamente a cero. Las condiciones para que las funciones

σ_i califiquen como superficies deslizantes son; en primer lugar que las funciones σ_i sean funciones escalares suaves y que las restricciones $\sigma_i = 0$ se cumplan al menos localmente (es decir en una vecindad de $\sigma_i = 0$). La manera de lograr esto es imponiendo en la evolución de cada σ_i la dinámica no lineal [6]

$$\sigma_i^+ = \Gamma_i(\sigma_i) = \begin{cases} K_i sign(\sigma_i), & para \quad |\sigma_i| > A_i, \\ \frac{K_i}{A_i - B_i}(|\sigma_i| - B_i)(sign(\sigma_i)), & para \quad B_i \leq |\sigma_i| \leq A_i \ , \\ 0, & para \quad |\sigma_i| < B_i, \end{cases} \quad i = 1, 2, 3, \tag{3.24}$$

donde K_i, A_i, B_i son coeficientes reales positivos con $A_i > B_i$ y $sign(\cdot)$ es la función signo. Se puede decir que las trayectorias del sistema $\sigma_i^+ = \Gamma(\sigma_i)$ tienden asintótica y globalmente hacia cero en tiempo finito si y sólo si $K_i < A_i$. Más aún, cuando $B_i \leq |\sigma_i| \leq A_i$, σ_i converge globalmente a cero en sólo un paso si y sólo si $K_i < B_i$. (véase [6], apéndice A1, para los detalles de la demostración). Al imponer entonces la evolución (3.24) se obtienen, a partir de (3.21), las señales v_i, es decir

$$v_i = \Gamma_i(\sigma_i) + x_{id}^{\alpha+1} + \sum_{j=0}^{\alpha} k_{ij}(x_i^j - x_{id}^j), \tag{3.25}$$

donde a partir de (3.5)

$$\begin{aligned} x_1^j &= x_1^{j-1} + T z_j \psi(w_j) cos(\gamma(x_3^{j-1}, w_j)), \\ x_2^j &= x_2^{j-1} + T z_j \psi(w_j) cos(\gamma(x_3^{j-1}, w_j)), \\ x_3^j &= x_3^{j-1} + T w_j, \end{aligned} \tag{3.26}$$

para $j = 1, \ldots, \alpha$. El control discontinuo completo está entonces formado por las expresiones (3.7), (3.25) y (3.26).

3.3.1. Análisis del error de seguimiento.

Es importante analizar el comportamiento dinámico de los errores de seguimiento (3.22) al utilizar el control dado por (3.7), (3.25) y (3.26). Para esto, se considera el sistema descrito por las ecuaciones (3.6) con la retroalimentación dada por las ecuaciones (3.7), además de las señales v_i descritas por (3.26). Se sustituye primero los controles en el estado x_3 a fin de escribir su ecuación sólo en términos del error de seguimiento e_i, esto es

$$\begin{aligned} x_3^{\alpha+1} &= x_3^{\alpha} + T\Big(\frac{1}{T}(v_3 - x_3^{\alpha})\Big) = v_3 \\ &= \Gamma_3(\sigma_3) + x_{3d}^{\alpha+1} + \sum_{j=0}^{\alpha} k_{3j}(x_3^j - x_{3d}^j) \end{aligned} \tag{3.27}$$

Obteniéndose entonces la siguiente ecuación que es función del error

$$e_3^{\alpha+1} + \sum_{j=0}^{\alpha} k_{3j}(e_3^j) + \Gamma_3(\sigma_3) = 0 \tag{3.28}$$

Entonces, eligiendo de manera adecuada los valores k_{3j} y sabiendo que $\Gamma_3(\sigma_3) \leq K_3$ para $t \geq 0$, se puede decir que el error de seguimiento del estado x_3 tiende a cero rápidamente si no existen perturbaciones. Siguiendo un procedimiento similar se puede llegar a las ecuaciones del error para los estados x_1 y x_2. Para el primer estado se tiene que

$$x_1^{\alpha+1} = x_1^{\alpha} + \Big([(v_1 - x_1^{\alpha})cos(\gamma(x_3^{\alpha}, u_2)) - (v_2 - x_2^{\alpha})sin(\gamma(x_3^{\alpha}, u_2))]\Big)cos(\gamma(x_3^{\alpha}, u_2)). \tag{3.29}$$

Sabiendo que

$$cos^2(\gamma(x_3^{\alpha}, u_2)) = 1 - sin^2(\gamma(x_3^{\alpha}, u_2)), \tag{3.30}$$

Se obtiene

$$x_1^{\alpha+1} = v_1 - (v_1 - x_1^{\alpha})sin^2(\gamma(x_3^{\alpha}, u_2)) + (v_2 - x_2^{\alpha})cos(\gamma(x_3^{\alpha}, u_2))sin(\gamma(x_3^{\alpha}, u_2)). \tag{3.31}$$

Entonces

$$\begin{aligned} x_1^{\alpha+1} &= \Gamma_1(\sigma_1) + x_{1d}^{\alpha+1} + \sum_{j=0}^{\alpha} k_{1j}(x_1^j - x_{1d}^j) - (v_1 - x_1^{\alpha})sin^2(\gamma(x_3^{\alpha}, u_2)) \\ &+ (v_2 - x_2^{\alpha})cos(\gamma(x_3^{\alpha}, u_2))sin(\gamma(x_3^{\alpha}, u_2)). \end{aligned} \tag{3.32}$$

Definiendo ahora

$$\begin{aligned} a &= x_3^{\alpha} + \frac{T}{2}u_2 = \frac{1}{2}(v_3 + x_3^{\alpha}) = \frac{1}{2}(v_3 + e_3^{\alpha} + x_{3d}^{\alpha}) \\ &= \frac{1}{2}\Big(x_{3d}^{\alpha+1} + x_{3d}^{\alpha} + \Gamma_3(\sigma_3) + \sum_{j=0}^{\alpha} k_{3j}(e_3^j) + e_3^{\alpha}\Big). \end{aligned} \tag{3.33}$$

También, se puede reescribir

$$v_1 - x_1^{\alpha} = x_{1d}^{\alpha+1} - x_{1d}^{\alpha} + \Gamma_1(\sigma_1) + x_{1d}^{\alpha+1} + \sum_{j=0}^{\alpha} k_{1j}(e_1^j) + e_1^{\alpha}. \tag{3.34}$$

Tomando en consideración las siguientes igualdades trigonométricas:

$$\begin{aligned} sin^2(a) &= \frac{1}{2}(1 - 2cos(a)), \\ cos^2(a) &= \frac{1}{2}(1 + 2cos(a)), \\ sin(a)cos(a) &= \frac{1}{2}sin(2a), \end{aligned} \tag{3.35}$$

y al sustituir (3.34) y (3.35) en (3.33), además de valernos de las igualdades (3.36), es posible obtener

$$\begin{aligned} e_1^{\alpha+1} &= k_{1\alpha}e_1^{\alpha} - \frac{1}{2}(k_{1\alpha}-1)e_1^{\alpha} + \frac{1}{2}\Gamma_1 + \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{1j}(e_1^j) - \frac{1}{2}(x_{1d}^{\alpha+1} - x_{1d}^{\alpha}) \\ &+ \frac{1}{2}(x_{1d}^{\alpha+1} - x_{1d}^{\alpha})cos(2a) + \frac{1}{2}\Gamma_1 cos(2a) + \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{1j}(e_1^j)cos(2a) \\ &+ \frac{1}{2}(k_{1\alpha}-1)e_1^{\alpha}cos(2a) + \frac{1}{2}(x_{2d}^{\alpha+1} - x_{2d}^{\alpha})sin(2a) + \frac{1}{2}\Gamma_2 sin(2a) \\ &+ \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{2j}(e_2^j)sin(2a) + \frac{1}{2}(k_{2\alpha}-1)e_2^{\alpha}sin(2a). \end{aligned} \tag{3.36}$$

De manera similar se puede llegar a la siguiente ecuación

$$\begin{aligned} e_2^{\alpha+1} &= k_{2\alpha}e_2^{\alpha} - \frac{1}{2}(k_{2\alpha}-1)e_2^{\alpha} + \frac{1}{2}\Gamma_2 + \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{2j}(e_2^j) - \frac{1}{2}(x_{2d}^{\alpha+1} - x_{2d}^{\alpha}) \\ &- \frac{1}{2}(x_{2d}^{\alpha+1} - x_{2d}^{\alpha})cos(2a) - \frac{1}{2}\Gamma_2 cos(2a) - \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{2j}(e_2^j)cos(2a) \\ &- \frac{1}{2}(k_{2\alpha}-1)e_2^{\alpha}cos(2a) + \frac{1}{2}(x_{1d}^{\alpha+1} - x_{1d}^{\alpha})sin(2a) + \frac{1}{2}\Gamma_1 sin(2a) \\ &+ \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{1j}(e_1^j)sin(2a) + \frac{1}{2}(k_{1\alpha}-1)e_1^{\alpha}sin(2a). \end{aligned} \tag{3.37}$$

Definiendo la igualdad

$$2a = m + n, \tag{3.38}$$

y utilizando las siguientes identidades trigonométricas:

$$\begin{aligned} cos(m+n) &= cos(m)cos(n) - sin(m)sin(n), \\ sin(m+n) &= sin(m)cos(n) + cos(m)sin(n), \end{aligned} \tag{3.39}$$

se obtienen las ecuaciones que describen la dinámica de los errores de seguimiento e_1 y e_2 como

$$\begin{aligned} e_1^{\alpha+1} &= \frac{1}{2}(k_{1\alpha}+1)e_1^{\alpha} + h_1(e_1^j, e_2^j, e_3^j) + h_2(x_{1d}^p, x_{2d}^p, e_3^j), \\ e_2^{\alpha+1} &= \frac{1}{2}(k_{1\alpha}+1)e_2^{\alpha} + h_3(e_1^j, e_2^j, e_3^j) + h_4(x_{1d}^p, x_{2d}^p, e_3^j), \end{aligned} \tag{3.40}$$

donde

$$
\begin{aligned}
h_1 &= \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{1j}e_1^j + \frac{1}{2}\Gamma_1 + \frac{1}{2}\Gamma_1 cos(m)cos(n) - \frac{1}{2}\Gamma_1 sin(m)sin(n) \\
&+ \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{1j}(e_1^j)cos(m)cos(n) - \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{1j}(e_1^j)sin(m)sin(n) + \frac{1}{2}(k_{1\alpha}-1)e_1^\alpha cos(m)cos(n) \\
&- \frac{1}{2}(k_{1\alpha}-1)e_1^\alpha sin(m)sin(n) + \frac{1}{2}\Gamma_2 sin(m)cos(n) + \frac{1}{2}\Gamma_2 cos(m)sin(n) \\
&+ \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{2j}(e_2^j)sin(m)cos(n) + \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{2j}(e_2^j)cos(m)sin(n) \\
&+ \frac{1}{2}(k_{2\alpha}-1)e_2^\alpha sin(m)cos(n) + \frac{1}{2}(k_{2\alpha}-1)e_2^\alpha cos(m)sin(n),
\end{aligned}
\tag{3.41}
$$

$$
\begin{aligned}
h_2 &= -\frac{1}{2}(x_{1d}^{\alpha+1} - x_{1d}^\alpha) + \frac{1}{2}(x_{1d}^{\alpha+1} - x_{1d}^\alpha)cos(m)cos(n) - \frac{1}{2}(x_{1d}^{\alpha+1} - x_{1d}^\alpha)sin(m)sin(n) \\
&+ \frac{1}{2}(x_{2d}^{\alpha+1} - x_{2d}^\alpha)sin(m)cos(n) + \frac{1}{2}(x_{2d}^{\alpha+1} - x_{2d}^\alpha)cos(m)sin(n),
\end{aligned}
\tag{3.42}
$$

$$
\begin{aligned}
h_3 &= \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{2j}e_2^j + \frac{1}{2}\Gamma_2 - \frac{1}{2}\Gamma_2 cos(m)cos(n) + \frac{1}{2}\Gamma_2 sin(m)sin(n) \\
&- \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{2j}(e_2^j)cos(m)cos(n) + \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{2j}(e_2^j)sin(m)sin(n) - \frac{1}{2}(k_{2\alpha}-1)e_2^\alpha cos(m)cos(n) \\
&+ \frac{1}{2}(k_{2\alpha}-1)e_2^\alpha sin(m)sin(n) + \frac{1}{2}\Gamma_1 sin(m)cos(n) + \frac{1}{2}\Gamma_1 cos(m)sin(n) \\
&+ \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{1j}(e_1^j)sin(m)cos(n) + \frac{1}{2}\sum_{j=0}^{\alpha-1} k_{1j}(e_1^j)cos(m)sin(n) \\
&+ \frac{1}{2}(k_{1\alpha}-1)e_1^\alpha sin(m)cos(n) + \frac{1}{2}(k_{1\alpha}-1)e_1^\alpha cos(m)sin(n),
\end{aligned}
\tag{3.43}
$$

$$
\begin{aligned}
h_4 &= \frac{1}{2}(x_{2d}^{\alpha+1} - x_{2d}^\alpha) - \frac{1}{2}(x_{2d}^{\alpha+1} - x_{2d}^\alpha)cos(m)cos(n) + \frac{1}{2}(x_{2d}^{\alpha+1} - x_{2d}^\alpha)sin(m)sin(n) \\
&+ \frac{1}{2}(x_{1d}^{\alpha+1} - x_{1d}^\alpha)sin(m)cos(n) + \frac{1}{2}(x_{1d}^{\alpha+1} - x_{1d}^\alpha)cos(m)sin(n).
\end{aligned}
\tag{3.44}
$$

De manera semejante al análisis presentado en [14] y [15] se define el vector

$$e_T = [e_1, e_1^+, ..., e_1^{\alpha-1}, e_2, e_2^+, ..., e_2^{\alpha-1}]^T, \tag{3.45}$$

$$e = [e_1^\alpha, e_2^\alpha]^T, \tag{3.46}$$

se puede ver que los términos h_1, h_2, h_3 y h_4 están acotados como

$$\| h_1 \| \leq k_{P1} \| e_T \| + k_{P2}, \tag{3.47}$$

$$\| h_2 \| \leq k_{P3}, \tag{3.48}$$

$$\| h_3 \| \leq k_{P4} \| e_T \| + k_{P5}, \tag{3.49}$$

$$\| h_4 \| \leq k_{P6}. \tag{3.50}$$

donde los valores $k_{P1}, k_{P2}, k_{P3}, k_{P4}, k_{P5}$ y $k_{P6} \in \mathbb{R}^+$. A partir de las ecuaciones (3.41), la dinámica de e_1 y e_2 se puede expresar entonces como

$$e^+ = Ae + H_1 + H_2, \tag{3.51}$$

donde

$$H_1 = \begin{bmatrix} h_1(KT) \\ h_3(KT) \end{bmatrix}, \quad H_2 = \begin{bmatrix} h_2(KT) \\ h_4(KT) \end{bmatrix} \quad y \quad A = \begin{bmatrix} \frac{1}{2}(k_{1\alpha}+1) & 0 \\ 0 & \frac{1}{2}(k_{2\alpha}+1) \end{bmatrix}. \tag{3.52}$$

Al observar la evolución del sistema (3.52) en el tiempo, se tiene que

$$\begin{aligned} e(1) &= Ae(0) + H_1(0) + H_2(0), \\ e(2) &= A^2e(0) + AH_1(0) + AH_2(0) + H_1(1) + H_2(1), \\ e(3) &= A^3e(0) + A^2H_1(0) + A^2H_2(0) + AH_1(1) + AH_2(1) + H_1(2) + H_2(2) \\ &= A^3e(0) + A^2H_{1,2}(0) + AH_{1,2}(1) + H_{1,2}(2), \\ e(4) &= A^4e(0) + A^3H_{1,2}(0) + A^2H_{1,2}(1) + AH_{1,2}(2) + H_{1,2}(3). \end{aligned} \tag{3.53}$$

Generalizando la evolución anterior se puede escribir

$$e(n) = A^ne(0) + A^{n-1}H(0) + A^{n-2}H(1) + ... + AH(m-2) + H(m-1), \tag{3.54}$$

con

$$H(m) = H_1(m) + H_2(m). \tag{3.55}$$

Entonces, la solución del sistema (3.56) está dada por

$$e(k) = A^ke(0) + \sum_{m=0}^{k-1} A^{(k-1-m)}H(m). \tag{3.56}$$

A partir de (3.47) a (3.51) $H(m)$ puede acotarse como

$$\| H(m) \| \leq k_{P7} \| e_T(m) \| + k_{P8}, \tag{3.57}$$

donde k_{P7} y K_{P8} son constantes. Mayorando entonces (3.56), se obtiene

$$\| e(k) \| \leq \| A \|^k \| e(0) \| + \sum_{m=0}^{k-1} \| A \|^{(k-1-m)} (k_{P7} \| e_T(m) \| + k_{P8}). \tag{3.58}$$

Siguiendo nuevamente el análisis desarrollado en [14] y [15] se considera el valor máximo de $\| e(k) \|$ para todo k dado, igual a $\| e(k) \|_s = sup \| e(k) \|$, $k = 1, 2, 3...$, teniéndose que

$$\| e(k) \|_s \leq \| A \|^k \| e(0) \| + \sum_{m=0}^{k-1} \| A \|^{(k-1-m)} (k_{P7} \| e_T(m) \|_s + k_{P8}). \tag{3.59}$$

Debido a que los valores propios de la matriz A satisfacen $\lambda_i\{A\} < 1$. Es claro que

$$\begin{aligned} &\lim_{k \to \infty} \left(\| A \|^k \right) = 0, \\ &\lim_{k \to \infty} \left(\sum_{m=0}^{k-1} \| A \|^{(k-1-m)} \right) = \frac{1}{1 - \| A \|}. \end{aligned} \tag{3.60}$$

De lo anterior, se obtiene entonces

$$\| e(k) \|_s \leq \frac{k_{P7} \| e_T(m) \|_s + k_{P8}}{1 - \| A \|}, \tag{3.61}$$

y, considerando que $\| e(k) \| \leq \| e(k) \|_s$, se obtiene finalmente

$$\| e(k) \|_s \leq \frac{k_{P8}}{1 - \| A \| - k_{P7}}. \tag{3.62}$$

Esta última expresión indica que el sistema es prácticamente estable bajo la condición de que

$$\| A \| + k_{P7} < 1. \tag{3.63}$$

Esta conclusión es similar a la encontrada en [14] y [15], la cual permite asegurar que los errores de seguimiento alcanzan una vecindad del origen.

Capítulo 4

Teleoperación de ganancia variable para un robot móvil (2,0)

La manera de teleoperar el robot móvil fue igual a la que se muestra en la ecuación (2.10) donde la velocidad traslacional u_1 depende de la constante positiva k_{xm} y de la coordenada horizontal del mando x_m. Mientras la velocidad rotacional u_2 depende de la constante positiva k_{ym} y de la coordenada vertical del mando y_m. Este esquema es suficiente para que un usuario pueda teleoperar al robot libremente, sin embargo es necesario incluir un método de retroalimentación de fuerzas para evitar las posibles colisiones entre el robot y algún obstáculo. El método seleccionado fue el de fuerza de retroalimentación con ganancia variable [9].

4.1. Teleoperación con fuerza de retroalimentación de ganancia variable.

En este caso particular se busca solamente generar una fuerza de retroalimentación que afecte a la velocidad traslacional, pues de esta manera se simplifica el problema. Además, limitando la velocidad traslacional a través de la fuerza de retroalimentación es posible evitar las colisiones sin tener que incluir otra fuerza para la velocidad rotacional.

El esquema de teleoperación que se presenta aquí, se basa primero en la generación de la fuerza F_x que depende de la distancia $p(x, x_{obs})$, de la velocidad de aproximación dr/dt y del ángulo entre el robot y el obstáculo β.

Se define entonces la ganancia variable k_x como

$$k_x = \begin{cases} k_{xmin}, & para \quad \frac{dr}{dt} \geq 0 \\ \frac{1}{\gamma_x}(k_{xmax} - k_{xmin})\frac{dr}{dt} + k_{xmin}, & para \quad -\gamma_x < \frac{dr}{dt} < 0 \\ k_{xmax}, & para \quad \frac{dr}{dt} \leq -\gamma_x \end{cases} \tag{4.1}$$

Continuamos con la definición de una fuerza auxiliar

$$f_{1aux} = \begin{cases} \frac{k_x cos(\beta)}{r}, & para \quad r \leq r_0 \\ \frac{1}{d(x,x_g)}, & para \quad r > r_0 \end{cases} \tag{4.2}$$

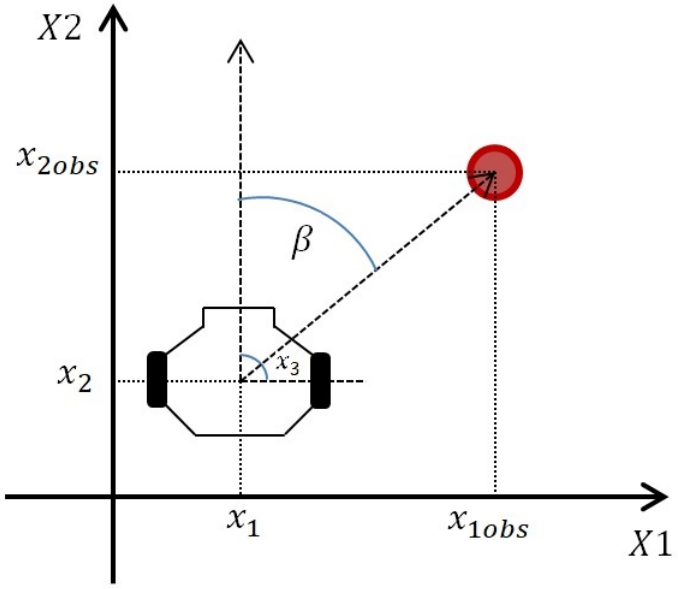


Figura 4.1: Ángulo de aproximamiento.

Finalmente, la fuerza de retroalimentación de ganancia variable queda de la siguiente manera:

$$F_x = -f_{1aux} - c_1 \frac{dx_m}{dt} \tag{4.3}$$

Donde c_1 es una constante positiva y dx_m/dt es la velocidad en la coordenada horizontal del mando háptico, en conjunto simulan un amortiguador que ayuda al mando a disipar la energía cuando no existe otra fuerza en este eje.
Para la velocidad rotacional no se utilizó una fuerza de retroalimentación pero sí se necesitó de la fuerza que simula un amortiguador F_y para ayudar al mando a disipar la energía en el eje vertical. Cabe mencionar que el mando háptico que se utilizó cuenta con tres grados de libertad, por lo que fue necesario incluir una fuerza de amortiguamiento F_z que disipara la energía en este eje. Las fuerzas quedaron de la siguiente forma:

$$\begin{aligned} F_y &= -c_2 \frac{dy_m}{dt} \\ F_z &= -c_3 \frac{dz_m}{dt} \end{aligned} \tag{4.4}$$

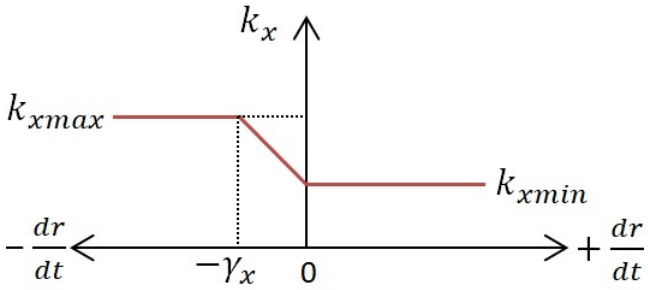


Figura 4.2: Ganancia variable.

Donde c_2 y c_3 son constantes positivas, dy_m/dt es la velocidad del mando en la coordenada vertical y dz_m/dt es la velocidad del mando en el eje z_m.

4.1.1. Teleoperación más control automático.

Para combinar el uso del control automático con la teleoperación utilizamos las siguientes ecuaciones:

$$\begin{aligned} u_1 &= \begin{cases} u_{1aut}, & para \quad p \geq p_0 \\ u_{1tel}, & para \quad p < p_0 \end{cases} \\ u_2 &= \begin{cases} u_{2aut}, & para \quad p \geq p_0 \\ u_{2tel}, & para \quad p < p_0 \end{cases} \end{aligned} \tag{4.5}$$

Donde

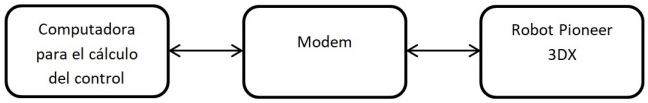
$$\begin{aligned} u_{1aut} &= \frac{1}{T\psi(u_2)}[(v_1 - x_1^\alpha)cos(\gamma(x_3^\alpha, u_2)) - (v_2 - x_2^\alpha)sin(\gamma(x_3^\alpha, u_2))], \\ u_{1tel} &= k_{xm}xm, \\ u_{2aut} &= \frac{1}{T}(v_3 - x_3^\alpha), \\ u_{2tel} &= k_{ym}ym. \end{aligned} \tag{4.6}$$

Esto quiere decir que cuando el robot se encuentre fuera del campo de repulsión virtual generado por el obstáculo, el esquema de control será automático, siendo ayudado por el método de potenciales artificiales para generar las trayectorias deseadas que lleven a la meta. Por otra parte, una vez que el robot entra al campo de repulsión virtual, el control es realizado por el usuario humano, siendo el esquema de teleoperación de retroalimentación de fuerza de ganancia variable.

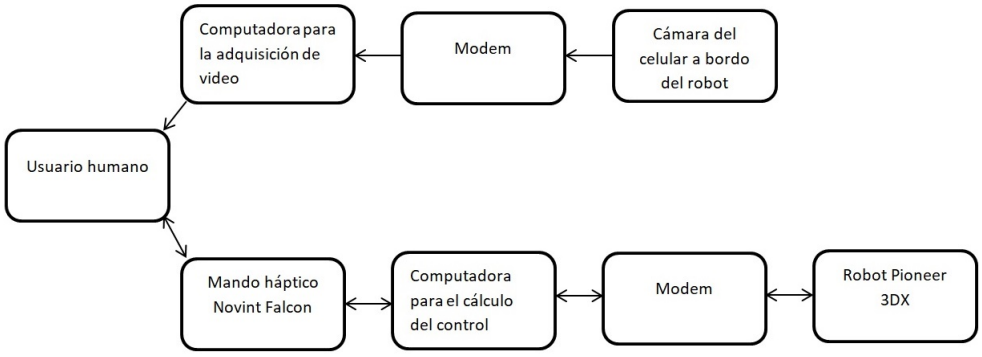
Capítulo 5

Plataforma experimetal

En este capítulo se da una breve explicación acerca de la plataforma experimental integrada para validar las estrategias de control diseñadas en este trabajo. En la figura 5.1 se muestran las dos estrategias que se utilizaron para controlar al robot. En el modo automático no interviene el operador humano. En el modo automático más teleoperado, el usuario interviene cuando el robot se aproxima demasiado a algún obstáculo.



(a) Automático.



(b) Automático más teleoperado.

Figura 5.1: Esquemas de control.

5.1. Robot Pioneer 3-DX

Los robots tipo Pioneer 3-DX de la marca Mobile Robots se utilizan en la investigación de la robótica móvil desde hace ya unos años debido a su fácil programación y resistencia, lo cual los convierte en una alternativa ideal para su uso dentro del laboratorio. Dicho robot cuenta con dos ruedas sólidas rellenas de espuma. Cada una de las ruedas tiene movimiento diferencial y un encoder de 500 pulsos de resolución. Se puede alcanzar hasta $1.2m/s$ de velocidad lineal y una velocidad de rotación de hasta $300°/s$, las llantas son de $19.5cm$ de diámetro. El robot puede soportar una carga de hasta 23 kg e incorpora 8 sensores ultrasónicos, 3 baterías de 12 v y 7.2 Ah cada una, con las cuales se tiene una autonomía de 8 a 10 horas dependiendo de la carga que se este soportando.

El robot cuenta además con un controlador integrado tipo PID para el movimiento de las ruedas que evita el daño de los motores por una posible aceleración excesiva. Éste controlador regula la velocidad lineal y angular del robot. También, su computadora integrada proporciona la estimación relativa de la posición del robot (x_1, x_2, x_3) gracias a los encoders de cada motor, muestra además los datos sobre la carga de la batería e información adquirida por los sonares que sirven para la detección de obstáculos. En éste trabajo se ocuparon dichos encoders para estimar la posición del robot pues se quiso experimentar con la mayor autonomía posible. En cambio, no se usaron los sonares, pues se conocía la posición del obstáculo a priori, esto redujo en buena parte el código a programar. La Figura 5.2 muestra el robot utilizado.

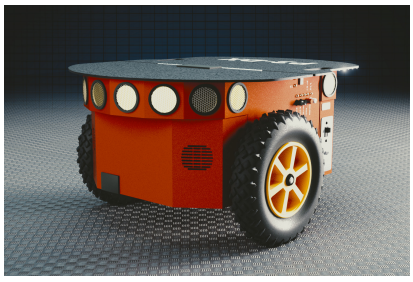


Figura 5.2: Robot Pioneer 3DX.

El software ARCOS se utiliza para configurar el periodo de envío de paquetes en conexión cliente-servidor del robot a 10 milisegundos ya que su configuración de fábrica es para enviar datos cada 100 milisegundos (aunque el periodo de muestreo utilizado fue de 20ms). Además de modificar la velocidad máxima de velocidad traslacional a un máximo de $500mm/s$ la cual viene configurada de fábrica con un máximo de $1200mm/s$. Esta modificación se realiza por seguridad, para evitar que el robot alcance velocidades muy grandes y llegue a dañar algo en el espacio de trabajo, que es reducido.

5.2. Novint Falcon.

El Mando háptico *Novint Falcon*^{TM} es capaz de generar fuerzas que simulan distintos entornos, tales fuerzas permiten transmitir al usuario la sensación del tacto al usuario. Este dispositivo se diseñó principalmente para la industria del entretenimiento, videojuegos o sustitución de periféricos. Pero gracias a su bajo costo, se convirtió en pionero de la categoria de productos hápticos para el mercado de consumo. Esta interfaz háptica permite enviar órdenes para el movimiento en tres dimensiones. A continuación se presentan las especificaciones técnicas:

- Espacio de trabajo tridimensional: 4"x 4"x 4".
- Magnitud de la fuerza que puede producir: mayor a 2 lb.
- Resolución de posición: mayor a 400 dpi.
- Puerto de comunicación: USB 2.0.
- Tamaño: 9"x 9"x 9".
- Peso: 6 lb.
- Entrada al dispositivo: 30V DC, 1.0A.
- Potencia: 30W, 100V - 240V, 50Hz - 60Hz.

También, este mando puede ser programado en sus diversas funciones a través de la llamada capa de abstracción HDAL, con funciones precargadas para ser configuradas según las necesidades. La compañia Novint proporciona esta herramienta por medio del kit de desarrollo de software SDK, el cual se instala en el equipo donde se hace el desarrollo de la aplicación. Se requiere configurar el programa de los parámetros para que pueda encontrar las librerias respectivas. En la figura 5.3 se muestra el mando háptico utilizado.

Figura 5.3: Novint Falcon.

La capa de abstracción HDAL ha sido diseñada para proporcionar un estándar en la programación del Falcon y otros dispositivos hápticos. Permite que el programador

vaya directamente a configurar los aspectos importantes cómo las fuerzas y nos entrega directamente la posicion de la empuñadura en todo momento. Es necesario tener en cuenta un modo fluido y consistente para proveer a la HDAL de la información que se genera a partir de la interacción con el entorno como variables de un lenguaje de programación de alto nivel. La comunicación entre la capa de simulación háptica y las funciones de la HDAL se realiza a través de una función llamada callback invocada desde el interior de la HDAL a una velocidad de mil veces por segundo (servo-tick). Dentro de esta función el usuario lee la posición del efector final del Novint Falcon, calcula los niveles de fuerza y los envía para se aplicados al dispositivo. Para conocer los detalles de la programación del mando háptico por favor véase [5].

5.3. Comunicación inalámbrica.

Para la comunicación inalámbrica se ocupa un Wibox de la marca Lantronix que es un dispositivo que permite conectar dispositivos con conexión serial a redes inalámbricas 802.11b, es útil ya que la comunicación con el robot es por medio de cable serial y con estos dispositivos se puede acceder a una red inalámbrica, la cual es creada por el robot y se pueden enviar y recibir datos de esta manera desde una computadora conectada a esta red.

Figura 5.4: Wibox.

5.4. Cámara a bordo.

Para que el operador humano del mando háptico fuera capaz de ver la trayectoria que seguía el robot móvil, se montó un celular Huawei Y635 en la parte superior del robot a través de un soporte. Mediante la cámara frontal de 2MP se adquirió imagen en tiempo real haciendo videollamada por internet usando el software de Skype.

Figura 5.5: Huawei Y635.

Dimensiones	$72.87 \times 143.97 \times 10.2mm$
Peso	$178g$
SoC	Qualcomm Snapdragon 410 MSM8916
Procesador	ARM Cortex-A53, 1200 MHz, 4 núcleos
Procesador gráfico	Qualcomm Adreno 306, 400 MHz
Memoria RAM	1 GB, 533 MHz
Memoria Interna	4 GB
Pantalla	5 in, TFT, 480 x 854 píxeles, 24 bit
Batería	2000 mAh, Li-ión
Sistema operativo	Android 5.0 Lollipop
Cámara	2592 x 1944 píxeles, 1920 x 1080 píxeles, 30 fps
Wi-Fi	b,g,n, Wi-Fi Hotspot

Tabla 5.1: Especificaciones del Huawei Y635.

Capítulo 6

Resultados experimentales.

En este cápitulo se presentan los resultados de cinco experimentos representativos. Primero se muestra el seguimiento de una trayectoria cerrada usando el control automático. Después se muestra el seguimiento de una trayectoria punto a punto evadiendo un obstáculo fijo usando el método de potenciales artificiales y el control automático. Se continua mostrando los resultados del seguimiento de una trayectoria abierta evadiendo un obstáculo fijo combinando el uso del control automático y la teleoperación. Después se muestra el seguimiento de una trayectoria punto a punto evadiendo un obstáculo fijo combinando el control automático con la teleoperación. Finalmente se muestra el resultado del experimento previamente descrito pero en esta ocasión se evade un obstáculo móvil.

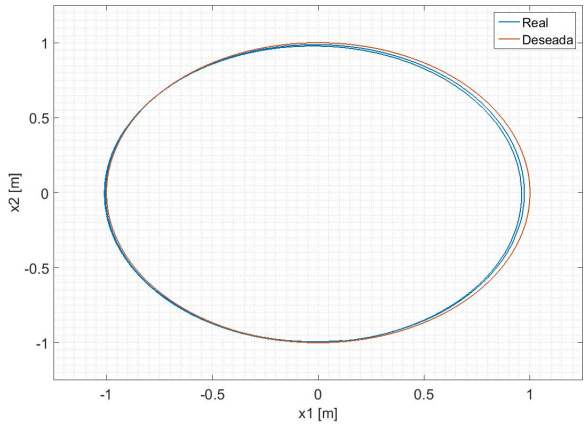
6.1. Experimento 1: Seguimiento de trayectoria cerrada.

El primer experimento consistió en el seguimiento de una trayectoria circular con las coordenadas iniciales del robot iguales a las coordenadas iniciales de la trayectoria deseada, se utilizó un robot virtual para asegurar que la trayectoria deseada fuera realizable y el esquema de control fue automático, es decir no hubo intervención de un usuario humano. Los parámetros utilizados en la primera prueba se muestran en la tabla 6.1, donde $\lambda = \alpha T$.

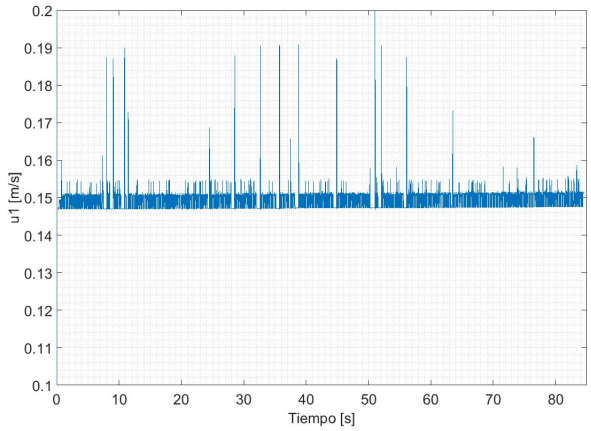
$T\ [s]$	α	$k_{1,-1}$	$k_{1,0}$	$k_{2,-1}$	$k_{2,0}$	$k_{3,-1}$	$k_{3,0}$	K_1	K_2
0.02	1	0.01	0.01	0.01	0.001	0.01	0.01	0.1	0.1
K_3	A_1	A_2	A_3	B_1	B_2	B_3	$u_{1d}\ [m/s]$	$u_{2d}\ [rad/s]$	
0.9	0.1	0.1	0.1	0.01	0.01	0.01	0.15	0.15	

Tabla 6.1: Parámetros para el control, experimento 1

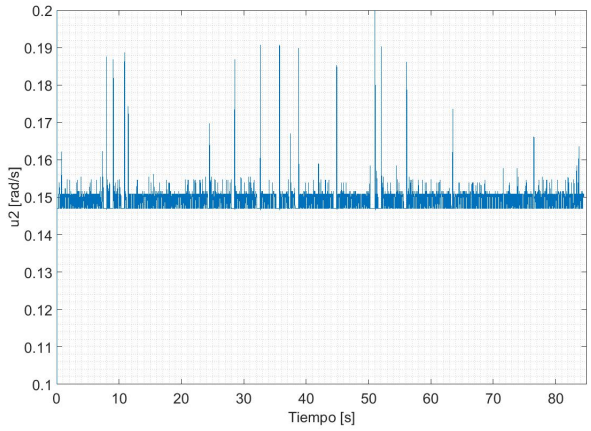
Además, los controles se limitaron a los siguientes valores, $u_1 = \pm 0.3 m/s$, $u_2 = \pm 0.5 rad/s$. Esto se hizo por cuestiones de seguridad y para garantizar la afinidad entre el robot y el modelo que lo representa, pues a bajas velocidades influyen menos



(a) Trayectoria real vs trayectoria deseada del experimento No.1.

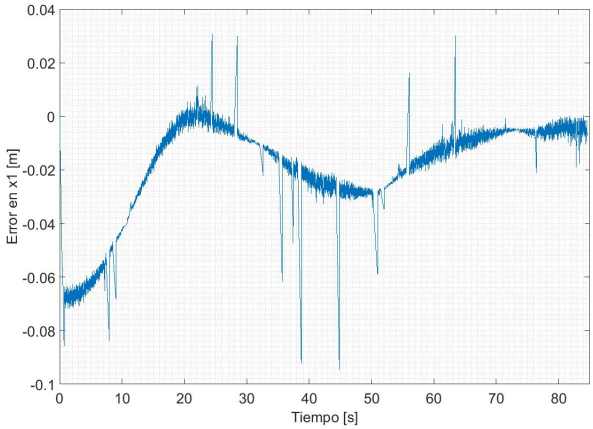


(b) Control u_1.

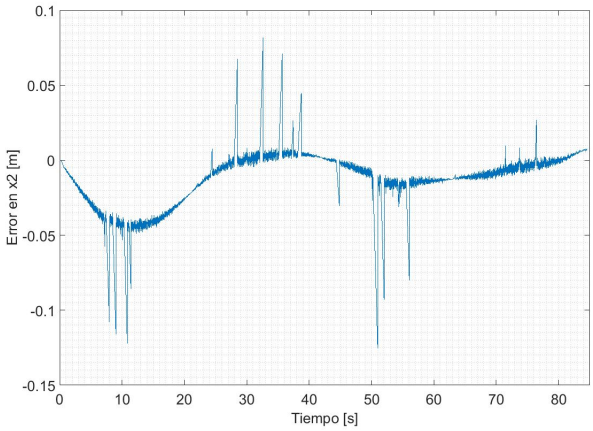


(c) Control u_2.

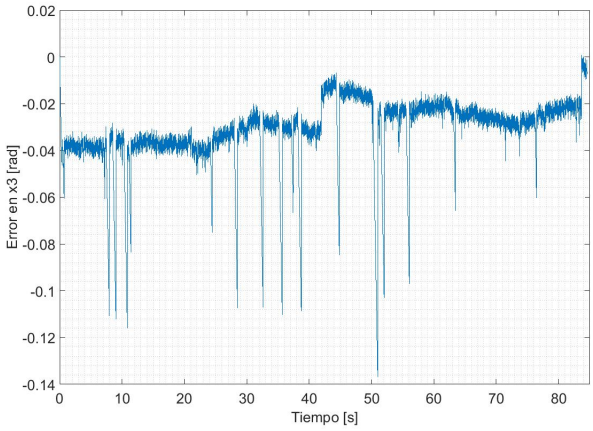
Figura 6.1: Controles del experimento No.1.



(a) Error en x_1.



(b) Error en x_2.



(c) Error en x_3.

Figura 6.2: Errores del experimento No.1.

las fuerzas inerciales. Las condiciones iniciales del robot virtual y del robot físico se muestran en la tabla 6.2.

$x_1(0)$ $[m]$	$x_2(0)$ $[m]$	$x_3(0)$ $[rad]$	$x_{1d}(0)$ $[m]$	$x_{2d}(0)$ $[m]$	$x_{3d}(0)$ $[rad]$
0	0	0	0	0	0

Tabla 6.2: Condiciones iniciales, experimento 1

Con los diferentes experimentos en cuanto al seguimiento de una trayectoria cerrada se pudo ver que las ganancias $k_{1,-1}$, $k_{2,-1}$, $k_{3,-1}$, $k_{1,0}$, $k_{2,0}$, $k_{3,0}$, K_1, K_2, y K_3 debían ser pequeñas, los intervalos entre A_i y B_i que definen la conmutación deben ser lo más pequeños posible. También, entre menor sea el tiempo de muestreo, es mejor el comportamiento del sistema, aunque por limitaciones propias del robot, el valor mínimo de T fue $T = 0.02s$.

En la figura 6.1 se muestra el resultado de la primera prueba, se ve que el error de seguimiento es menor a 10cm y también se muestran los controles, los cuales cabe señalar que son las señales que se generan desde la computadora pero dichas señales pasan después por el controlador PID de cada rueda, entonces la señal que llega a los motores es más suave que la señal mostrada, además se pueden apreciar algunos picos bastante pronunciados, los que al parecer son causados por la pérdida de comunicación entre el robot y la computadora, esto se puede notar en los datos recabados, pues cuando existen dichos picos, la posición del robot permanece igual, es decir como si permaneciera estático o hiciera una pausa, lo que no es cierto en la realidad. La falla en la comunicación es evidente en las gráficas de los errores, pues sería físicamente imposible para este experimento que en un periodo de tiempo exista un error de seguimiento en x_1 igual a $10cm$ y al siguiente periodo de muestreo el error sea aproximadamente cero. Se puede ver también que durante todo el experimento $u_1 \approx 0.15m/s$ y $u_2 \approx 0.15rad/seg$.

En la figura 6.2 se muestran los errores de seguimiento, los cuales se mantienen pequeños durante todo el experimento. En la primera prueba se puede apreciar que el mayor problema se presenta al seguir las variables x_1d y x_2d, mientras que el seguimiento del ángulo se realiza de mejor manera. También se observa que el desempeño del sistema no es muy bueno a medida que se aleja el robot de las condiciones iniciales de la trayectoria deseada (errores de seguimiento apreciables en x_1 y x_2).

6.2. Experimento 2: Seguimiento de trayectoria punto a punto.

El segundo experimento consistió en alcanzar una posición deseada evadiendo un obstáculo fijo, partiendo de una condición inicial alejada de dicha meta, se utilizó el método de potenciales artificiales para generar la trayectoria deseada y para evadir al

obstáculo, el control fue automático sin teleoperación. Los parametros del control se muestran en la tabla 6.3, donde $p_0 = r_0$, las condiciones iniciales del robot se muestran en la tabla 6.4, mientras que la posición del obstáculo y de la meta se encuentran en la tabla 6.5.

$T\ [s]$	α	$k_{1,-1}$	$k_{1,0}$	$k_{2,-1}$	$k_{2,0}$	$k_{3,-1}$	$k_{3,0}$	K_1	K_2	$p_0\ [m]$
0.02	1	0.01	0.01	0.01	0.001	0.01	0.01	0.1	0.1	1
K_3	A_1	A_2	A_3	B_1	B_2	B_3	η	ζ	k_v	k_0
0.1	0.1	0.1	0.1	0.01	0.01	0.01	2	0.1	0.08	0.1

Tabla 6.3: Parámetros para el control, experimento 2

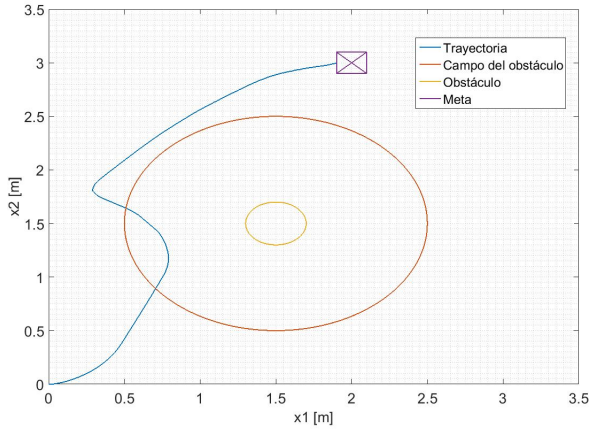
$x_1(0)\ [m]$	$x_2(0)\ [m]$	$x_3(0)\ [rad]$	$x_{1d}(0)\ [m]$	$x_{2d}(0)\ [m]$	$x_{3d}(0)\ [rad]$
0	0	0	0	0	0

Tabla 6.4: Condiciones iniciales, experimento 2

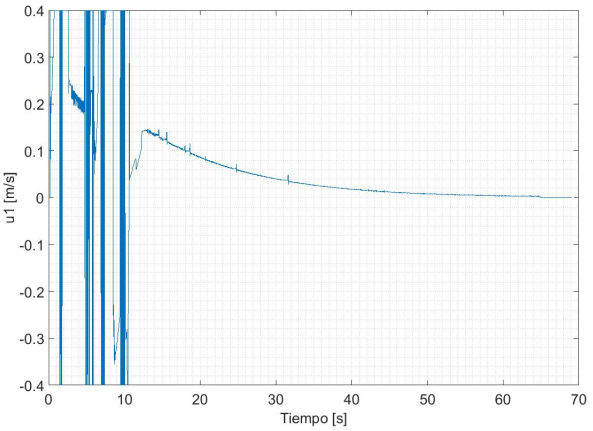
Los controles se limitaron a los valores, $u_1 = \pm 0.4 m/s$ y $u_2 = \pm 0.5 rad/s$. En la figura 6.3 se muestra el resultado de la segunda prueba, Donde con un retardo de $20ms$ el robot alcanza la meta en poco más de un minuto, sin embargo se lograron experimentos exitosos con un retardo de hasta $200ms$. La magnitud del retardo entre los valores mencionados influye directamente con la trayectoria seguida por el robot, así como en la duración del experimento, sin embargo el objetivo principal que es llegar a la meta evitando una colisión, se cumple. Cabe mencionar que cuando no se trabaja con retardos el modelo discreto no presenta ventajas sobre el modelo en tiempo continuo, así que queda a consideración de cada quien cual modelo utilizar. Por otra parte, es evidente que en condiciones reales, el retardo es variable, sin embargo, si se puede medir y asegurar el valor máximo del retardo en la comunicación, es posible a través de una programación adecuada mantener al retardo constante. Es decir, suponiendo que el retardo máximo fuera de $100ms$ y nuestra estrategia de control nos permitiera compensar retardos de hasta $200ms$, entonces podríamos asegurar siempre un periodo de muestreo igual a $100ms$ y un retardo $\alpha = 2$.

Las velocidades son pequeñas pues el robot que se utilizó es bastante pesado, es decir que el modelo cinemático pierde validez a velocidades más altas, lo que provoca también que el control no funcione de la mejor manera.

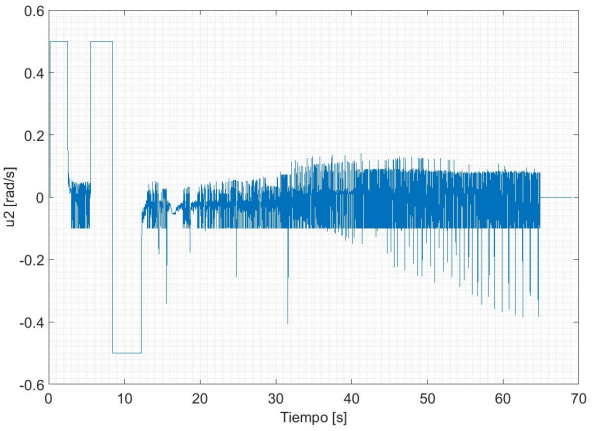
Variando las ganancias que se ocupan en el método de potenciales artificiales se puede modificar el comportamiento del sistema así como la velocidad y la distancia de penetración en el campo del obstáculo. En la figura 6.3 también se muestran los controles calculados en (3.7), es necesario mencionar que el robot tiene su propio controlador interno para evitar que se dañen por error los motores, así que las señales de control



(a) Trayectoria del experimento No.2.

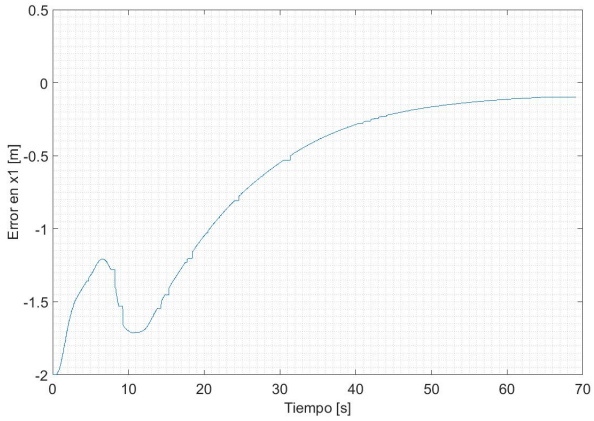


(b) Control u_1.

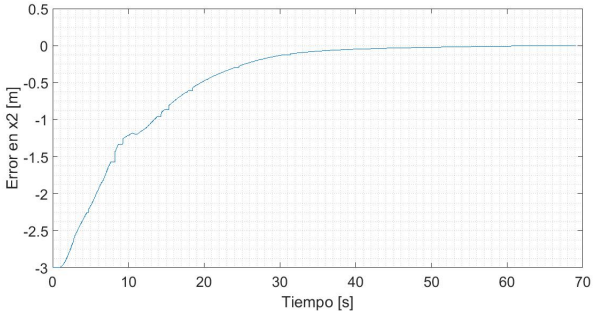


(c) Control u_2.

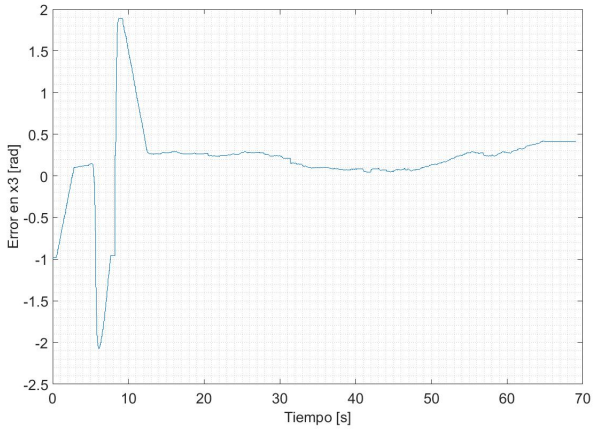
Figura 6.3: Controles del experimento No.2.



(a) Error en x_1.

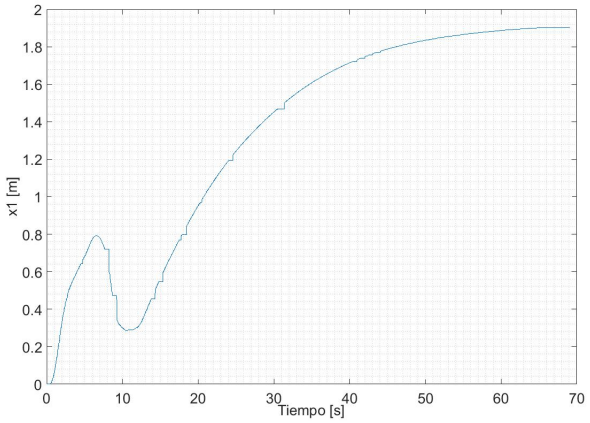


(b) Error en x_2.

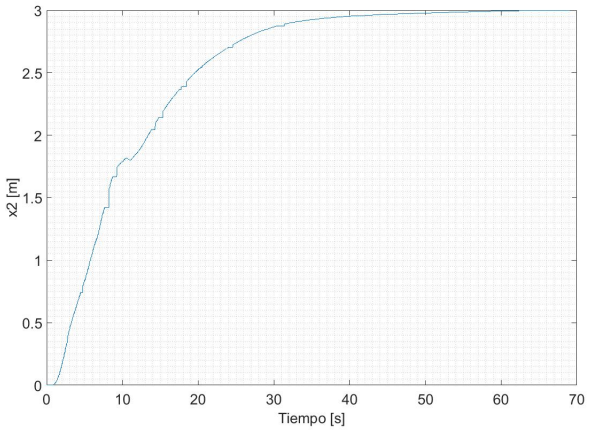


(c) Error en x_3.

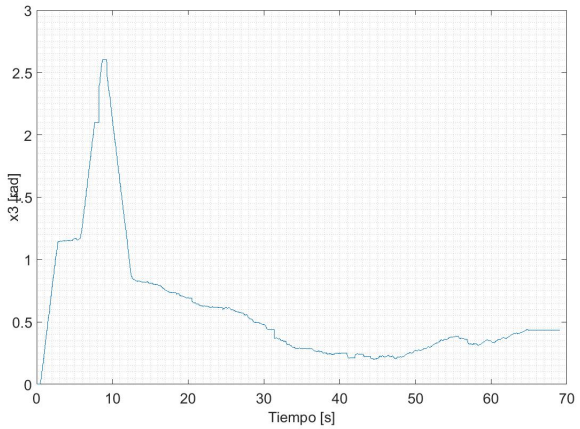
Figura 6.4: Errores del experimento No.2.



(a) Estado x_1.



(b) Estado x_2.



(c) Estado x_3.

Figura 6.5: Estados del experimento No.2.

x_{1obs} $[m]$	x_{2obs} $[m]$	x_{1g} $[m]$	x_{2g} $[m]$
1.5	1.5	2	3

Tabla 6.5: Posiciones del obstáculo y la meta del experimento No.2.

que llegan a las ruedas no conmutan tan bruscamente.

En la figura 6.4 se muestran los errores de seguimiento, los cuales carecen de relevancia en este experimento, pues lo que se busca es llegar a la meta sin importar la trayectoria seguida o el ángulo de llegada. Por otra parte, para esta segunda prueba se puede apreciar un buen comportamiento del robot, el cual alcanza la meta, evadiendo al obstáculo que se le presenta. Cabe mencionar que cuando el robot se encuentra a una distancia menor a 10cm de la meta, el control se apaga, esto para evitar condiciones excesivas en el algoritmo de control discontinuo.

6.3. Experimento 3: Seguimiento de trayectoria abierta con teleoperación.

En este experimento se sigue una trayectoria generada por un robot virtual. Además durante la trayectoria se lleva a cabo la evasión de un obstáculo fijo. Cuando el robot móvil (2,0) se acerca a una distancia p_0 del obstáculo el control cambia de automático a manual, operado por un usuario a través del mando háptico. El usuario es capaz de obtener imagen en tiempo real del recorrido del robot gracias a la cámara del celular que está montado en el robot y que envía la señal a una pantalla que el operador humano puede ver.

Los parámetros del control se muestran en la tabla 6.6, las condiciones iniciales en la tabla 6.7 y la posición del obstáculo fue en $x_{1obs} = 1m$ y $x_{2obs} = 1.25m$. Para generar la trayectoria se utilizaron las siguientes velocidades deseadas $u_{1d} = 0.1m/s$ y $u_{2d} = 0.15 - tan^{-1}(0.05t)0.18$. En la figura 6.6 se observa la trayectoria que siguió el robot, se puede ver que la commutación a la trayectoria deseada después de salir del campo del obstáculo es lenta. Esto también depende de la velocidad, posición y orientación con la que se deje el campo del obstáculo. En este experimento se ocupó un retardo de $20ms$ pero se llevaron a cabo experimentos usando hasta un retardo de $60ms$, sin embargo la commutación hacia la trayectoria deseada era aún más lenta.

En la figura 6.6 también se muestran los controles enviados al robot, se aprecia claramente el cambio entre control automático y el control manual generado por la teleoperación. Mientras que en la figura 6.7 se muestran los errores de seguimiento, se puede apreciar que los errores se mantienen bajos antes de entrar al campo del obstáculo, y una vez que se deja dicho campo tienden a disminuir nuevamente. Cabe resaltar que éste experimento es el caso extremo pues el obstáculo está posicionado

$T\ [s]$	α	$k_{1,-1}$	$k_{1,0}$	$k_{2,-1}$	$k_{2,0}$	$k_{3,-1}$	$k_{3,0}$	K_1	K_2	$p_0\ [m]$	k_{xmin}
0.02	1	$1x10^{-5}$	$1x10^{-5}$	$1x10^{-5}$	$1x10^{-5}$	0.02	$1x10^{-5}$	$1x10^{-4}$	$1x10^{-4}$	0.95	1
K_3	A_1	A_2	A_3	B_1	B_2	B_3	k_{xmax}	γ_x	c_1	c_2	c_3
5.5	0.1	0.1	0.1	0.01	0.01	0.01	1250	200	25	25	25

Tabla 6.6: Parámetros para el control, experimento 3

$x_1(0)\ [m]$	$x_2(0)\ [m]$	$x_3(0)\ [rad]$	$x_{1d}(0)\ [m]$	$x_{2d}(0)\ [m]$	$x_{3d}(0)\ [rad]$
0	0	0	0	0	0

Tabla 6.7: Condiciones iniciales, experimento 3

justamente sobre la trayectoria deseada.

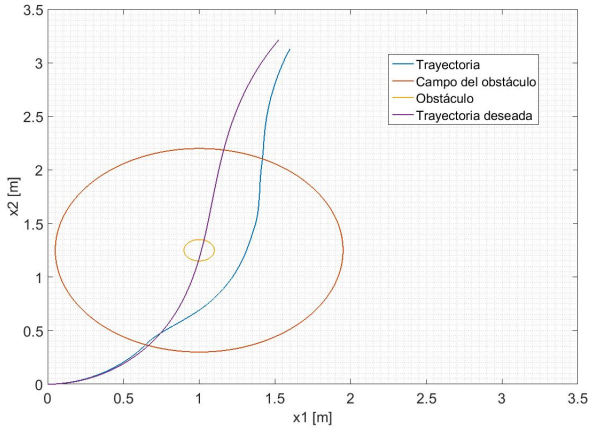
6.4. Experimento 4: Seguimiento de trayectoria punto a punto con teleoperación.

En éste experimento el robot parte de una posición inicial y llega a una posición final x_g deseada, la meta. La generación de la trayectoria fue hecha usando el método de potenciales artificiales, durante el seguimiento de la trayectoria se evade a un obstáculo fijo. Cuando el robot se encuentra lo suficientemente cerca del obstáculo, se cambia del control automático a manual, operado por un usuario a través de un mando háptico. El usuario es capaz de obtener imagen en tiempo real del recorrido del robot gracias a la cámara del celular que está montado en el robot y que envía la señal a una pantalla que el operador humano puede ver.

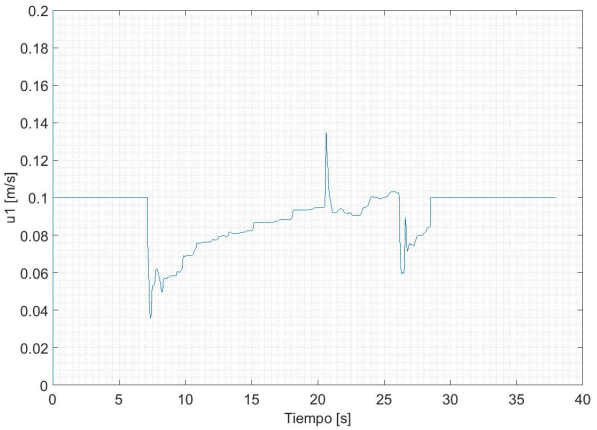
$T\ [s]$	α	$k_{1,-1}$	$k_{1,0}$	$k_{2,-1}$	$k_{2,0}$	$k_{3,-1}$	$k_{3,0}$	K_1	K_2	$p_0\ [m]$	k_{xmin}
0.02	3	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.95	1
K_3	A_1	A_2	A_3	B_1	B_2	B_3	k_{xmax}	γ_x	c_1	c_2	c_3
0.01	0.1	0.1	0.1	0.01	0.01	0.01	1250	200	25	25	25
$K_{1,1}$	$k_{1,2}$	$k_{2,1}$	$k_{2,2}$	$k_{3,1}$	$k_{3,2}$						
0.1	0.1	0.1	0.1	0.1	0.1						

Tabla 6.8: Parámetros para el control, experimento 4

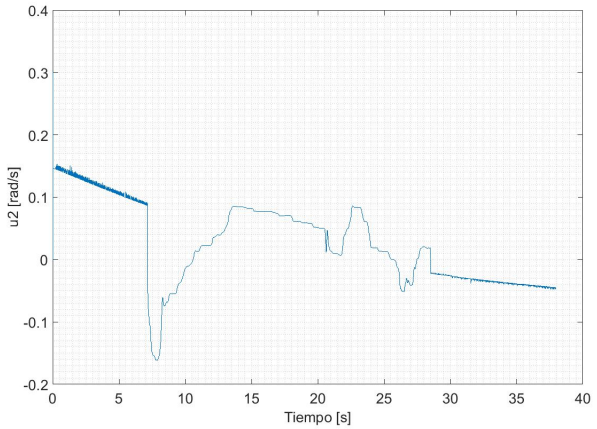
En la tabla 6.8 se muestran los parámetros de control usados para este experimento, se puede notar que existen más ganancias que en experimentos anteriores, esto es porque el retardo es mayor, por lo tanto, en nuestro esquema de control necesitamos más ganancias. En la tabla 6.9 se muestran las condiciones iniciales, mientras que en la tabla 6.10 se observan las posiciones tanto de la meta como del obstáculo.



(a) Trayectoria real vs trayectoria deseada del experimento No.3.

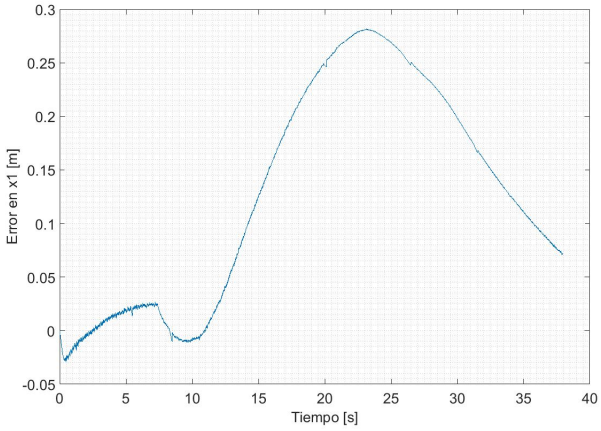


(b) Control u_1.

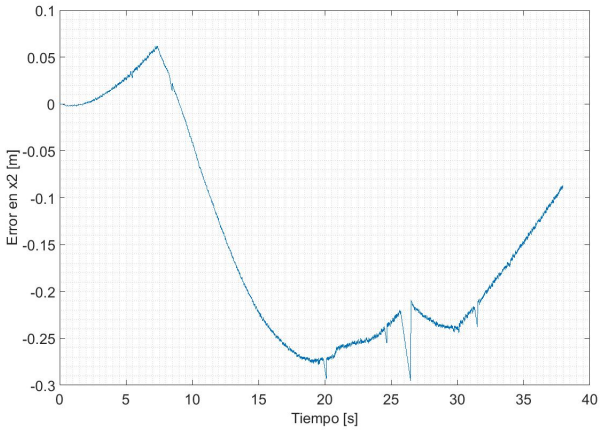


(c) Control u_2.

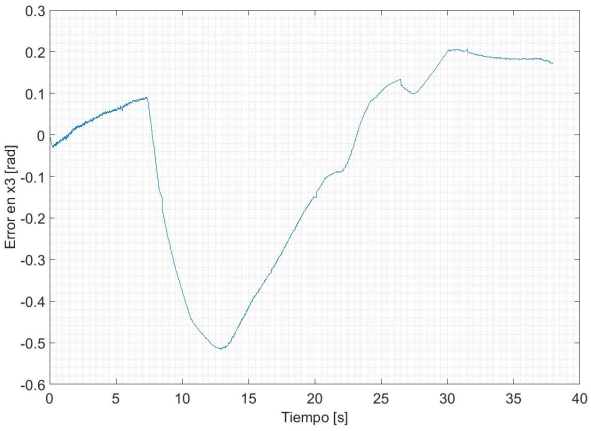
Figura 6.6: Controles del experimento No.3.



(a) Error en x_1.

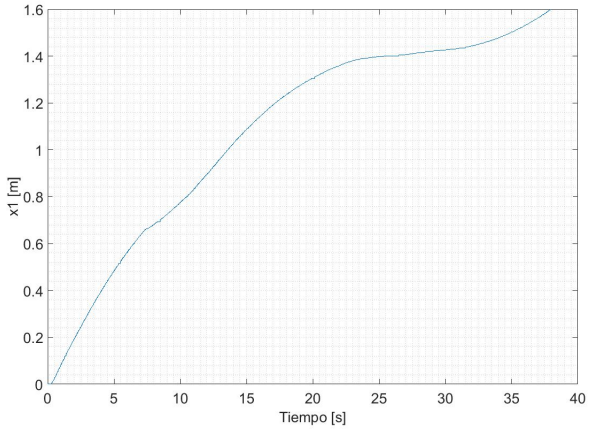


(b) Error en x_2.

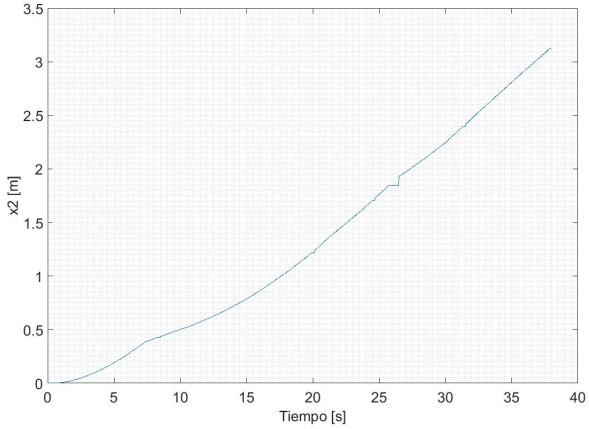


(c) Error en x_3.

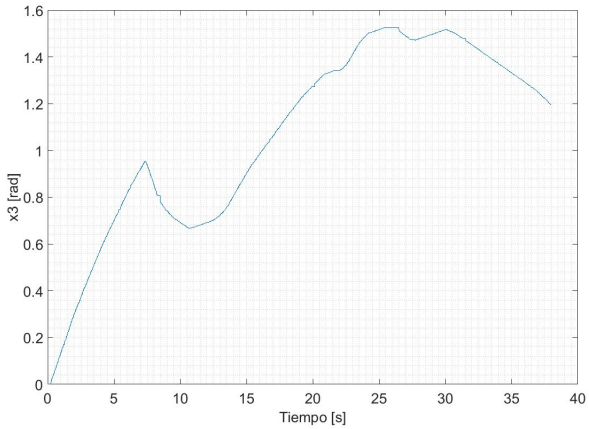
Figura 6.7: Errores del experimento No.3.



(a) Estado x_1.



(b) Estado x_2.



(c) Estado x_3.

Figura 6.8: Estados del experimento No.3.

En la figura 6.9 se observa la trayectoria del robot, se aprecia que llega a la meta, dicha tarea la realiza en menos de 2 minutos. Existen varios factores que incluyen la trayectoria descrita por el robot, pues cuando se encuentra en modo manual, es decir que algún usuario humano lo está teleoperando, hay una infinidad de posibles trayectorias y velocidades. Sin embargo se observo que si el ángulo del robot se encuentra orientado hacia la meta y las velocidades cuando se deja el campo son pequeñas, el desempeño del control es mejor.

$x_1(0)$ $[m]$	$x_2(0)$ $[m]$	$x_3(0)$ $[rad]$	$x_{1d}(0)$ $[m]$	$x_{2d}(0)$ $[m]$	$x_{3d}(0)$ $[rad]$
0	0	0	0	0	0

Tabla 6.9: Condiciones iniciales, experimento 4

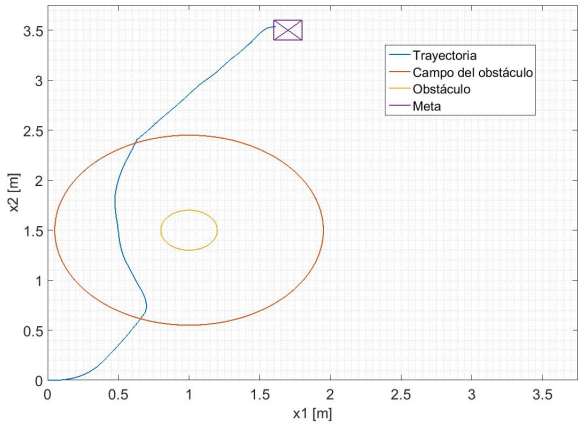
x_{1obs} $[m]$	x_{2obs} $[m]$	x_{1g} $[m]$	x_{2g} $[m]$
1	1.5	1.7	3.5

Tabla 6.10: Posiciones del obstáculo y la meta del experimento No.4.

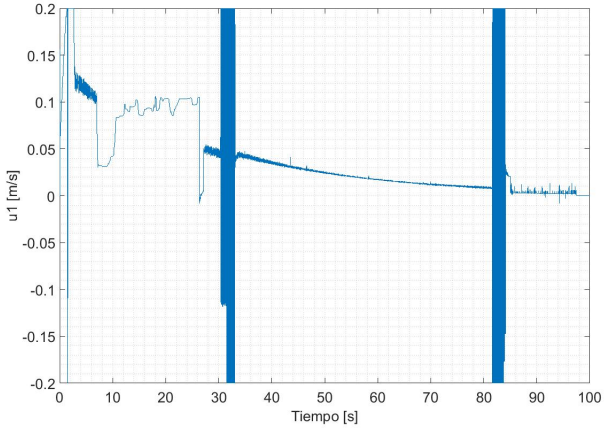
En la figura 6.9 también se muestran los contoles enviados al robot, se puede apreciar que existe una señal suave en cada una de las gráficas, esto sucede cuando el robot se encuentra dentro del campo del obstáculo, pues se cambia del control automático (esquema de control discontinuo) a la teleoperación (esquema de control continuo). Hay que recordar que las señales que llegan a los motores son más suaves a las que se muestran porque pasan a través de un controlador PID que tiene el robot. En la figura 6.10 se muestran los errores de seguimiento.

6.5. Experimento 5: Seguimiento de trayectoria punto a punto con teleoperación y obstáculo móvil.

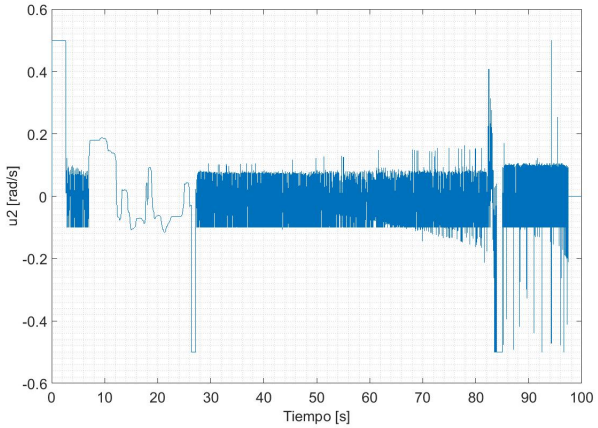
En éste experimento el robot parte de una posición inicial y llega a una posición final x_g deseada, la meta. La generación de la trayectoria fue hecha usando el método de potenciales artificiales, durante el seguimiento de la trayectoria se evade a un obstáculo móvil. Cuando el robot se encuentra lo suficientemente cerca del obstáculo, se cambia del control automático a manual, operado por un usuario a través de un mando háptico. El usuario es capaz de obtener imagen en tiempo real del recorrido del robot gracias a la cámara del celular que está montado en el robot y que envía la señal a una pantalla que el operador humano puede ver. Este experimento se llevo a cabo principalmente para explorar las posibilades a futuro con este esquema de control.



(a) Trayectoria del experimento No.4.

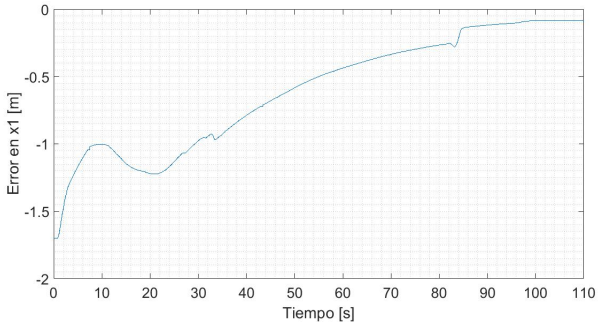


(b) Control u_1.

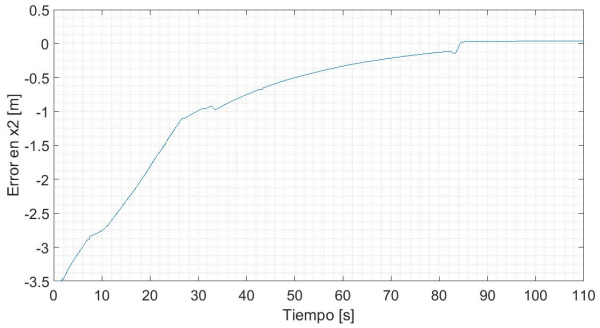


(c) Control u_2.

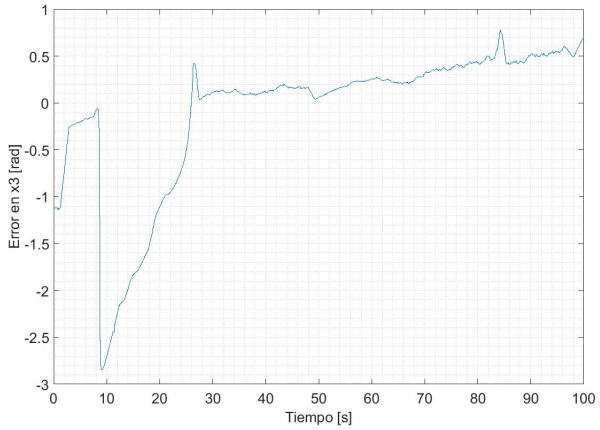
Figura 6.9: Controles del experimento No.4.



(a) Error en x_1.

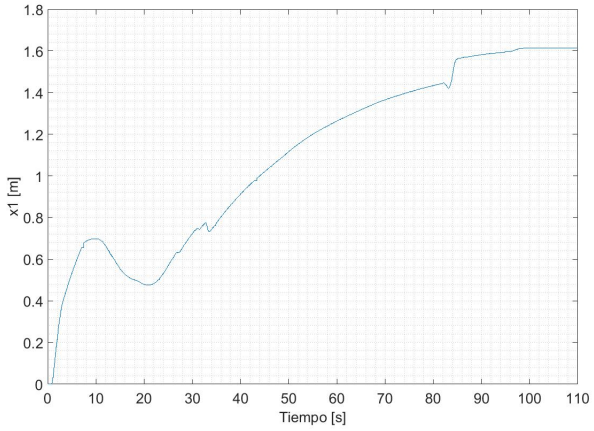


(b) Error en x_2.

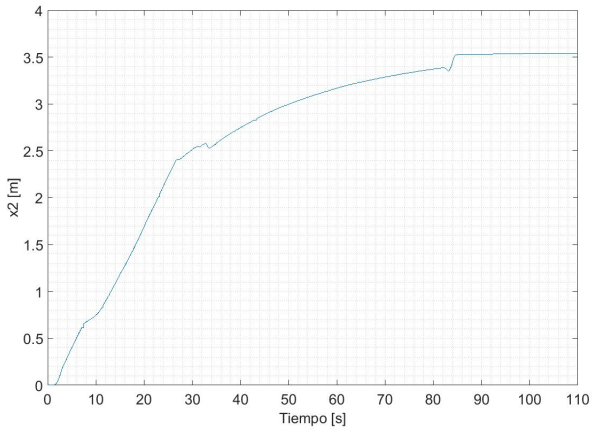


(c) Error en x_3.

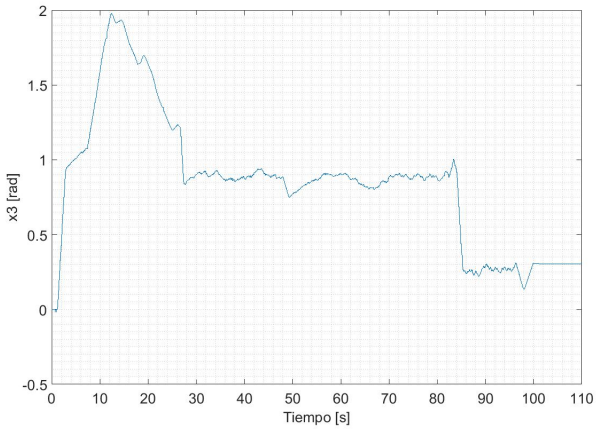
Figura 6.10: Errores del experimento No.4.



(a) Estado x_1.



(b) Estado x_2.

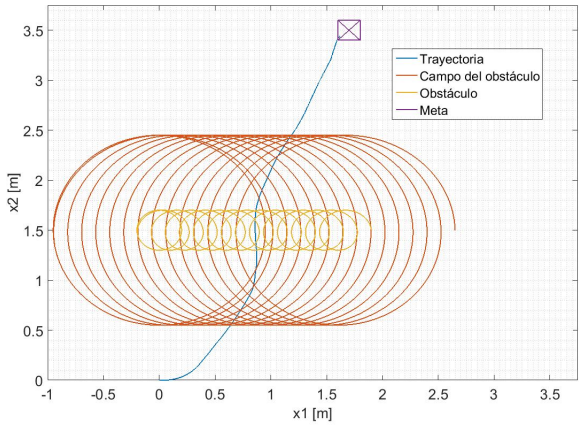


(c) Estado x_3.

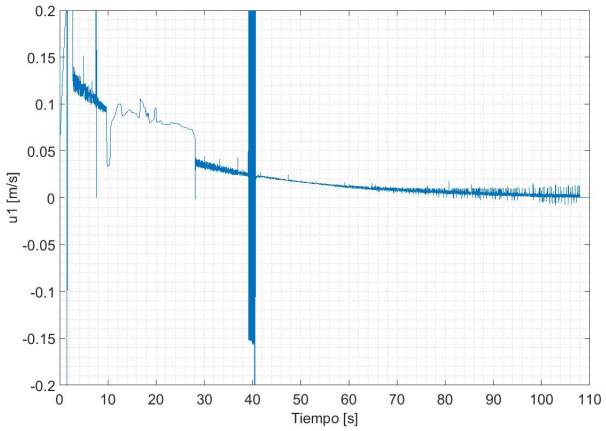
Figura 6.11: Estados del experimento No.4.

Los parámetros de control, así como las condiciones iniciales son los mismos que en el experimento anterior, por lo que se pueden revisar las tablas 6.8 y 6.9. El cambio más significativo es que en este experimento el obstáculo, que es otro robot Pioneer 3DX se encuentra en movimiento, su posición fue de $x_{1obs} = 1.7 - 0.05t$ y $x_{2obs} = 1.5$ para $t \geq 0$.

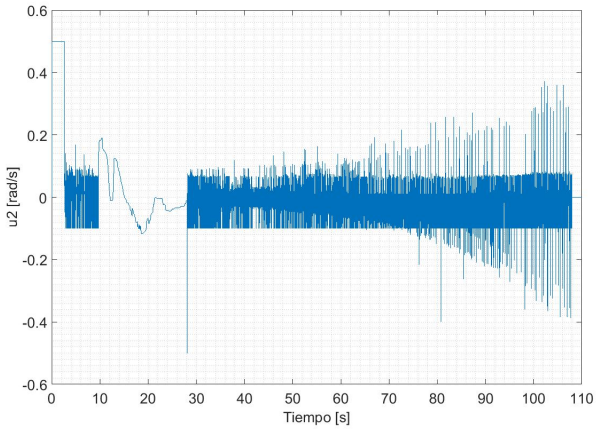
En la figura 6.12 se muestra la trayectoria descrita por el robot y la trayectoria del obstáculo móvil. En este experimento se hicieron algunos ajustar para tratar de evitar la colisión entre los robots, la más significativa fue la de mantener una velocidad bastante pequeña del obstáculo, además se desplazó en linea recta, estas dos caracteristicas hicieron que el usuario fuera capaz de predecir los movimientos necesarios para evitar una colisión. El tiempo para alcanzar la meta fue poco menos de dos minutos.
En la figura 6.12 también se muestran los controles enviados al robot. Se puede apreciar que el tiempo de teleoperación es pequeño en comparación con la duración del experimento. Esto se debe a que el obstáculo y el robot interactúan por un periodo breve. En la figura 6.13 se muestran los errores de seguimiento que como en casos anteriores carecen de relevancia.



(a) Trayectoria del experimento No.5.

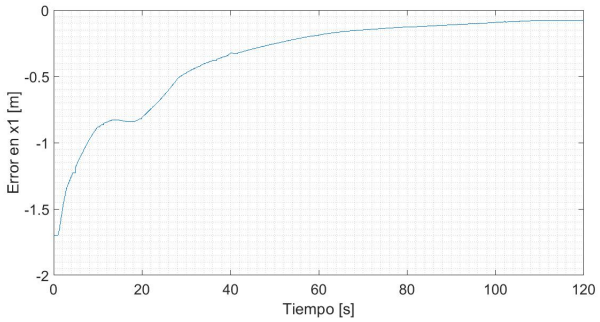


(b) Control u_1.

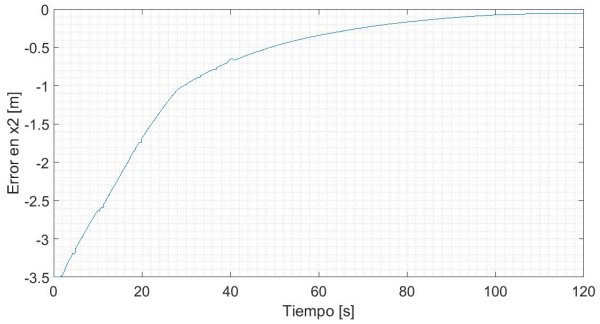


(c) Control u_2.

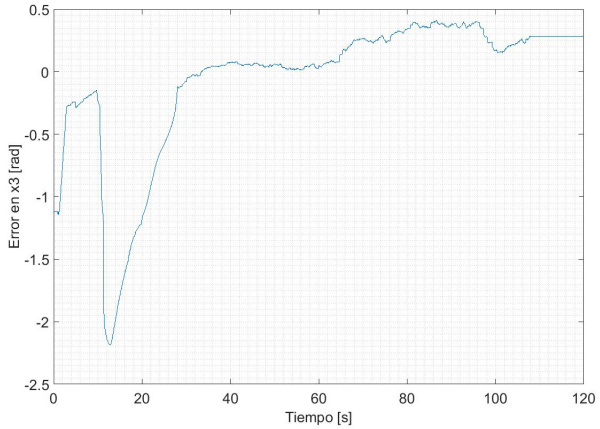
Figura 6.12: Controles del experimento No.5.



(a) Error en x_1.

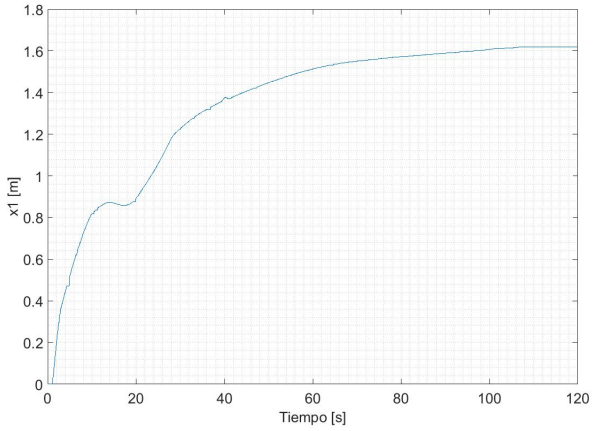


(b) Error en x_2.

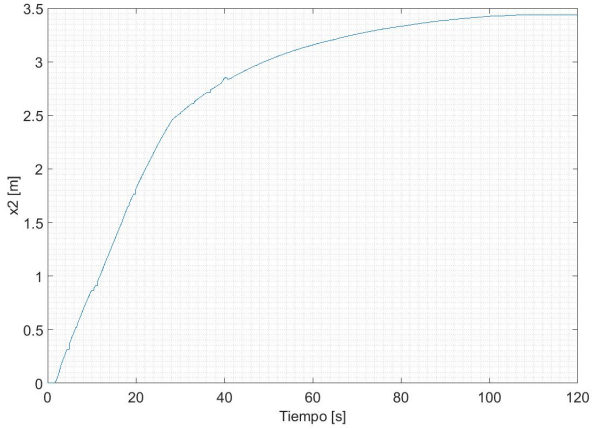


(c) Error en x_3.

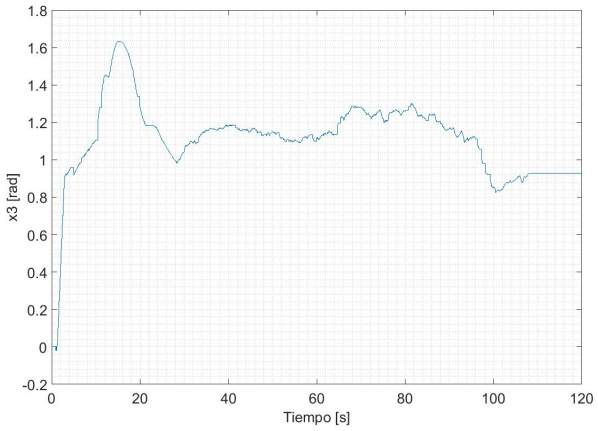
Figura 6.13: Errores del experimento No.5.



(a) Estado x_1.



(b) Estado x_2.



(c) Estado x_3.

Figura 6.14: Estados del experimento No.5.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones.

- El esquema de control automático fue capaz de seguir una trayectoria cerrada (generada por un robot virtual) manteniendo los errores de seguimiento cercanos a cero con un retardo en las señales de control inducido de $20ms$. Sin embargo, cuando se aumenta el retardo o cuando las condiciones iniciales del robot están alejadas de la trayectoria deseada, los errores de seguimiento aumentan también. Se puede decir que bajo ciertas condiciones las leyes de control propuestas para este caso son aceptables.

- Por otra parte, la estrategia de control automático para el seguimiento punto a punto, es decir, usando el método de potenciales artificiales fue capaz de cumplir con su objetivo de alcanzar una meta evadiendo un obstáculo fijo, en el experimento mostrado se trabajó con un retardo en las señales de control de $20ms$ pero hubo experimentos en los que se pudo compensar um retardo de hasta $200ms$ sin que el desempeño bajara demasiado. Por lo tanto se puede decir, que para este caso el control utilizado funciona satisfactoriamente.

- Para el caso de seguimiento de una trayectoria abierta (generada por un robot virtual) evadiendo un obstáculo fijo, es decir cuando se combina el control automático con la teleoperación. La estrategia de control automático permite seguir la trayectoria cuando el robot se encuentra fuera del campo de repulsión del obstáculo, manteniendo bajos los errores de seguimiento. Sin embargo cuando se deja el campo del obstáculo, la convergencia a la trayectoria deseada es lenta. Dicha convergencia depende de varios factores, como lo son las velocidades, la posición y la orientación del robot al dejar el campo de repulsión del obstáculo. En el experimento mostrado se trabajó con un retardo inducido de $20ms$ en las señales de control pero se llegó a trabajar con un retardo de hasta $60ms$. Sin embargo como sucedió en casos anteriores, cuando el retardo es mayor o las condiciones iniciales del robot están alejadas de la trayectoria, el desempeño disminuye de manera considerable.

- En cuanto al seguimiento de trayectoria punto a punto evadiendo un obstáculo fijo usando el método de potenciales combinando el control automático con la teleoperación. Se puede decir que el desempeño fue satisfactorio. Puesto que se alcanzó la meta y se evadió al obstáculo sin dificultad alguna. Se compensó un retardo de $60ms$ y se llegó a trabajar con un retardo de hasta $80ms$ sin que hubiera un cambio significativo en el rendimiento del esquema de control.
- Hablando puntualmente del esquema de teleoperación. Resultó muy fácil poder teleoperar al robot y evitar cualquier obstáculo cuando se puede ver el camino a seguir, es decir, cuando es posible ver a través de una cámara la trayectoria del robot. Las fuerzas generadas por el mando háptico nos dan una idea de la proximidad al obstáculo pero es necesario tener conocimiento previo del funcionamiento de la estrategia de teleoperación. Sin embargo, cuando sólo se usa la teleoperación para alcanzar una meta, la estrategia no es adecuada, pues no guía al usuario de manera adecuada. También, cuando no se tiene imagen de la trayectoria del robot, la teleoperación resulta inutil.
- Finalmente, la plataforma experimental resultó bastante práctica y útil, sin embargo hubieron algunos problemas significativos. Por ejemplo, el robot Pioneer 3DX muchas veces presentaba errores en la comunicación con la computadora por lo que introducia retardos que no estaban contemplados, además en algunas ocasiones no sólo era el retardo, sino que había perdida de información. Por otra parte, el mando háptico Novint Falcon tiene un periodo de trabajo ideal entre $1ms$ y $10ms$, sin embargo en este trabajo por limitaciones técnicas el periodo de muestreo fue de $20ms$ lo que dificulto el uso del mando háptico, porque algunas veces las fuerzas generadas por dicho aparato no eran tan suaves o continuas.

7.2. Trabajo futuro.

- Utilizar para la experimentación robots más ligeros, con un mejor desempeño para el envío y recepción de datos, que trabajen a menos de $10ms$.
- Encontrar una mejor manera de aquirir imagen en tiempo real.
- Encontrar la mejor manera de estimar la posición del robot.
- Seguir experimentando para encontrar las ganancias óptimas según el caso en el que se esté trabajando.
- Continuar con la experimentación para el caso de la evasión de un obstáculo móvil.
- Encontrar una manera práctica de estimar la posición de los obstáculos.
- Experimentar con retardos reales variables y discontinuos para encontrar la manera de trabajar con ellos y minimizar su efecto negativo en el desempeño de la estrategia de control.

Bibliografía

[1] A. Trebi-Ollennu, C. Leger, E.T. Baumgartner, and R.G. Bonitz. Robotic arm in-situ operations for the mars exploration rovers mission. *Systems, Man and Cybernetics, 2005 IEEE International Conference*, pages 1799–1806, 2005.

[2] P.A. Niño-Suarez, E. Aranda-Bricaire, and M. Velasco-Villa. Discrete-time sliding mode path-tracking control for a wheeled mobile robot. *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3052–3057, 2006.

[3] O. Martínez-Zúñiga, M. Velasco-Villa, and R. Castro-Linares. Control de seguimiento práctico de un robot móvil tipo (2,0) en tiempo discreto. *Congreso Nacional de Control Automático 2013*, pages 610–615, 2013.

[4] M. Velasco-Villa, A. Álvarez Aguirre, and G. Rivera-Zago. Discrete-time control of an omnidireccional mobile robot subject to transport delay. *Proceedings of the 2007 American Control Conference*, pages 2171–2176, 2007.

[5] M. Arteaga-Escamilla, R. Castro-Linares, and J. Álvarez Gallegos. Teleoperación de un robot móvil con evasión de obstáculos. *Congreso Nacional de Control Automático, AMCA 2015*, pages 588–593, 2015.

[6] H. Sira-Ramírez and R. Castro-Linares. Sliding mode rest-to-rest stabilization and trajectory tracking for a discretized flexible joint manipulator. *Dynamics and Control*, pages 87–105, 2000.

[7] Ildar Farkhatdinov and Jee-Hwan Ryu. Hybrid position-position and position-speed command strategy for the bilateral teleoperation of a mobile robot. *International Conference on Control, Automation and Systems 2007*, pages 2442–2447, 2007.

[8] Sajad Salmanipour and Shahin Sirouspour. Teleoperation of a mobile robot with model-predictive obstacle avoidance control. *39th Annual Conference of the IEEE Industrial Electronics Society*, pages 4270–4275, 2013.

[9] Ildar Farkhatdinov, Jee-Hwan Ryu, and Jinung An. A preliminary experimental study on haptic teleoperation of mobile robot with variable force feedback gain. *IEEE Haptics Symposium 2010*, pages 251–256, 2010.

[10] C. Canudas, B. Siciliano, G. Bastin, B. Brogliato, G. Campion, B. D'Andrea-Novel, A.D Luca, W. Khalil, R. Lozano, R. Ortega, C. Samson, and P. Tomei. *Theory of Robot Control*. Springer London, 1996.

[11] Ildar Farkhatdinov, Jee-Hwan Ryu, and Jury Poduraev. A user study of command strategies for mobile robot teleoperation. *Intel Serv Robotics*, pages 95–104, 2009.

[12] M. Velasco-Villa, E. Aranda-Bricaire, and R. Orosco-Guerrero. Discrete-time modeling and path tracking for a wheelled mobile robot. *Computación y Sistemas*, pages 142–160, 2009.

[13] F. Rosales-Hernández, M. Velasco-Villa, R. Castro-Linares, B. Del Muro-Cuellar, and M. A. Hernández-Pérez. Discrete-time synchronization strategy for input time-delay mobile robots. *Journal of the Franklin Institute*, pages 2911–2935, 2013.

[14] Oscar Martínez Zúñiga. Sincronización de robots móviles tipo (2,0) en tiempo discreto. *Tesis de Maestría, CINVESTAV-Unidad Zacatenco*, 2013.

[15] Francisco Rosales Hernández. Sincronización de robots móviles sujetos a retardo de tiempo. *Tesis de Maestría, CINVESTAV-Unidad Zacatenco*, 2012.

Apéndice A

Artículos Publicados

- H. Mendoza Hernández, R. Castro Linares y J. Álvarez Gallegos *Seguimiento de Trayectoria para un Robot Móvil (2,0) con retardos en la entrada Usando un Esquema de Control Discontinuo*. Congreso Nacional de Control Automático 2017, llevado a cabo del 4 al 6 de Octubre del 2017 en Nuevo León, México.

Apéndice B

Código para control automático más teleoperación

```
//TELEOPERACIÓN CON MODELO DE GANANCIA VARIABLE CUANDO SE ESTÁ DENTRO DEL CAMPO DEL OBSTÁCULO
//Y MODO AUTÓNOMO CUANDO SE ENCUENTRA FUERA DE DICHO CAMPO

#include "Aria.h"
#include <windows.h>
#include <iostream>
#include <fstream>
#include <time.h>
#include <dos.h>
#include <stdlib.h>
#include <string>
#include <conio.h>
#include <sys/types.h>
#include <fcntl.h>
#include <errno.h>
#include <stdio.h>
#include <math.h>
#include "haptics.h"
#include <hdl/hdl.h>
#include <hdlu/hdlu.h>
double cursorPosWC[3];
double x1=0;
double xp;
double yp;
double zp;
HapticsClass gHaptics;
double valor=0;
using namespace std;
//PARA ODOMETRÍA
//para amortiguador falcon
double xpm1=0;
double ypm1=0;
double zpm1=0;
double dxp=0;
double dyp=0;
```

```
double dzp=0;
//PARA AMORTIGUADOR ROBOT
double xom1=0;
double yom1=0;
double dxo=0;
double dyo=0;
//variables a seguir
double xps=0;
double dxps=0;
double xpsm1=0;
//PARA MODELO DE GANANCIA CONSTANTE
double delta=0;
double xobs=1;
double yobs=1.5;
//EXTRA PARA MODELO DE GANCIA VARIABLE
double kx=0;
double kminx=1;
double kmaxx=1250;
double alphax=200;
//PARA MOPDELO CON DISTANCIA
double distobs=0;
//PARA CALCULAR LA VELOCIDAD DE APROXIMAMIENTO AL OBSTACULO
double distobs1=0;
double distobs2=0;
double distobs3=0;
double distobs4=0;
double distobs5=0;
double distobs6=0;
double distobs7=0;
double distobs8=0;
double distobs9=0;
double distobs10=0;
double distobs11=0;
double distobs12=0;
double distobs13=0;
double distobs14=0;
double distobs15=0;
double distobs16=0;
double distobs17=0;
double distobs18=0;
double distobs19=0;
double distobs20=0;
double drobs;
//PARA SABER SI SE ACERCA O SE ALEJA EL ROBOT DEL OBSTACULO
double sig=0;
double sig2=0;
//NUEVAS VARTIABLES PARA MODELO CON ANGULO DE APROXIMACIÓN
double distmeta=0;
double beta=0;
double xmeta=1.7;
double ymeta=3.5;
double distxobs=0;
double distyobs=0;
double incrementod1=0;
double thd1=0;
```

```
double thd01=0;
double incrementocd1=0;
double thr21=0;
double tha21=0;
//PARA GRAFICAR LOS OBSTÁCULOS
double xpobs=0;
double ypobs=0;
double y1pobs=0;
double xpmeta=0;
double ypmeta=0;
double p03=0.1;
double distxmeta=0;
double distymeta=0;
//VARIABLES DE RP
//ESTADOS ADELANTADOS
double x1a=0.2;
double x2a=-2;
double x3a=0.2;
double x1am=0;
double x2am=0;
double x3am=0;
double x1r=0;
double x2r=0;
double x3r=0;
double u1=0;
double u2=0;
double u11=0;
double u22=0;
double T=0.02;
double tiempo=0;
int i=1;
double f1=0;
double f2=0;
double FT=0;
double dg=0;
double E=0.1;
double Vd=0;
double x1g=1.7;
double x2g=3.5;
double y1d=0;
double y2d=0;
double y3d=0;
double x1d=0;
double x2d=0;
double x3d=0;
double x1dd=0;
double x2dd=0;
double x3dd=0;
double k1=0.01; //0.1
double k2=0.01;  //0.1
double k3=0.01;   //0.8
//NUEVAS CONSTANTES PARA INTEGRADOR
double k4=0.01;   //0.01
double k5=0.01;   //0.01
double k6=0.01;   //0.01
```

```
double v1=0;
double v2=0;
double v3=0;
double incremento=0;
double incrementoc=0;
double thr=0;
double tha=0;
double th11=0;
double th22=0;
double gamma=0;
double psi=0;
double thaux=0;
double dx=0;
double dy=0;
double thd=0;
double incrementod=0;
double thd0=0;
double incrementocd=0;
double thr2=0;
double tha2=0;
double thi=0;
double u222=0;
double T2=0;
double th1;
//AÑADIDO PARA EVASIÓN
double f1r=0;
double f2r=0;
//IMPORTANTE
double eta=2;
double k0=0.1;
//IMPORTANTE
double p0=0.75;
double p02=0.2;
double p=0;
double x1obs=1;
double x2obs=1.5;
double Vd2=0;
double invp=0;
double invp0=0;
double p3=0;
double invp3=0;
//POSICIÓN DEL OBSTACULO
double x1pobs=0;
double x2pobs=0;
double x11pobs=0;
double x22pobs=0;
//CONTROL DISCONTINUO
double sigma1k0=0;
double sigma1k0abs=0;
double sigma1k1=0;
//NUEVO
double errx1kmenos2=0;
double errx1kmenos1=0;
double errx1k0=0;
double k11=0.99;
```

```
double signo1=0;
//NUEVO PARA FUNCIÓN SIGNO SUAVE
double signo11=0;
double A11=0.1;
double A1=0.2;
double B1=0.01;
double K1=0.01;
double sigma2k0=0;
double sigma2k0abs=0;
double sigma2k1=0;
//NUEVO
double errx2kmenos2=0;
double errx2kmenos1=0;
double errx2k0=0;
double k22=0.99;
double signo2=0;
//NUEVO PARA FUNCIÓN SIGNO SUAVE
double signo22=0;
double A22=0.1;
double A2=0.2;
double B2=0.01;
double K2=0.01;
double sigma3k0=0;
double sigma3k0abs=0;
double sigma3k1=0;
//NUEVO
double errx3kmenos2=0;
double errx3kmenos1=0;
double errx3k0=0;
double k33=0.99;
double signo3=0;
//NUEVO PARA FUNCIÓN SIGNO SUAVE
double signo33=0;
double A33=0.2;
double A3=0.1;
double B3=0.01;
double K3=0.1;
//NUEVAS VARIABLES PARA RETARDO
double u1n[500000];
double u1r[500000];
double u2n[500000];
double u2r[500000];
double u11r[500000];
double u22r[500000];
//NÚMERO DE RETARDOS
int nr=1;
//NUEVA VELOCIDAD MINIMA
double u1c=100;
//PARA ODOMETRÍA
double xo=0;
double yo=0;
double tho=0;
double yaw3=0;
//PARA CONTROL CORREGIDO
double k7=0.01;   //0.01
```

```
double k8=0.01;  //0.01
double k9=0.01;  //0.01
double errx1kmas2=0;
double errx2kmas2=0;
double errx3kmas2=0;
double errx1kmas1=0;
double errx2kmas1=0;
double errx3kmas1=0;
double x1dmas2=0;
double x2dmas2=0;
double x3dmas2=0;
//VARIABLES DE CONTROL PARA MODELO HÍBRIDO
double sc1=0;
double sc2=0;
int main(int argc, char **argv)
{
        /* Conexión a Robot */
        Aria::init();
        ArArgumentBuilder arg_nut(20);
        ArArgumentParser parser_nut(&arg_nut);
        ArRobot robot_nut;
        arg_nut.add("-rh 10.0.126.12");

        ArRobotConnector robotConnector_nut(&parser_nut, &robot_nut);
        if(!robotConnector_nut.connectRobot())
        {
                ArLog::log(ArLog::Terse, "simpleConnect:Could not connect to the robot.");
                if(parser_nut.checkHelpAndWarnUnparsed())
                {
                        Aria::logOptions();
                        Aria::exit(1);
                }
        }
        if (!Aria::parseArgs() || !parser_nut.checkHelpAndWarnUnparsed())
        {
                Aria::logOptions();
                Aria::exit(1);
        }
        ArLog::log(ArLog::Normal, "simpleConnect: Connected to robot");
        robot_nut.runAsync(true);
        ofstream rtraydata("datos.txt");
        robot_nut.lock();
        robot_nut.enableMotors();
        robot_nut.unlock();
        robot_nut.setVel(0);
        robot_nut.setRotVel(0);
        //INICIA FALCON
        time_t tstart=0;
        // Set up the scene with graphics and haptics. Call the haptics initialization function
        gHaptics.init();
        Sleep(200);
        while( i<7500){
        tiempo=i*T-T;
                xo=robot_nut.getX()*0.001;
                yo=robot_nut.getY()*0.001;
```

```
tho=ArMath::degToRad(robot_nut.getTh());
x1r=xo;
x2r=yo;
yaw3=tho;
if (yaw3<0){
        th1=tho+2*3.141592;
}
else{
}
incremento=yaw3-th11;
incrementoc=incremento*incremento;
th11=yaw3;
if (incrementoc>38.44){
        thr=tha;
}
else{
        thr=tha+incremento;
        tha=thr;
}
x3r=thr;
//DISTANCIA A LA META
distmeta=sqrt((x1g-x1r)*(x1g-x1r)+(x2g-x2r)*(x2g-x2r));
//ÁNGULO DE APROXIMAMIENTO AL OBSTACULO
distxobs=x1obs-x1r;
distyobs=x2obs-x2r;
thd1=atan2(distyobs,distxobs);
if (thd1<0){
        thd1=tho+2*3.141592;
}
else{
}
incrementod1=thd1-thd01;
incrementocd1=incrementod1*incrementod1;
thd01=thd1;
if (incrementocd1>2.2){
        thr21=tha21;
}
else{
        thr21=tha21+incrementod1;
        tha21=thr21;
}
beta=thr-thr21;
//PARA GRAFICAR LA POSICIÓN DEL OBSTÁCULO
x1pobs=p0*cos(tiempo)+x1obs;
x2pobs=p0*sin(tiempo)+x2obs;
x11pobs=p02*cos(tiempo)+x1obs;
x22pobs=p02*sin(tiempo)+x2obs;
xpmeta=p03*cos(tiempo)+x1g;
ypmeta=p03*sin(tiempo)+x2g;
//SIMULACION DE GANANCIA LINEAL
//MODIFICADO PARA FUNCIONAR CON DISTANCIA MEDIDA AL OBSTACULO EN DOS COORDE
distobs=sqrt((x1obs-x1r)*(x1obs-x1r)+(x2obs-x2r)*(x2obs-x2r));
distmeta=sqrt((x1g-x1r)*(x1g-x1r)+(x2g-x2r)*(x2g-x2r));
distxmeta=x1g-x1r;
distymeta=x2g-x2r;
```

```
drobs=1000*(distobs-distobs20)*(0.05/T);
distobs20=distobs19;
distobs19=distobs18;
distobs18=distobs17;
distobs17=distobs16;
distobs16=distobs15;
distobs15=distobs14;
distobs14=distobs13;
distobs13=distobs12;
distobs12=distobs11;
distobs11=distobs10;
distobs10=distobs9;
distobs9=distobs8;
distobs8=distobs7;
distobs7=distobs6;
distobs6=distobs5;
distobs5=distobs4;
distobs4=distobs3;
distobs3=distobs2;
distobs2=distobs1;
distobs1=distobs;

if (distobs>distobs20){
        sig2=-1;
}
else{
        sig2=1;
}
if (xp<0){
        if (xp>-0.3){
                xp=-0.3;
        }
        else{
        }
}
else{
        if (xp<0.3){
                xp=0.3;
        }
        else{
        }
}
dxo=sig2*sqrt(xp*xp)*150;
if (dxo>0){
        if (xp>0){
                sig=1;
        }
        else{
                sig=-1;
        }
}
else{
        if (xp<0){
                        sig=1;
        }
```

```
        else{
                sig=-1;
        }
}
//xom1=xo;
if (dxo<0){
        kx=kminx;
}
else{
        if (dxo<alphax){
                kx=(1/alphax)*(kmaxx-kminx)*dxo+kminx;
        }
        else{
                kx=kmaxx;
        }
}
//MODELO HIBRIDO
if (distobs<0.05){
        distobs=0.05;
}
else{
}
if (distmeta<0.05){
        distmeta=0.05;
}
else{
}
if (distobs<p0){

        xps=(kx*cos(beta))/(distobs);
}
else{
        xps=1/distmeta;
}

dyo=(yo-yom1)/T;
yom1=yo;
//xps=0.1;
//  dxps=xps-xpsm1;
//xpsm1=xps;
gHaptics.init2(x1);
printf("Posiciones \n");
//printf("dX=%.2lf\tX=%.2lf\n",dxo,xo);
//printf("distxmeta=%.4lf\tdistymeta=%.4lf\tdistmeta=%.4lf\n",distxmeta,dist
//robot_nut.lock();
//robot_nut.enableMotors();
//robot_nut.unlock();
//VELOCIDADES FALCON
dxp=xp-xpm1;
xpm1=xp;
dyp=yp-ypm1;
ypm1=yp;
dzp=zp-zpm1;
zpm1=zp;
p=sqrt((x1r-x1obs)*(x1r-x1obs)+(x2r-x2obs)*(x2r-x2obs));
```

```
invp=(1/p);
p3=p*p*p;
invp3=(1/p3);
invp0=(1/p0);
if (p<p0){
        f1r=eta*(invp-invp0)*invp3*(x1r-x1obs);
        f2r=eta*(invp-invp0)*invp3*(x2r-x2obs);
        //f1r=0;
        //f2r=0;
}
else{
        f1r=0;
        f2r=0;
}
//FUERZAS
f1=-E*(x1r-x1g)+f1r;
f2=-E*(x2r-x2g)+f2r;
FT=sqrt(f1*f1+f2*f2);
//cout<<"FT "<<FT<<endl;
//DISTANCIA A LA META
dg=sqrt((x1r-x1g)*(x1r-x1g)+(x2r-x2g)*(x2r-x2g));
/*//VELOCIDAD DESEADA
Vd=0.03*dg;*/
//VELOCIDAD DESEADA
Vd=0.03*(dg-k0*invp);
//PLANTA DESEADA
y1d=x1d+T*Vd*(f1/FT);
x1dmas2=x1d+2*T*Vd*(f1/FT);
x1dd=x1d;
//NUEVO
errx1kmenos2=errx1kmenos1;
errx1kmenos1=errx1k0;
errx1k0=x1r-x1d;
errx1kmas1=y1d-x1a;
errx1kmas2=x1dmas2-x1am;
sigma1k0=errx1kmas1-k7*errx1k0-k1*errx1kmenos1-k4*errx1kmenos2;
if (sigma1k0<0){
        signo1=-1;
        sigma1k0abs=-1*sigma1k0;
}
else{
        if (sigma1k0>0){
                signo1=1;
                sigma1k0abs=sigma1k0;
        }
        else{
                signo1=0;
                sigma1k0abs=sigma1k0;
        }
}
//NUEVO PARA FUNCION SIGNO SUAVE signo11
if (sigma1k0<-A11){
        signo11=signo1;
}
else{
```

```
        if (sigma1k0>A11){
                signo11=signo1;
        }
        else{
                signo11=(signo1*sigma1k0)/A11;
        }
}
//CAMBIADO A SIGNO11
if (sigma1k0abs>A1){
        sigma1k1=K1*signo11;
}
else{
        if (sigma1k0abs>B1){
                sigma1k1=(K1/(A1-B1))*(sigma1k0abs-B1)*signo11;
        }
        else{
                sigma1k1=0;
        }
}
//NUEVO
v1=x1dmas2+k7*errx1kmas1+sigma1k1+k1*errx1k0+k4*errx1kmenos1;
//v1=y1d+sigma1k1+k1*(x1r-x1d);
//v1=y1d+k1*(x1r-x1d);
x1d=y1d;
dx=(y1d-x1dd)/T;
y2d=x2d+T*Vd*(f2/FT);
x2dmas2=x2d+2*T*Vd*(f2/FT);
x2dd=x2d;
//NUEVO
errx2kmenos2=errx1kmenos1;
errx2kmenos1=errx2k0;
errx2k0=x2r-x2d;
errx2kmas1=y2d-x2a;
errx2kmas2=x2dmas2-x2am;
sigma2k0=errx2kmas1-k8*errx2k0-k2*errx2kmenos1-k5*errx2kmenos2;
if (sigma2k0<0){
        signo2=-1;
        sigma2k0abs=-1*sigma2k0;
}
else{
        if (sigma2k0>0){
                signo2=1;
                sigma2k0abs=sigma2k0;
        }
        else{
                signo2=0;
                sigma2k0abs=sigma2k0;
        }
}
//NUEVO PARA FUNCION SIGNO SUAVE signo22
if (sigma2k0<-A22){
        signo22=signo2;
}
else{
        if (sigma2k0>A22){
```

```
                signo22=signo2;
        }
        else{
                signo22=(signo2*sigma2k0)/A22;
        }
}
//CAMBIADO A SIGNO22
if (sigma2k0abs>A2){
        sigma2k1=K2*signo22;
}
else{
        if (sigma2k0abs>B2){
                sigma2k1=(K2/(A2-B2))*(sigma2k0abs-B2)*signo22;
        }
        else{
                sigma2k1=0;
        }
}
//NUEVO
v2=x2dmas2+k8*errx2kmas1+sigma2k1+k2*errx2k0+k5*errx2kmenos1;
//v2=y2d+sigma2k1+k2*(x2r-x2d);
//v2=y2d+k2*(x2r-x2d);
x2d=y2d;
dy=(y2d-x2dd)/T;
thd=atan2(dy,dx);
if (thd<0){
        thd=atan2(dy,dx)+2*3.141592;
}
else{
}
incrementod=thd-thd0;
incrementocd=incrementod*incrementod;
thd0=thd;
if (incrementocd>2.2){
        thr2=tha2;
}
else{
        thr2=tha2+incrementod;
        tha2=thr2;
}
thaux=thr2+T*thr2;
x3dmas2=thr2+2*T*thr2;
x3dd=thr2;
//NUEVO
errx3kmenos2=errx3kmenos1;
errx3kmenos1=errx3k0;
errx3k0=thr-x3d;
errx3kmas1=thaux-x3a;
errx3kmas2=x3dmas2-x3am;
sigma3k0=errx3kmas1-k9*errx3k0-k3*errx3kmenos1-k6*errx3kmenos2;
if (sigma3k0<0){
        signo3=-1;
        sigma3k0abs=-1*sigma3k0;
}
else{
```

```
        if (sigma3k0>0){
                signo3=1;
                sigma3k0abs=sigma3k0;
        }
        else{
                signo3=0;
                sigma3k0abs=sigma3k0;
        }
}
//NUEVO PARA FUNCION SIGNO SUAVE signo33
if (sigma3k0<-A33){
        signo33=signo3;
}
else{
        if (sigma2k0>A22){
                signo33=signo3;
        }
        else{
                signo33=(signo3*sigma3k0)/A33;
        }
}
//CAMBIADO A SIGNO33
if (sigma3k0abs>A3){
        sigma3k1=K3*signo33;
}
else{
        if (sigma3k0abs>B3){
                sigma3k1=(K3/(A3-B3))*(sigma3k0abs-B3)*signo33;
        }
        else{

                sigma3k1=0;
        }
}
//NUEVO
//v3=thaux+k3*(x3-thr2);//thaux
v3=x3dmas2+k9*errx3kmas1+sigma3k1+k3*errx3k0+k6*+errx3kmenos1;
//v3=thaux+sigma3k1+k3*(thr-thr2);
//v3=thaux+k3*(thr-thr2);//thaux
//Planta real "CONTINUA"
if (i<=1){
        x1a=x1r;
        x2a=x2r;
        x3a=thr;
}
else{
}
//NUEVAS
u2=(1/T)*(v3-x3a);
if (u2>0.5){
        u2=0.5;
}
else{
}
if (u2<-0.5){
```

```
            u2=-0.5;
}
else{
}
if (u2<0){
            if (u2>-0.01){
                        u2=-0.1;
            }
            else{
            }
}
else{
            if (u2<0.01){
                        u2=0.01;
            }
            else{
            }
}

gamma=x3a+(T/2)*u2;
psi=(sin(T*u2/2))/(T*u2/2);
//Control u1
u1=(1/(psi*T))*((v1-x1a)*cos(gamma)+(v2-x2a)*sin(gamma));
u1c=u1*u1;
if (u1>0.2){
            u1=0.2;
}
else{
}
if (u1<-0.2){
            u1=-0.2;
}
else{
}
if (dg<0.1){
            u1=0;
            u2=0;
}
else{
}
x1pobs=p0*cos(tiempo)+x1obs;
x2pobs=p0*sin(tiempo)+x2obs;
x11pobs=p02*cos(tiempo)+x1obs;
x22pobs=p02*sin(tiempo)+x2obs;
//ESTADOS ADELANTADOS
x1am=x1a+T*u1*psi*cos(gamma);
x1a=x1am;
x2am=x2a+T*u1*psi*sin(gamma);
x2a=x2am;
x3am=x3a+T*u2;
x3a=x3am;
u1n[i]=u1;
u2n[i]=u2;
if (i<=nr){
            u1r[i]=0;
```

```
                u2r[i]=0;
            }
            else{
                u1r[i]=u1n[i-nr];
                u2r[i]=u2n[i-nr];
            }
            //AGREGADO PARA GFRAFICAR CONTROLES CON RETARDO
            u11r[i]=u1r[i]*1000;
            u22r[i]=u2r[i]*57.3;
            u22=u2*52.7;
            u11=u1*1000;
            printf("Posición del robot x = %f, y = %f \n",xo,yo);
            //NUEVAS VARIABLES DE CONTROL
            if (distobs<p0){
                sc1=xp*50;
                sc2=yp*10;
            }
            else{
                sc1=u11r[i];
                sc2=u22r[i];

            }
            rtraydata<<tiempo<<" "<<x1r<<" "<<x2r<<" "<<x3r<<" "
"<<x1pobs<<" "<<x2pobs<<" "<<x11pobs<<" "<<x22pobs<<";"<<x1a<<" "<<x2a<<" "
"<<x3a<<" "<<x1dd<<" "<<x2dd<<" "<<x3dd<<" "<<u11<<" "<<u22<<" "
"<<u11r[i]<<" "<<u22r[i]<<" "<<f1<<" "<<f2<<" "<<xps<<" "<<sc1<<" "
"<<sc2<<endl;
            // rtraydata<<tiempo<<" "<<xo<<" "<<yo<<" "<<thr2<<" "
"<<xpobs<<" "<<ypobs<<" "<<x1pobs<<" "<<y1pobs<<" "<<xpmeta<<" "<<ypmeta<<"
"<<beta<<" "<<thr21<<" "<<distmeta<<endl;
            //robot_nut.setRotVel(yp*25);
            //robot_nut.setRotVel(0);
            //robot_nut.setVel(xp*150);
            robot_nut.setVel(sc1);
            robot_nut.setRotVel(sc2);
            //robot_nut.setVel(0);
            cout<<i<<endl;
            i++;
            Sleep(T*1000);
        }
        /* Desconexión del robot */
        ArLog::log(ArLog::Normal, "simpleConnect: Ending robot thread...");
        robot_nut.stopRunning();
        // wait for the thread to stop
        robot_nut.waitForRunExit();
        // exit
        ArLog::log(ArLog::Normal, "simpleConnect: Exiting.");
        Aria::exit(0);
        //ADIOS FALCON
        printf("Estas en el codigo principal \n");
        //valor=3;
        //gHaptics.setForce(valor);     // Esta operacion manda un valor a la clase gHaptic
        // dicha variable en cualquier parte del subprograma que controla al joystick
        /*do{
            //Esperar a que pase el tiempo correpondiente
```

```
        }while(clock()-tstart <8000);*/
        //gHaptics.setForce(6);
        // Es posible obtener la posicion del efector y mostrarlas en pantalla con la instruccion
        // gHaptics.getPosition(cursorPosWC);
        //getchar();
        gHaptics.uninit();
        Sleep(2000);
        return 0;
}
```

Apéndice C

Código para el Falcon

```
#include <stdio.h>
#include "haptics.h"
#include <windows.h>
#include <math.h>

// Valores auxiliares
double kq1=1, kf1=4;
extern double xp;
extern double yp;
extern double zp;
extern double dxp;
extern double dyp;
extern double dzp;
//variables a seguir

extern double xps;
extern double dxps;

extern double kx;
extern double sig;
extern double sig2;
// Continuous servo callback function
HDLServoOpExitCode ContactCB(void* pUserData)
{
printf("Estas en ContactCB \n");
Sleep(1000);
// Get pointer to haptics object
HapticsClass* haptics = static_cast< HapticsClass* >( pUserData );

// Get current state of haptic device
hdlToolPosition(haptics->m_positionServo);
hdlToolButton(&(haptics->m_buttonServo));

// Call the function that does the heavy duty calculations.
haptics->feedForce();

// Send forces to device
hdlSetToolForce(haptics->m_forceServo);
// Make sure to continue processing
```

```
//haptics->uninit();

//return HDL_SERVOOP_EXIT;

return HDL_SERVOOP_CONTINUE;
}

// On-demand synchronization callback function
HDLServoOpExitCode GetStateCB(void* pUserData)
{
// printf("Estas en GetSateCB \n");
//Sleep(100);
// Get pointer to haptics object
HapticsClass* haptics = static_cast< HapticsClass* >( pUserData );

// Call the function that copies data between servo side
// and client side
haptics->synch();

// Get current state of haptic device
hdlToolPosition(haptics->m_positionServo);
hdlToolButton(&(haptics->m_buttonServo));


// Call the function that does the heavy duty calculations.
//haptics->getPosition(double pos[3]);

haptics->feedForce();

hdlSetToolForce(haptics->m_forceServo);

// Only do this once.  The application will decide when it
// wants to do it again, and call CreateServoOp with
// bBlocking = true
return HDL_SERVOOP_EXIT;
}

// Constructor--just make sure needed variables are initialized.
HapticsClass::HapticsClass()
: m_deviceHandle(HDL_INVALID_HANDLE),
m_servoOp(HDL_INVALID_HANDLE),
m_ref1(0),
m_inited(false)

{
for (int i = 0; i < 3; i++)
m_positionServo[i] = 0;
}

// Destructor--make sure devices are uninited.
HapticsClass::~HapticsClass()
{
uninit();
}
```

```
double HapticsClass::init2(double x1)
{
//printf("Eres un cabron \n");
//Sleep(100);
m_servoOp = hdlCreateServoOp(GetStateCB, this, bNonBlocking);
// Get pointer to haptics object
//HapticsClass* haptics = static_cast< HapticsClass* >( pUserData );

// Get current state of haptic device
//hdlToolPosition(haptics->m_positionServo);
//hdlToolButton(&(haptics->m_buttonServo));
//xp=x1;

xp=m_positionApp[0];
yp=m_positionApp[1];
zp=m_positionApp[2];
//m_forceServo[0] = 0;
//m_forceServo[2] = 0;
//x1=xp;
//printf("Esta es la primera \n");
//printf("X=%.2lf\n",m_positionApp[0]);
return(xp);

}
void HapticsClass::init()
{
printf("Estas en init \n");
Sleep(100);

HDLError err = HDL_NO_ERROR;

// Passing "DEFAULT" or 0 initializes the default device based on the
// [DEFAULT] section of HDAL.INI.   The names of other sections of HDAL.INI
// could be passed instead, allowing run-time control of different devices
// or the same device with different parameters.  See HDAL.INI for details.
m_deviceHandle = hdlInitNamedDevice("DEFAULT");
testHDLError("hdlInitDevice");

if (m_deviceHandle == HDL_INVALID_HANDLE)
{
printf("No es posible abrir el dispositivo\nPresione enter para terminar"); getchar();
exit(0);
}

// Now that the device is fully initialized, start the servo thread.
// Failing to do this will result in a non-funtional haptics application.
hdlStart();
testHDLError("hdlStart");

// Set up callback function
m_servoOp = hdlCreateServoOp(GetStateCB, this, bNonBlocking);
// m_servoOp = hdlCreateServoOp(ContactCB, this, bNonBlocking);
if (m_servoOp == HDL_INVALID_HANDLE)
{
printf("Invalido servo op handle, falla en el dispositivo\nPresione enter para terminar");
```

```
}
testHDLError("hdlCreateServoOp");

// Make the device current.  All subsequent calls will
// be directed towards the current device.
hdlMakeCurrent(m_deviceHandle);
testHDLError("hdlMakeCurrent");

// Get the extents of the device workspace.
// Used to create the mapping between device and application coordinates.
// Returned dimensions in the array are minx, miny, minz, maxx, maxy, maxz
//                                      left, bottom, far, right, top, near)
// Right-handed coordinates:
//   left-right is the x-axis, right is greater than left
//   bottom-top is the y-axis, top is greater than bottom
//   near-far is the z-axis, near is greater than far
// workspace center is (0,0,0)
hdlDeviceWorkspace(m_workspaceDims);
testHDLError("hdlDeviceWorkspace");

// Establish the transformation from device space to app space
// To keep things simple, we will define the app space units as
// inches, and set the workspace to approximate the physical
// workspace of the Falcon.  That is, a 4" cube centered on the
// origin.  Note the Z axis values; this has the effect of
// moving the origin of world coordinates toward the base of the
// unit.
double gameWorkspace[] = {0,-2,-2,2,2,2};
bool useUniformScale = true;
hdluGenerateHapticToAppWorkspaceTransform(m_workspaceDims,
gameWorkspace,
useUniformScale,
m_transformMat);
testHDLError("hdluGenerateHapticToAppWorkspaceTransform");

m_inited = true;
}

// uninit() undoes the setup in reverse order.  Note the setting of
// handles.  This prevents a problem if uninit() is called
// more than once.
void HapticsClass::uninit()
{
// printf("Estas en uninit \n");
//Sleep(100);
if (m_servoOp != HDL_INVALID_HANDLE)
{
hdlDestroyServoOp(m_servoOp);
m_servoOp = HDL_INVALID_HANDLE;
}
hdlStop();
if (m_deviceHandle != HDL_INVALID_HANDLE)
{
hdlUninitDevice(m_deviceHandle);
m_deviceHandle = HDL_INVALID_HANDLE;
```

```
}
m_inited = false;
}

// This is a simple function for testing error returns.  A production
// application would need to be more sophisticated than this.
void HapticsClass::testHDLError(const char* str)
{
//printf("Estas en testHDLError \n");
//Sleep(100);
HDLError err = hdlGetError();
if (err != HDL_NO_ERROR)
{
printf("HDAL ERROR\nPresione enter para terminar\n"); getchar();
abort();
}
}

// This is the entry point used by the application to synchronize
// data access to the device.  Using this function eliminates the
// need for the application to worry about threads.
void HapticsClass::synchFromServo()
{

printf("Estas en synchFromServo \n");
Sleep(100);
hdlCreateServoOp(GetStateCB, this, bBlocking);
}

// GetStateCB calls this function to do the actual data movement.
void HapticsClass::synch()
{
//printf("Estas en synch\n");
//Sleep(100);
// m_positionApp is set in cubeContact().
m_buttonApp = m_buttonServo;
}

void HapticsClass::setForce(double a_ref1)
{
printf("Estas en setForce\n");
Sleep(100);
m_ref1 = a_ref1;
}

// A utility function to handle matrix multiplication.  A production application
// would have a full vector/matrix math library at its disposal, but this is a
// simplified example.
void HapticsClass::vecMultMatrix(double srcVec[3], double mat[16], double dstVec[3])
{
// printf("Estas en vecMultMatrix\n");
//Sleep(100);

dstVec[0] = mat[0] * srcVec[0]
+ mat[4] * srcVec[1]
```

```
+ mat[8] * srcVec[2]
+ mat[12];

dstVec[1] = mat[1] * srcVec[0]
+ mat[5] * srcVec[1]
+ mat[9] * srcVec[2]
+ mat[13];

dstVec[2] = mat[2] * srcVec[0]
+ mat[6] * srcVec[1]
+ mat[10] * srcVec[2]
+ mat[14];
}

// Here is where the heavy calculations are done.  This function is
// called from ContactCB to calculate the forces based on current
// cursor position and cube dimensions.  A simple spring model is
// used.
void HapticsClass::feedForce()
{
// printf("Estas en feedForce \n");
//Sleep(100);
// Convert from device coordinates to application coordinates.
vecMultMatrix(m_positionServo, m_transformMat, m_positionApp);

//m_forceServo[0] = -7.5*xp-25*dxp;
//m_forceServo[0] = -7.5*(xp-1)-25*dxp;
//PARA GANANCIA LINEAL
m_forceServo[0] = -xps-25*dxp;
m_forceServo[1] = -7.5*yp-25*dyp;
m_forceServo[2] = -12.5*zp-25*dzp;

// Skip the whole thing if not initialized
if (!m_inited) return;

double posC[3]; //Auxiliares para las posiciones

// Get the cursor position in Local Coordinates.
posC[0] = m_positionApp[0];
posC[1] = m_positionApp[1];
posC[2] = m_positionApp[2];
//x1=1;
//m_forceServo[0] = -kf1*(kq1*posC[0]);
//m_forceServo[2] = -m_kf2*m_Ef2;

// Saturaciones
//if(m_forceServo[0]<-m_ref1) m_forceServo[0]=-m_ref1;
//if(m_forceServo[0]>m_ref1) m_forceServo[0]=m_ref1;

//if(m_forceServo[2]<-4) m_forceServo[2]=-4;
//if(m_forceServo[2]>4) m_forceServo[2]=4;
m_ref1=0;
//x1=posC[0];
//printf("X=%.2lf\tref1=%.2lf\tFx=%.2lf\n",posC[0],m_ref1,m_forceServo[0]);
//Sleep(200);
```

```
}

// Interface function to get current position
void HapticsClass::getPosition(double pos[3])
{
//printf("Estas en getPosition \n");
//Sleep(100);
double posC[3];
pos[0] = m_positionApp[0];
pos[1] = m_positionApp[1];
pos[2] = m_positionApp[2];
printf("X=%.2lf\n",posC[0]);
}

// Interface function to get button state.  Only one button is used
// in this application.
bool HapticsClass::isButtonDown()
{
// printf("Estas en isButtonDown \n");
//Sleep(100);
//return (0);
return m_buttonApp;
}

// For this application, the only device status of interest is the
// calibration status.  A different application may want to test for
// HDAL_UNINITIALIZED and/or HDAL_SERVO_NOT_STARTED
bool HapticsClass::isDeviceCalibrated()
{
//printf("Estas en isDeviceCalibrated \n");
//Sleep(100);
unsigned int state = hdlGetState();
//return (0);
return ((state & HDAL_NOT_CALIBRATED) == 0);

}
```