UNIVERSIDAD DEL PAPALOAPAN

CAMPUS LOMA BONITA

INGENIERÍA EN MECATRÓNICA

SEGUIMIENTO DE TRAYECTORIAS CON EVASIÓN DE COLISIONES PARA UN ROBOT MÓVIL AUTOMINY 4.0.

Tesis profesional para obtener el Título de:

INGENIERO EN MECATRÓNICA

Presenta:

DARIEL GUSTAVO HERNÁNDEZ MONTALVO

Director de tesis:

Dr. Jesús Santiaguillo Salinas

LOMA BONITA, OAX. 2024

UNIVERSIDAD DEL PAPALOAPAN

Campus Loma Bonita

Oficio: IMEC-23-103
Loma Bonita, Oaxaca; a 17 de noviembre de 2023.
Asunto: Solicitud de autorización de impresión de tesis

Dra. Tania Zúñiga Marroquín
Encargada de despacho de Vice-Rector Académico

PRESENTE

En base al artículo 120 del reglamento de alumnos, por medio de la presente solicito a usted la aprobación de la impresión final de la tesis y la programación del examen profesional del egresado de la carrera de Ingeniería en Mecatrónica **C. Dariel Gustavo Hernández Montalvo,** que presenta la tesis titulada **"Seguimiento de Trayectorias con Evasión de Colisiones para un Robot Móvil Autominy 4.0",** la tesis estuvo bajo la dirección de Dr. Jesús Santiaguillo Salinas.

Sin otro particular quedo atento para cualquier aclaración.

A t e n t a m e n t e
terra uberrima, mens aperta
Bou lo tama, chi, ji, ji

UNIVERSIDAD PAPALOAPAN
UNPA
terra uberrima, mens aperta
Bou Lo-tamá, chi ji já
OAXACA
FATURA CARRE
INGENIERIA
DATE

M.C. Luis Alberto Hernández Zuccolotto
Jefe de la Carrera de Ingeniería en Mecatrónica

c.c.p.: L.P. Yesenia Barrientos Arenal. Jefa del departamento de Servicios Escolares.
c.c.p: Archivo

UNIVERSIDAD DEL PAPALOAPAN

Campus Loma Bonita

Oficio: IMEC-23-105
Loma Bonita, Oaxaca, a 17 de noviembre de 2023

M.E. Yesenia Barrientos Arenal
Jefa del Departamento de Servicios Escolares
PRESENTE

Mediante la presente, le informo que la jefatura de carrera a mi cargo, con visto bueno de la Vice-Rectoría Académica, ha designado a los siguientes profesores como sinodales para examen profesional del **C. Dariel Gustavo Hernández Montalvo,** quien defenderá su trabajo de tesis titulado **"Seguimiento de Trayectorias con Evasión de Colisiones para un Robot Móvil Autominy 4.0"**, para obtener el título de Licenciado en Ingeniería en Mecatrónica.

Titulares:

Presidente: **Dr. Hiram Netzahualcóyotl García Lozano**
Secretario: **M.C. Rafael Fernando González Zarate**
Vocal: **Dr. Jesús Santiaguillo Salinas**

Suplentes:

M.C. Edmundo Mendieta Fernández
M.C. Luis Alberto Hernández Zuccolotto

A t e n t a m e n t e
terra uberrima, mens aperta
Bou lo-tama, chi. jf jú

M.C. Luis Alberto Hernández Zuccolotto
Jefe de Carrera de Ingeniería en Mecatrónica

Dra. Tania Zúñiga Marroquín
Encargada de despacho de Vice-Rector Académico

c.c.p. Dra. Tania Zúñiga Marroquín. Encargada de despacho de Vice-Rector académico. Para su conocimiento
c.c.p. Archivo

II

Agradecimientos

A Dios por darme la vida, la sabiduría y la oportunidad de culminar mis estudios de licenciatura, y a quien dedico mis talentos y conocimientos adquiridos para servirle de mejor manera a Él y a la humanidad.

A Mamá y papá, quienes me han dado amor y todo en la vida, quienes siempre estan ahí para apoyarme e impulsarme a seguir adelante. A quienes veo todos los días como se esfuerzan por salir adelante. Mi amor y mas grande admiración para ellos. Este logro es de ustedes.

A mi hermano y mejor amigo Isaí. Por ser mi confidente y compañero de aventuras de por vida.

A mis amigos Cristobal y Jafet. Con quienes compartí muchos buenos momentos y experiencias durante estos años de universidad, a quienes aprecio, estimo mucho y a quienes les deseo mucho exito en su vida.

A mis profesores, al Dr. Hiram, al M.C. Rafael y al Dr. Jesús, por su apoyo durante la elaboración de este trabajo y por la dedicación que cada uno de ellos pone en enseñar.

Resumen

El presente trabajo está enfocado en el diseño, simulación e implementación de una estrategia de control para el seguimiento de trayectorias, considerando evasión de colisiones de un robot móvil terrestre en configuración Ackerman, AutoMiny 4.0. Este vehículo cuenta con cuatro ruedas, las ruedas traseras proporcionan tracción y las ruedas delanteras la dirección. La finalidad de este trabajo es lograr que el robot se desplace dentro de un ambiente controlado, siguiendo una trayectoria específica. Adicionalmente, en caso que exista riesgo de colisión con algún objeto que se encuentre obstruyendo su camino, pueda evitarlo. El control seleccionado para el seguimiento de trayectoria está acotado por una función tangente hiperbólico, esto con la finalidad de no saturar la señal de control. Para lograr la evasión de colisiones se utilizan campos vectoriales repulsivos (RVF por sus siglas en inglés). Los resultados teóricos obtenidos son validados a través de una simulación numérica y de manera experimental. La implementación se llevó a cabo mediante ROS, la detección de los obstáculos se realizó mediante un sensor LiDAR (Laser Imaging Detection and Ranging) y la posición y orientación del AutoMiny se obtuvo mediante un sistema de cámaras OptiTrack. Estas cámaras infrarrojas obtienen información de coordenadas espaciales mediante la detección de marcadores colocados sobre el AutoMiny, por medio del software Motive.

Abstract

The present work is focused on the design, simulation and implementation of a control strategy for trajectory tracking, considering collision avoidance of a land mobile robot in Ackerman configuration, AutoMiny 4.0. This vehicle has four wheels, the rear wheels provide traction and the front wheels provide steering. The purpose of this work is to ensure that the robot moves within a controlled environment, following a specific trajectory. Additionally, if there is a risk of collision with an object that is obstructing the path, it can avoid it. The control selected for trajectory tracking is bounded by a hyperbolic tangent function, in order not to saturate the control signal. To achieve collision avoidance, repulsive vector fields (RVF) are used. The theoretical results obtained are validated through a numerical simulation and experimentally. The implementation was carried out using ROS, the detection of obstacles was carried out using a LiDAR (Laser Imaging Detection and Ranging) sensor and the position and orientation of the AutoMiny was obtained using an OptiTrack camera system. These infrared cameras obtain spatial coordinate information by detecting markers placed on the AutoMiny, using Motive software.

Contenido

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

En este capítulo se da una breve explicación sobre la importancia que tienen los sistemas de seguridad en automóviles, para salvaguardar la vida de los ocupantes. Se introduce el AutoMiny 4.0 como plataforma experimental para la investigación de nuevas estrategias de control para vehículos autónomos, además se presentan algunos trabajos realizados ocupando esta plataforma experimental. Adicionalmente se abordan algunos trabajos relacionados con el problema de la evasión de colisiones utilizando distintas estrategias, en particular utilizando campos vectoriales repulsivos. Posteriormente se plantean los objetivos del trabajo de tesis, se da la justificación y finalmente, se muestra la organización de la tesis.

1.1. Estado del Arte

El automóvil es sin lugar a dudas uno de los grandes inventos que cambió la historia de la humanidad, no solo revolucionó la forma en que los seres humanos se transportan, si no que condujo a un futuro más productivo para

la humanidad. Junto al petróleo permitió el avance en guerras, la creación de incontables negocios, y ha abierto la posibilidad de desarrollar el mundo hasta los increíbles límites que en la actualidad se encuentra [10]. Hoy en día es evidente el gran avance tecnológico que tienen los automóviles. Vehículos cada vez más inteligentes y capaces de realizar acciones independientes para salvaguardar la vida de los pasajeros. Parte de estos avances son los elementos de seguridad en los vehículos, los cuales son diseñados para prevenir los accidentes y en el caso de que estos sucedan, buscan disminuir el daño que podrían recibir los ocupantes del vehículo, así como el propio vehículo. El ejemplo más común son las bolsas de aire, las cuales consisten en bolsas que, mediante un sistema pirotécnico, se inflan en fracciones de segundo cuando el coche choca con un objeto a una velocidad considerable [20]. En los últimos años se ha optado por incorporar en los vehículos importantes innovaciones en materia de seguridad, como el ABS (Sistema antibloqueo de frenos), el ESP (Control electrónico de estabilidad), sensores de aceleración que actúan con las bolsas de aire, sistemas de posicionamiento GPS, detectores de distancia y de asistencia para estacionar [21], dirección asistida, indicador de mantenimiento, control de presión de los neumáticos, entre otros, con el fin de disminuir el riesgo de accidentes automovilísticos [22]. Según cifras del INEGI, en el año 2020 se registraron un total de 301,678 accidentes de tránsito en las zonas urbanas de México. Durante ese año, uno de cada 100 eventos correspondió a accidentes con pérdidas humanas, mientras que en 18 de cada 100 solo resultaron heridos [14]. Una alternativa para reducir el número de accidentes es cambiar completamente a sistemas de conducción autónoma

vehicular. Los vehículos autónomos (AV´s) se han discutido ampliamente en los últimos años tanto en trabajos académicos y de la industria. Se espera que estos se integren con nuestro estilo de vida en una o más formas, como sistemas de entrega de drones autónomos, automóviles sin conductor, vehículos guiados automatizados en almacenes, dispositivos autónomos para asistentes domésticos y para soluciones de energías sustentables [15]. Sin embargo la transición completa a vehículos autónomos podría tardar muchos años. Por tal motivo, resulta relevante el estudio de nuevas estrategias de seguridad vehicular. Estas estrategias deben ser abordadas en centros de investigación (laboratorios de robótica móvil) antes de ser implementadas en vehículos comerciales.

Con este fin se han desarrollado diversas plataformas experimentales como lo es el AutoMiny versión 4.0. Esta plataforma es un modelo de vehículo autónomo a escala 1:10 desarrollado en la Freie Universität Berlin (Universidad Libre de Berlin), con fines educativos y de investigación [1]. En esta plataforma se han desarrollado numerosos trabajos de investigación como el estudio del control de seguimiento de trayectorias [27], un sistema de control de crucero adaptativo basado en lógica difusa [5], detección de semáforos por medio de visión artificial para modelos de autos [6], navegación autónoma sin obstáculos, con un obstáculo estático y estacionamiento autónomo a travéz de visión artificial [8] entre otros, siendo este último donde la evasión de obstáculos toma más relevancia.

El problema de la evasión de obstáculos o de colisiones es un tema actual de interés, en donde, a fin de aportar en su solución se han desarrollando

distintas estrategias como: la utilización de un controlador difuso para la navegación autónoma en robots móviles con evasión de colisiones de obstáculos fijos [17], esquema de evasión de colisiones basado en funciones potenciales repulsivas [13] [9], evasión de colisiones mediante el uso de campos vectoriales repulsivos (RVF's por sus siglas en inglés) del tipo foco inestable [26], [24], o la evasión de colisiones usando aprendizaje reforzado profundo [30], entre otros.

1.2. Objetivos

A continuación se presenta el objetivo general de la tesis.

Realización de la navegación del AutoMiny a través de distintos tipos de trayectorias obteniendo la información necesaria sobre su posición y orientación por medio de un sistema de cámaras OptiTrack, así como también el diseño de una estrategia de control que le permita detectar obstáculos; y que pueda evitar colisionar con los mismos a través de la utilización de RVF's.

Objetivos específicos:

- Realizar la localización del robot por medio de un sistema de cámaras OptiTrack simulando un GPS.
- Implementar un algoritmo de control para la navegación autónoma, para el seguimiento de trayectorias.
- Realizar la detección de obstáculos por medio de un sistema LiDAR

abordo del AutoMiny.

- Diseñar e implementar una estrategia para la evasión de colisiones por medio de una ley de control complementaria, basada en RVF.

1.3. Justificación

La justificación de este trabajo de tesis es el siguiente.

- Referente al tema de seguridad vial. Resulta relevante ampliar los estudios con respecto a la evasión de colisiones, como se ilustra en la cita [18] "Los vehículos autónomos prometen numerosas mejoras al tráfico vehicular, al prever un aumento en la capacidad de la carretera y el flujo de tráfico debido al tiempo de respuesta más rápido, y un menor consumo de combustible y de contaminación gracias a una conducción más previsora, y con suerte, menos accidentes gracias a los sistemas de prevención de colisiones".

- Muchos de los trabajos publicados que estudian la evasión de colisiones aplicados a un robot tipo Ackerman, son basados en visión artificial para la detección y evasión de los obstáculos como lo reportado en [8].

- A diferencia de los trabajos [19, 23] donde se aborda el problema de generación de trayectorias, dado un punto inicial y un punto final al cual debe llegar el robot, en este trabajo se plantea el seguimiento de trayectorias. Además por lo general, en estos trabajos tratan al robot móvil como una partícula que se encuentra bajo la influencia de un

campo potencial artificial el cual es típicamente definido como la suma de un potencial de atracción que atrae al robot a la meta designada y un potencial de repulsión que aleja al robot de los obstáculos. En estos trabajos el campo vectorial es continuo de forma radial, en los cuales se pueden presentar equilibrios no deseados. Para evitar estas dinámicas se propone el uso de campos vectoriales discontinuos.

1.4. Planteamiento del problema

Es necesario definir el problema al cual se quiere dar solución para saber con certeza cuales son los resultados que se desean obtener. Por lo tanto, el planteamiento del problema se enuncia a continuación.

Sea $m(t) = \left[\, m_x(t), m_y(t) \, \right]^T$ una trayectoria preestablecida continuamente diferenciable, ξ_{oi} la posición del i-ésimo obstáculo en el plano con $i = 1, \ldots, n$ y d la distancia mínima permitida a cada uno de los obstáculos. El objetivo de este trabajo de tesis es diseñar e implementar una estrategia de control, $u = f(P(t))$, para un robot AutoMiny 4.0, tal que:

- Se logre el seguimiento asintótico de la trayectoria deseada $m(t)$, (control de seguimiento):

$$\lim_{t \to \infty} (P(t) - m(t)) = 0. \tag{1.1}$$

- Se logre la evasión de colisiones contra un obstáculo fijo a la vez. Y el robot mantenga una distancia mínima de seguridad a dicho obstáculo:

$$\| \, P(t) - \xi_o \, \| \geq d, \forall t \geq 0. \tag{1.2}$$

- Se logre la evasión de colisiones contra un obstáculo en movimiento a la vez. Y el robot mantenga una distancia mínima de seguridad a dicho obstáculo.

$$\| P(t) - \xi_{o1}(t) \| \geq d, \forall t \geq 0. \tag{1.3}$$

- Se logre la evasión de colisiones contra más de un obstáculo en movimiento a la vez y el robot mantenga la distancia mínima de seguridad a dichos obstáculos:

$$\begin{aligned} \| P(t) - \xi_{o1}(t) \| &\geq d, \forall t \geq 0 \\ \| P(t) - \xi_{o2}(t) \| &\geq d, \forall t \geq 0 \\ &\vdots \\ \| P(t) - \xi_{on}(t) \| &\geq d, \forall t \geq 0. \end{aligned} \tag{1.4}$$

1.5. Conceptos Matemáticos

Para la obtención del modelo del AutoMiny, el diseño de las estrategias de control de seguimiento de trayectorias y de evasión de colisiones, así como los análisis de estabilidad de estas estrategias, se requiere del conocimiento de ciertos conceptos matemáticos relevantes. Estos conceptos se presentan a continuación.

Producto punto y producto cruz de vectores.

Teorema 1.1 *: Sean A y B dos vectores diferentes de cero. Si θ es el ángulo entre ellos, entonces el producto punto o escalar entre ellos es:*

$$\vec{A} \cdot \vec{B} = \parallel A \parallel \parallel B \parallel \cos\theta \tag{1.5}$$

Teorema 1.2 *: Sean A y B dos vectores diferentes de cero. Si θ es el ángulo entre ellos, entonces el producto cruz o vectorial entre ellos es:*

$$\vec{A} \times \vec{B} = \parallel A \parallel \parallel B \parallel \sin\theta \tag{1.6}$$

Norma euclidiana de un vector.

Definición 1.1 *: La norma euclidiana* $\| x \|$ *de un vector* $x \in \mathbb{R}^n$ *se define como*

$$\| x \| = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x^T x}, \tag{1.7}$$

donde sólo se considera la parte positiva de la raíz cuadrada. Es inmediato comprobar que la norma euclidiana satisface las siguientes propiedades:

- $\| x \| = 0$ *si y sólo si* $x = 0 \in \mathbb{R}^n$.
- $\| x \| > 0$, *para todo* $x \in \mathbb{R}^n$ *con* $x \neq 0 \in \mathbb{R}^n$.
- $\| \alpha x \| = |\alpha| \, \| x \|$, *para todo* $\alpha \in \mathbb{R}$ *y* $x \in \mathbb{R}^n$.
- $\| x \| - \| y \| \leq \| x + y \| \leq \| x \| + \| y \|$, *para todo* $x, y \in \mathbb{R}^n$.
- $\left| x^T y \right| \leq \| x \| \| y \|$, *para todo* $x, y \in \mathbb{R}^n$ *(desigualdad de Schwarz) [16].*

Matriz singular

Definición 1.2 *: Una matriz cuadrada* $A \in \mathbb{R}^{n \times n}$ *es singular si su determinante es nulo, i.e., si* $det[A] = 0$, *en caso contrario es no singular. Una característica de una matriz singular es que ésta no tiene inversa.*

Valores propios de una matriz cuadrada

Definición 1.3 *: Para cada matriz cuadrada* $A \in \mathbb{R}^{n \times n}$ *existen n valores propios (números complejos en general) denotados por* $\lambda_1 \{A\}, \lambda_2 \{A\}, , \lambda_n \{A\}$. *Los valores propios de la matriz* $A \in \mathbb{R}^{n \times n}$ *satisfacen:*

$det[\lambda_i \{A\} I - A] = 0$, *para* $i = 1, 2, \cdots, n$

donde $I \in \mathbb{R}^{n \times n}$ *es la matriz identidad de dimensión* n.

Norma espectral

Definición 1.4 *: La norma espectral* $\| A \|$ *de una matriz* $A \in \mathbb{R}^{n \times m}$ *se define como:*

$$\| A \| = \sqrt{\lambda_{Max} \{A^T A\}}, \tag{1.8}$$

donde $\lambda_{Max} \{A^T A\}$ *denota el valor propio máximo de la matriz simétrica* $A^T A \in \mathbb{R}^{m \times m}$. *En el caso particular de matrices simétricas* $A = A^T \in \mathbb{R}^{n \times n}$, *se tiene que:*

- $\| A \| = max_i |\lambda_i \{A\}|$.
- $\| A^{-1} \| = \frac{1}{min_i |\lambda_i \{A\}|}$.

Matriz de rotación

Definición 1.5 *: En álgebra lineal, una matriz de rotación es la matriz que representa una rotación de un vector en el espacio euclídeo [28]. Al rotar un vector* $(x, y, z) \in \mathbb{R}^3$ *alrededor del origen por un ángulo* θ *obtenemos otro vector* $(x', y', z') \in \mathbb{R}^3$. Dicho vector puede ser obtenido por medio de la matriz de rotación como se expresa a continuación.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R(\theta) \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{1.9}$$

Para expresar la rotación alrededor del eje coordenado z se utiliza la siguiente matriz, conocida como matriz fundamental.

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.10}$$

En este caso se tiene que $R_z(\theta)$ rota el plano xy alrededor del origen por un ángulo θ.

Campos vectoriales.

Definición 1.6 *: Sea D un conjunto en $\mathbb{R}^2$ (una región plana). Un campo vectorial en $\mathbb{R}^2$ es una función F que asigna a cada punto (x, y) en D un vector bidimensional $F(x, y)$ [29].*

$$F(x, y) = P(x, y)i + Q(x, y)j = \langle P(x, y), Q(x, y) \rangle \tag{1.11}$$

Los campos vectoriales tienen ciertas características que los describen y clasifican:

1. Campos vectoriales conservativos (Fig. 1.1): Estos campos tienen la propiedad de que la integral de línea de un campo vectorial a lo largo de una curva cerrada es cero. Esto significa que el trabajo realizado por el campo vectorial al mover un objeto de un punto a otro depende sólo de los puntos inicial y final, y no de la trayectoria seguida entre ellos. Un campo vectorial F se llama campo vectorial conservativo si es el gradiente de alguna función escalar, es decir, si existe una función f tal que

$F = \bigtriangledown f$. En esta situación, f se llama función potencial de F. Otra forma de comprobar que un campo F es conservativo es si P y Q tienen derivadas parciales continuas de primer orden en un dominio D, entonces a todo lo largo de D se tiene

$$\frac{\partial P}{\partial y} = \frac{\partial Q}{\partial x} \tag{1.12}$$

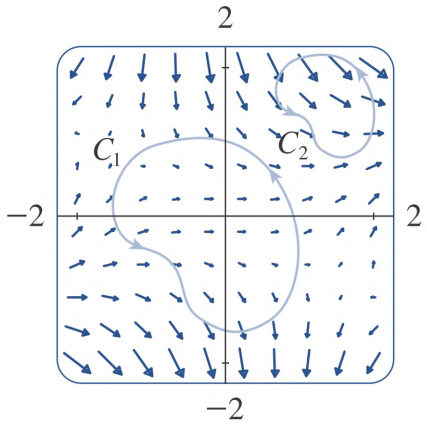


Figura 1.1: Campo vectorial conservativo.

2. Campos vectoriales irrotacionales (Fig. 1.2): Los campos vectoriales irrotacionales tienen la propiedad de que su rotacional es cero en todas partes ($rotF = 0$). Esto significa que la circulación de un campo vectorial a lo largo de cualquier camino cerrado es cero. Si $F = Pi + Qj + Rk$ es un campo vectorial en $\mathbb{R}^3$ y todas las derivadas parciales de P, Q y R existen, el rotacional de F es el campo vectorial en $\mathbb{R}^3$ definido por:

$$rotF = \bigtriangledown \times F = \begin{vmatrix} i & j & k \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ P & Q & R \end{vmatrix} \tag{1.13}$$

Un campo vectorial irrotacional también es un campo vectorial conservativo.

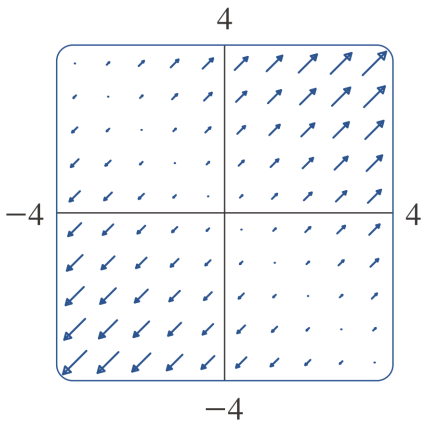


Figura 1.2: Campo vectorial irrotacional.

3. Campos vectoriales solenoidales: Otra propiedad de los campos vectoriales es la divergencia. Si $F = Pi + Qj + Rk$ es un campo vectorial en $\mathbb{R}^3$ y todas las derivadas parciales de P, Q y R existen, entonces la divergencia de F es la función de tres variables definida por:

$$divF = \bigtriangledown \cdot F = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z} \tag{1.14}$$

Físicamente se interpreta como la propiedad de un campo vectorial para a partir de un punto focal los vectores se dirijan hacia la periferia del campo (hacia afuera) como se observa en la Fig. 1.3. Implicitamente si la divergencia de un campo es negativa se le llama convergencia, la cual es la propiedad de que los vectores apunten hacia un punto focal del campo. Los campos vectoriales solenoidales tienen la propiedad de que su divergencia es cero en todas partes. Esto significa que la cantidad de flujo de un campo vectorial que sale de cualquier superficie cerrada es igual a la cantidad que entra (Fig. 1.4).

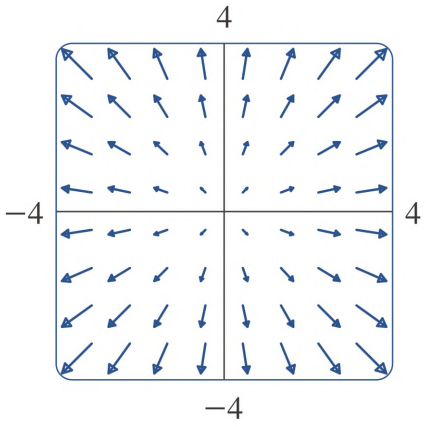


Figura 1.3: Campo vectorial divergente.

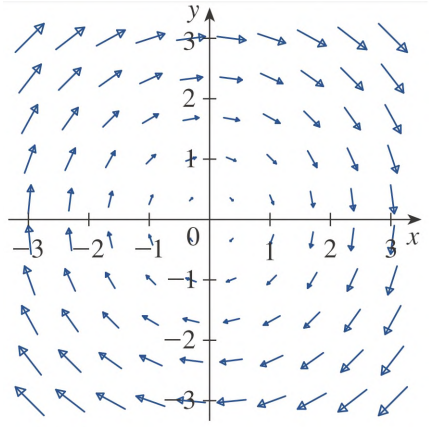


Figura 1.4: Campo vectorial solenoidal.

4. Campos vectoriales gradientes: Los campos vectoriales gradientes (Fig. 1.5), son aquellos que se pueden expresar como el gradiente de una función escalar, para funciones de dos variables se expresa en la ecuación (1.15) . Estos campos tienen la propiedad de que el trabajo realizado por el campo vectorial a lo largo de cualquier camino entre dos puntos es igual a la diferencia de valores de la función escalar en esos dos puntos.

$$\bigtriangledown f(x,y) = f_x(x,y)i + f_y(x,y)j \tag{1.15}$$

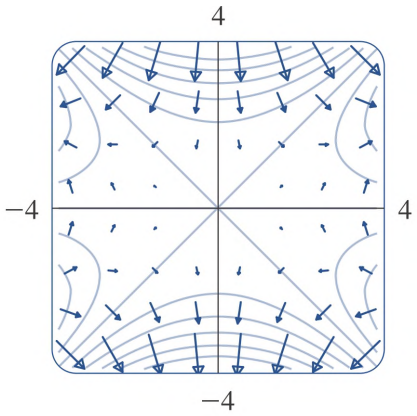


Figura 1.5: Campo vectorial gradiente.

5. Campos vectoriales no conservativos: Estos campos no tienen la propiedad de que la integral de línea a lo largo de una curva cerrada sea cero. En otras palabras, el trabajo realizado por el campo vectorial al mover un objeto de un punto a otro depende de la trayectoria seguida entre ellos.

Un tipo en particular son los campos vectoriales repulsivos de foco inestable, los cuales presentan algunas de las características que se mencionaron

anteriormente, como divergencia que parte de un foco central y rotación en sentido antihorario. Este tipo de campo vectorial se utiliza en la estrategia para la evasión de colisiones debido a la divergencia que posee, los vectores apuntan hacia el exterior del foco, por eso se le llama repulsivo e inestable. En la Fig. 1.6 podemos ver una imagen de este tipo de campo vectorial.

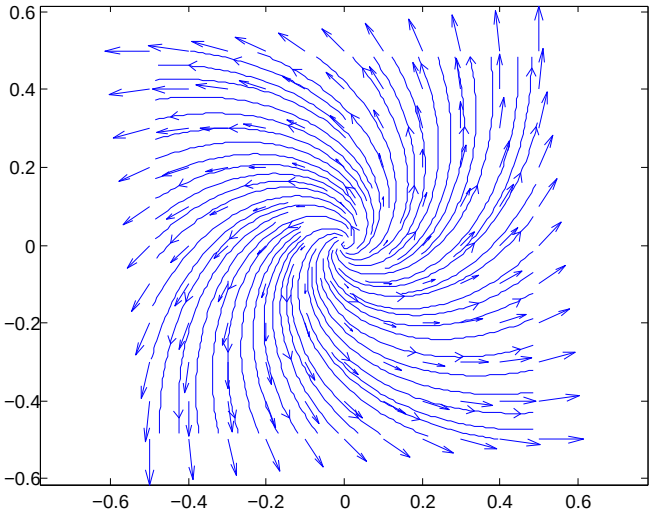


Figura 1.6: Campo vectorial repulsivo del tipo foco inestabale.

Teorema de estabilidad de Lyapunov

La teoría de estabilidad de Lyapunov tiene como principal objetivo estudiar el comportamiento de sistemas dinámicos descritos por ecuaciones diferenciales de la forma:

$$\dot{x}(t) = f(t, x(t)), \qquad x(0) \in \mathbb{R}^n \quad \forall t \geq 0, \tag{1.16}$$

donde el vector $x(t) \in \mathbb{R}^n$ se refiere al estado del sistema dinámico representado por (1.16) y $x(0) \in \mathbb{R}^n$ se denomina la condición inicial o estado inicial.

La función $f : \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}^n$ es una función continua en t y $x(t)$, y se supone que es tal que:

- La Ecuación (1.16) tiene una solución única en el intervalo $[0, \infty)$ correspondiente a cada condición inicial de $x(0)$.
- Si $x(t)$ es la solución de (1.16) correspondiente a la condición inicial $x(0)$, entonces $x(t)$ depende de una manera continua del estado inicial $x(0)$.

Si la función f no depende explícitamente del tiempo, i.e., $f(t, x(t)) = f(x(t))$, luego la Ecuación (1.16) se denomina autónoma. Si $f(t, x(t)) = A(t)x(t) + u(t)$ con $A(t)$ una matriz cuadrada de dimensión n y siendo $A(t)$ y $u(t)$ funciones únicamente de t o constantes, entonces la Ecuación (1.16) es lineal. En caso contrario es no lineal.

Teorema 1.3 *: El origen $x = 0 \in \mathbb{R}^n$ es un estado de equilibrio estable de la Ecuación* (1.16)*, si existe una función candidata de Lyapunov $V(t, x)$ tal que su derivada temporal satisfaga:*

$$\dot{V}(t, x) \leq 0, \qquad \forall t \geq 0 \qquad \textit{al menos para} \quad \|x\| \quad \textit{pequeña.} \tag{1.17}$$

El teorema anterior da condiciones suficientes para estabilidad del equilibrio en el sentido de Lyapunov. Conviene apuntar que la conclusión del teorema se mantiene obviamente si $\dot{V}(t, x) \leq 0$ para todo $t \geq 0$ y para todo $x \in \mathbb{R}^n$, o si la función candidata de Lyapunov $V(t, x)$ es una función definida positiva (globalmente) en lugar de ser definida positiva localmente. [16]

Capítulo 2

Modelado matemático

En este capítulo se desarrolla el modelo matemático utilizado para el control del AutoMiny 4.0, a partir del análisis cinemático del mismo.

2.1. Modelo cinemático del robot AutoMiny 4.0

El modelo cinemático de un sistema describe el movimiento de dicho sistema a partir de sus velocidades, sin considerar las fuerzas que causan el movimiento. Con la finalidad de simplificar el modelo cinemático de un robot móvil con ruedas se toman en cuenta las siguientes consideraciones:

- El robot se mueve sobre una superficie plana.
- No existen elementos flexibles en la estructura del robot incluyendo las ruedas.
- Se desprecia todo tipo de fricción en los elementos móviles del robot móvil contra el suelo.

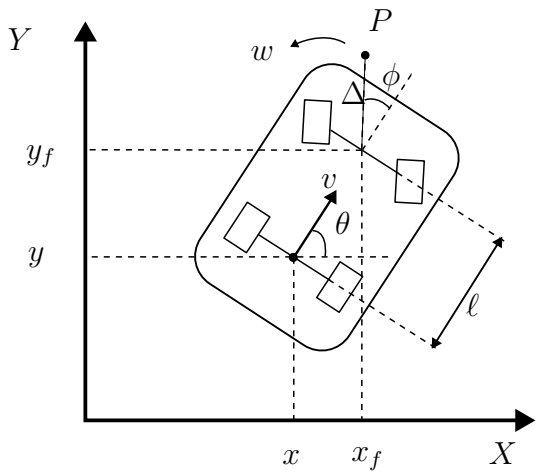


Figura 2.1: Esquema del robot AutoMiny 4.0

Con base en estas consideraciones, se procede a obtener el modelo matemático. Sea el AutoMiny un robot móvil tipo Ackerman como el mostrado en la Fig. 2.1, donde el punto medio de las ruedas traseras del robot tiene coordenadas $\xi = [x, y]^T$, θ es el ángulo de orientación del cuerpo del robot con respecto al eje X, ϕ es el ángulo de dirección de las ruedas delanteras, ℓ es la distancia entre los ejes de las ruedas traseras y delanteras, v es la velocidad del punto medio del eje de las ruedas traseras del robot y w es la velocidad angular de dirección.

Su modelo matemático se obtiene por medio del análisis cinemático de la figura anterior. Las componentes rectangulares de la velocidad v se obtienen como sigue:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta. \end{aligned} \tag{2.1}$$

La velocidad angular de la dirección las ruedas delanteras del robot es

$$w = \dot{\phi}. \tag{2.2}$$

De acuerdo con [7], en cualquier instante dado, la velocidad de las diversas partículas o puntos sobre un cuerpo rígido es la misma como si el cuerpo girara alrededor de cierto eje perpendicular a su plano. Este punto se conoce como eje de rotación instantáneo. Este eje interseca el plano sobre el que se encuentra el cuerpo en el punto ICC (Centro instantáneo de curvatura, por sus siglas en inglés). De este modo la velocidad lineal de un punto sobre el cuerpo rígido es igual al producto de la velocidad angular del cuerpo por la distancia al ICC.

Considerando lo anterior (Fig. 2.2) se obtienen las siguientes relaciones

$$\tan\phi = \frac{\ell}{d_r} \tag{2.3}$$

$$v = d_r \cdot \dot{\theta}. \tag{2.4}$$

Despejando d_r de (2.3) y sustituyendo en (2.4) obtenemos

$$v = \frac{\ell}{\tan\phi} \cdot \dot{\theta} \Rightarrow \dot{\theta} = v \cdot \frac{\tan\phi}{\ell}. \tag{2.5}$$

En forma matricial, el modelo cinemático es el que sigue:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ \frac{\tan\phi}{\ell} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \tag{2.6}$$

Se desea controlar el punto medio de las ruedas traseras ξ, por lo tanto se tiene que

$$\dot{\xi} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = A(\theta) \begin{bmatrix} v \\ w \end{bmatrix}, \tag{2.7}$$

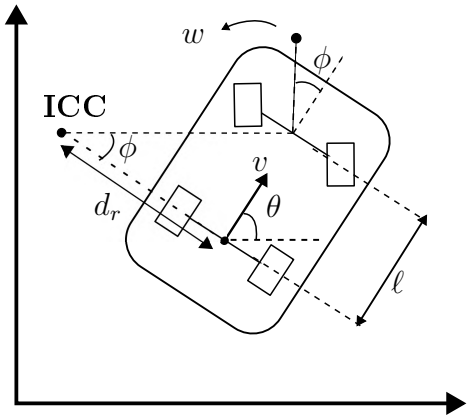


Figura 2.2: Centro de rotación instantaneo de un robot Ackerman

donde $A(\theta)$ es la matriz de desacoplamiento. La matriz $A(t)$ es singular, i.e., el $\det[A(\theta)] = 0$, por lo que no es posible obtener $A^{-1}(\theta)$ necesaria para la estrategia de control. Por lo tanto en este trabajo de tesis, se propone realizar el control de un punto P (ver Fig. 2.1), en la parte frontal del robot.

2.2. Modelo cinemático del punto frontal P

Para realizar el control del punto frontal P se procede de la siguiente manera. Este punto tiene coordenadas

$$P = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} x_f + \Delta\cos(\theta + \phi) \\ y_f + \Delta\sin(\theta + \phi) \end{bmatrix}. \tag{2.8}$$

Aquí Δ es la distancia del punto medio de las ruedas delanteras del robot a el punto frontal P a controlar ($\Delta \neq 0$) y $\xi_f = [x_f, y_f]^T$ son las coordenadas

cartesianas del punto medio del eje de las ruedas delanteras dadas por

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} x + \ell\cos\theta \\ y + \ell\sin\theta \end{bmatrix}. \tag{2.9}$$

Para obtener la cinématica del punto P se sustituyen las coordenadas de ξ_f en el punto P resultando

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} x + \ell\cos\theta + \Delta\cos(\theta+\phi) \\ y + \ell\sin\theta + \Delta\sin(\theta+\phi) \end{bmatrix}. \tag{2.10}$$

Se deriva la ecuación 2.10 con respecto al tiempo, y se obtiene la cinemática del punto P

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \begin{bmatrix} \dot{x} - \dot{\theta}\ell\sin\theta - (\dot{\theta}+\dot{\phi})\Delta\sin(\theta+\phi) \\ \dot{y} + \dot{\theta}\ell\cos\theta + (\dot{\theta}+\dot{\phi})\Delta\cos(\theta+\phi) \end{bmatrix},$$

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \begin{bmatrix} v\cos\theta - (\frac{\tan\phi}{\ell}v)\ell\sin\theta - (\frac{\tan\phi}{\ell}v + w)\Delta\sin(\theta+\phi) \\ v\sin\theta + (\frac{\tan\phi}{\ell}v)\ell\cos\theta + (\frac{\tan\phi}{\ell}v + w)\Delta\cos(\theta+\phi) \end{bmatrix}.$$

Se sustituye $\delta = (\theta + \phi)$

$$\begin{aligned} \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} &= \begin{bmatrix} v\cos\theta - v\tan\phi\sin\theta - \frac{\tan\phi}{\ell}v\Delta\sin\delta - w\Delta\sin\delta \\ v\sin\theta + v\tan\phi\cos\theta + \frac{\tan\phi}{\ell}v\Delta\cos\delta + w\Delta\cos\delta \end{bmatrix}, \\ &= \begin{bmatrix} (\cos\theta - \tan\phi(\sin\theta + \Delta\sin\delta/\ell))v - w\Delta\sin\delta \\ (\sin\theta + \tan\phi(\cos\theta + \Delta\cos\delta/\ell))v + w\Delta\cos\delta \end{bmatrix}, \\ &= \begin{bmatrix} \cos\theta - \tan\phi(\sin\theta + \Delta\sin\delta/\ell) & -\Delta\sin\delta \\ \sin\theta + \tan\phi(\cos\theta + \Delta\cos\delta/\ell) & \Delta\cos\delta \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}, \\ &= A(\theta,\phi)\begin{bmatrix} v \\ w \end{bmatrix}. \end{aligned} \tag{2.11}$$

La matriz $A(\theta, \phi)$ asi construída es no singular. Esto se verifica debido a que el determinante de la matríz es distinto de cero, como se muestra a continuación.

$$
\begin{aligned}
\det[A(\theta,\phi)] &= \Delta\cos\delta\cos\theta - \tan\phi\Delta\cos\delta\sin\theta - \tan\phi\Delta\cos\delta\Delta\sin\delta/\ell + ... \\
&... \quad +\Delta\sin\delta\sin\theta + \Delta\sin\delta\tan\phi\cos\theta + \Delta\sin\delta\tan\phi\Delta\cos\delta/\ell, \\
&= \Delta(\cos\delta\cos\theta + \sin\delta\sin\theta) + \Delta\tan\phi(\sin\delta\cos\theta - \cos\delta\sin\theta) + ... \\
&... \quad +\Delta^2\tan\phi/\ell(\sin\delta\cos\delta - \cos\delta\sin\delta), \\
&= \Delta\cos(\delta-\theta) + \Delta\tan\phi\sin(\delta-\theta), \\
&= \Delta\cos(\theta+\phi-\theta) + \Delta\tan\phi\sin(\theta+\phi-\theta), \\
&= \Delta\cos(\phi) + \Delta\tan\phi\sin(\phi), \\
&= \Delta\cos\phi + \Delta\frac{\sin\phi}{\cos\phi}\sin\phi, \\
&= \Delta(\cos\phi + \frac{\sin^2\phi}{\cos\phi}), \\
&= \Delta(\cos\phi + \frac{1-\cos^2\phi}{\cos\phi}), \\
&= \Delta(\cos\phi + \frac{1}{\cos\phi} - \frac{\cos^2\phi}{\cos\phi}), \\
&= \Delta(\cos\phi + \frac{1}{\cos\phi} - \cos\phi), \\
&= \frac{\Delta}{\cos\phi}. \qquad (2.12)
\end{aligned}
$$

La matriz $A(\theta, \phi)$ obtenida por este método, es la matriz de desacoplamiento del modelo construido. Debido a que el ángulo ϕ toma valores de $-\frac{\pi}{2} < \phi < \frac{\pi}{2}$, en la mayoría de los casos prácticos, su determinante es distinto de cero. La cinemática del punto P queda expresada por la siguiente

ecuación matricial.

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \begin{bmatrix} \cos\theta - \tan\phi(\sin\theta + \Delta\frac{\sin\delta}{\ell}) & -\Delta\sin\delta \\ \sin\theta + \tan\phi(\cos\theta + \Delta\frac{\cos\delta}{\ell}) & \Delta\cos\delta \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \tag{2.13}$$

En forma simplificada

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = A(\theta,\phi) \begin{bmatrix} v \\ w \end{bmatrix}. \tag{2.14}$$

Esta ecuación es la utilizada para la estrategia de control.

Capítulo 3

Diseño de las estrategias de control

En este capítulo se desarrollan cada una de las estrategias de control para el AutoMiny, tanto para el seguimiento de trayectorias como para la evasión de colisiones.

3.1. Control para el seguimiento de trayectorias

Para el seguimiento de trayectorias se propone una ley de control acotada [26]. Esta estrategia consiste de un controlador basado en el modelo cinemático del robot. La estrategia de control impone una trayectoria al sistema compensando la dinámica del error con el mismo modelo del sistema.

La estrategia de control es la siguiente

$$u = \begin{bmatrix} v \\ w \end{bmatrix} = A(\theta,\phi)^{-1} \begin{bmatrix} -k_x \tanh(p_x - m_x) + \dot{m}_x \\ -k_y \tanh(p_y - m_y) + \dot{m}_y \end{bmatrix}, \tag{3.1}$$

$$\begin{bmatrix} v \\ w \end{bmatrix} = A(\theta,\phi)^{-1} \left(-K \tanh(P - m) + \dot{m}\right), \tag{3.2}$$

$$\begin{bmatrix} v \\ w \end{bmatrix} = A(\theta,\phi)^{-1}\lambda. \tag{3.3}$$

donde $A(\theta,\phi)^{-1}$ es la inversa de la matriz de desacoplamiento, $m(t)$ es la trayectoria deseada, $\dot{m}(t)$ es la velocidad de la trayectoria deseada, $K = diag\left(\ k_x, k_y\ \right)$ es una matriz diagonal de ganancias de control de seguimiento.

Proposición 3.1 . *Considere el sistema* (2.14) *y la ley de control* (3.3). *Suponga que* $k > 0$, *donde* $k = max(k_x, k_y)$. *Entonces en el sistema en lazo cerrado* (2.14)-(3.3) *el robot converge a la trayectoria deseada de navegación, i.e.* $\lim_{t\to\infty}(P(t) - m(t)) = 0$.

Demostración: Sustituyendo la ley de control (3.3) en (2.14) tenemos:

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = A(\theta,\phi)u,$$

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \begin{bmatrix} -k \tanh(p_x - m_x) + \dot{m}_x \\ -k \tanh(p_y - m_y) + \dot{m}_y \end{bmatrix},$$

$$\dot{P} = -K \tanh(P - m) + \dot{m}. \tag{3.4}$$

Los errores del sistema están dados por

$$\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} p_x - m_x \\ p_y - m_y \end{bmatrix} \Longrightarrow e = P - m, \tag{3.5}$$

por lo que la dinámica del error esta dada por

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \end{bmatrix} = \begin{bmatrix} -k\tanh(e_x) \\ -k\tanh(e_y) \end{bmatrix}, \tag{3.6}$$

$$\dot{e} = -K\tanh(e). \tag{3.7}$$

Para demostrar que el sistema es estable y que los errores convergerán asintóticamente a cero hacemos uso del teorema de estabilidad de Lyapunov (Ver Teorema 1.3) [16].

Proponemos una función candidata de Lyapunov $V = \frac{1}{2}e^T K^{-1} e$ y se deriva con respecto al tiempo a lo largo de las trayectorias del sistema,

$$\dot{V} = e^T K^{-1}\dot{e} = -e^T\tanh(e) < 0 \ \forall e \text{ con } e \neq 0. \tag{3.8}$$

Estas condiciones satisfacen el teorema de Lyapunov por lo que los errores convergen asintóticamente a cero.

Proposición 3.2 . *Considere el sistema* (2.14) *y la ley de control* (3.3), *entonces en el sistema de lazo cerrado* (2.14)-(3.3) *la velocidad del punto P está limitada por* $k\sqrt{2} + \eta$.

Demostración: De la ecuación (3.4) tenemos que la norma del vector de velocidad es

$$\begin{aligned} \| \dot{P} \| &= \| -Ktanh(P - m) + \dot{m} \| \\ &\leq \| -K \| \| tanh(P - m) \| + \| \dot{m} \|, \end{aligned}$$

donde

$$\begin{aligned} \| -K \| &= |-1| \| K \| = \rho(K), \\ &= max(k_x, k_y) = k. \end{aligned}$$

Aquí $\rho(K)$ es el radio espectral de K o norma espectral de la matriz K,

$$\| tanh(P - m) \| = \sqrt{2}$$

debido a $-1 \leq tanh(p_x - m_x), tanh(p_y - m_y) \leq 1$ y $\| \dot{m} \| = \eta$, con η la velocidad máxima de la trayectoria, se tiene $\| \dot{P} \| \leq k\sqrt{2} + \eta$.

El resultado obtenido será relevante para el diseño de la estrategia de control para la evasiíon de colisiones.

3.2. Estrategia para la evasión de colisiones

Con la estrategia de control para el seguimiento de trayectorias diseñada, se aborda el problema de evasión de colisiones. Esta estrategia utiliza una ley de control complementaria, basada en campos vectoriales repulsivos, la cual depende de la distancia entre el robot y el obstáculo a evadir [12] [25].

3.2.1. Evasión de colisiones entre el robot y un obstáculo en movimiento.

Primero se aborda el problema de la evasión de colisiones contra un obstáculo en movimiento.

Sea $P(t) = [p_x, p_y]^T$ las coordenadas del punto frontal seleccionado en el robot y $\xi_o(t) = [x_0, y_0]^T$ las coordenadas cartesianas de un obstáculo.

Entonces dada la distancia entre estas $\| P(t) - \xi_o(t) \|$, el robot está en peligro de colisión cuando

$$\| P(t) - \xi_o(t) \| \leq d, \tag{3.9}$$

donde d es la mínima distancia permitida entre el robot y el objeto. Con el fin de evitar colisiones, proponemos campos vectoriales repulsivos definidos de la siguiente manera

$$\beta = \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix} = \epsilon\delta \begin{bmatrix} (p_x - x_o) - (p_y - y_o) \\ (p_x - x_o) + (p_y - y_o) \end{bmatrix}, \tag{3.10}$$

donde $\epsilon > 0$ y δ es el parámetro de activación el cual está dado de la siguiente manera:

$$\delta = \begin{cases} 1 \text{ si } \| P(t) - \xi_o(t) \| \leq d, \\ 0 \text{ si } \| P(t) - \xi_o(t) \| > d. \end{cases} \tag{3.11}$$

Los campos vectoriales propuestos estan constituidos por un foco inestable, que gira en sentido contrario a las manecillas del reloj (Fig. 3.1), centrado en la posición del objeto con el que existe riesgo de colisión.

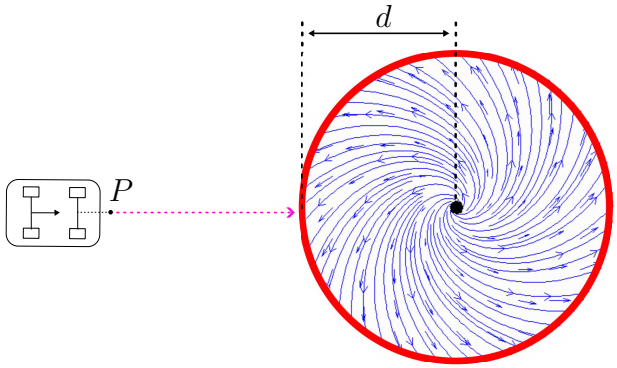


Figura 3.1: Campo vectorial repulsivo.

Para el análisis se asume que las posiciones iniciales del robot satisfacen $\| P(0) - \xi_o(0) \| > d$, es decir, no hay riesgo de colisión en el tiempo $t = 0$. La estrategia de control para el seguimiento de trayectorias con evasión de colisiones está dada por

$$\begin{bmatrix} v \\ w \end{bmatrix} = A(\theta, \phi)^{-1} (\lambda + \beta) . \tag{3.12}$$

El sistema en lazo cerrado (2.6) y (3.12) queda como

$$\dot{P} = -K \tanh(P - m) + \dot{m} + \beta, \tag{3.13}$$

por lo tanto la dinámica del error (3.7) ahora resulta en

$$\dot{e} = -K \tanh(e) + \beta. \tag{3.14}$$

Proposición 3.3 . *Considere el sistema* (2.6) *y la ley de control* (3.12). *Suponga que existe riesgo de colisión con un obstáculo en el instante t donde se satisface* $\epsilon > \frac{(k\sqrt{2}+\eta+\eta_o)}{d}$ *donde* η_o *es la velocidad máxima de un obstáculo en movimiento. Entonces, en el sistema en lazo cerrado* (2.6)-(3.12) *el AutoMiny mantiene una distancia mayor o igual a d,* $\forall t \geq 0$*, es decir evade el obstáculo.*

Demostración: Para mostrar que el AutoMiny evita colisionar con el obstáculo y mantiene una distancia mínima de él, se define una superficie dada por

$$\begin{aligned} \sigma_{ro} &= (p_x - x_o)^2 + (p_y - y_o)^2 - d^2 = 0, \\ \sigma_{ro} &= X_{ro}^2 + Y_{ro}^2 - d^2 = 0. \end{aligned} \tag{3.15}$$

La derivada de la diferencia de las velocidades del AutoMiny y el obstáculo,

cuando el primero se encuentra en peligro de colisión, es la siguiente:

$$\begin{aligned}\begin{bmatrix} \dot{X}_{ro} \\ \dot{Y}_{ro} \end{bmatrix} &= \begin{bmatrix} \dot{p}_x - \dot{x}_o \\ \dot{p}_y - \dot{y}_o \end{bmatrix}, \\ \begin{bmatrix} \dot{X}_{ro} \\ \dot{Y}_{ro} \end{bmatrix} &= -K\tanh(P-m) + \dot{m} + \epsilon \begin{bmatrix} (p_x - x_o) - (p_y - y_o) \\ (p_x - x_o) + (p_y - y_o) \end{bmatrix} - \dot{\xi}_o. \end{aligned} \tag{3.16}$$

Para determinar el comportamiento del AutoMiny, bajo la acción de los campos vectoriales repulsivos, se propone la siguiente función candidata de Lyapunov

$$V = \frac{1}{2}\sigma_{ro}^2. \tag{3.17}$$

La derivada con respecto del tiempo de la función está dada por $\dot{V} = \sigma_{ro}\dot{\sigma}_{ro}$. Por lo tanto, $\dot{V} \leq 0$ se logra si $\sigma_{ro}\dot{\sigma}_{ro} \leq 0$. A fin de presentar las condiciones en las cuales $\dot{V} < 0$ se considera que cuando existe riesgo de colisión, $(p_x - x_o), (p_y - y_o)$ se encuentra en la región interior de σ_{ro}, de modo que $\sigma_{ro} \leq 0$, de modo que el análisis para encontrar la ganancia de repulsión ϵ se reduce a determinar el valor de la ganancia para el cual $\dot{\sigma}_{ro} \geq 0$.

Entonces:

$$\begin{aligned}\dot{\sigma}_{ro} &= 2(p_x - x_o)(\dot{p}_x - \dot{x}_o) + 2(p_y - y_o)(\dot{p}_y - \dot{y}_o) \\ &= 2\begin{bmatrix} X_{ro} & Y_{ro} \end{bmatrix}\begin{bmatrix} \dot{X}_{ro} \\ \dot{Y}_{ro} \end{bmatrix}. \end{aligned} \tag{3.18}$$

Sustituyendo (3.16) en (3.18) obtenemos

$$\dot{\sigma}_{ro} = 2\begin{bmatrix} X_{ro} & Y_{ro} \end{bmatrix}\left[-K\tanh(P-m) + \dot{m} + \epsilon \begin{bmatrix} (p_x - x_o) - (p_y - y_o) \\ (p_x - x_o) + (p_y - y_o) \end{bmatrix} - \dot{\xi}_o\right] \tag{3.19}$$

$$\dot{\sigma}_{ro} = 2\begin{bmatrix} X_{ro} & Y_{ro} \end{bmatrix}[-K\tanh(P-m) + \dot{m}] + 2\epsilon(X_{ro}^2 + Y_{ro}^2) - 2\begin{bmatrix} X_{ro} & Y_{ro} \end{bmatrix}\dot{\xi}_o. \tag{3.20}$$

Para expresar la ecuación (3.20) en terminos escalares, se hace uso de la definición del producto punto (1.5) y norma de un vector (1.7). Para este análisis se considera el peor escenario de colisión, el cual se presenta cuando el AutoMiny y el obstáculo van directamente uno hacia el otro, como se muestra en la Fig.(3.2). Bajo estas consideraciones se obtiene:

$$\dot{\sigma}_{ro} = \left(-2\sqrt{X_{ro}^2 + Y_{ro}^2}\right)(k\sqrt{2} + \eta + \eta_o) + 2\epsilon\left(X_{ro}^2 + Y_{ro}^2\right) > 0. \tag{3.21}$$

Resolviendo para ϵ se tiene que,

$$\epsilon > \frac{2\sqrt{X_{ro}^2 + Y_{ro}^2}(k\sqrt{2} + \eta + \eta_o)}{2\left(X_{ro}^2 + Y_{ro}^2\right)} = \frac{(k\sqrt{2} + \eta + \eta_o)}{d}, \tag{3.22}$$

entonces $\dot{\sigma}_{ro} > 0$. Esto implica que el robot se alejará del objeto hasta una distancia d. Como $\parallel P(0) - \xi_o(0) \parallel \geq d$, entonces el robot no solo evita la colisión sino que también $\parallel P(t) - \xi_o(t) \parallel \geq d \ \forall t \geq 0$. Como caso particular tenemos que si la velocidad máxima del obstáculo es $\eta_o = 0$, es decir el obstáculo es fijo, entonces

$$\epsilon > \frac{(k\sqrt{2} + \eta)}{d}. \tag{3.23}$$

Para validar los resultados obtenidos anteriormente se realiza la simulación numérica y experimentación.

Resultados de simulación numérica

Antes de realizar experimentos con el AutoMiny se realiza la simulación numérica cuando se tiene un obstáculo fijo. Utilizando la ecuación (3.23) obtenida anteriormente.

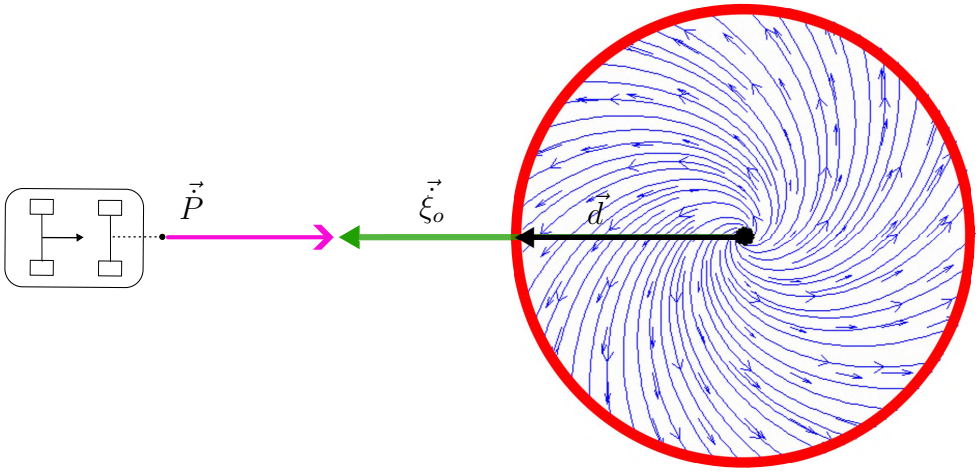


Figura 3.2: Peor escenario de colisión entre el AutoMiny y un obstáculo

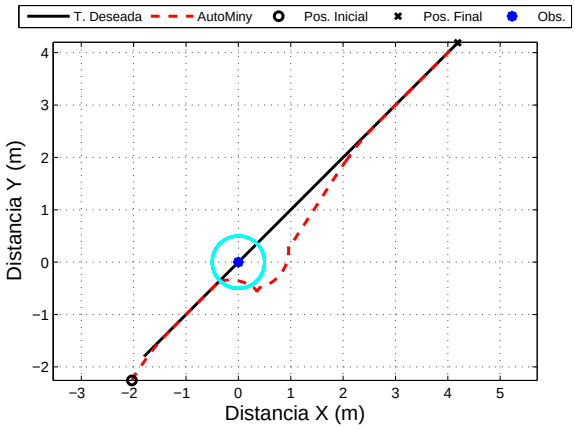
La simulación numérica se realizó utilizando la estrategia de control (3.12), (3.23) y los parámetros reales del robot. El punto P a controlar se encuentra a una distancia $\Delta = 0.1m$ del punto medio del eje de las ruedas delanteras, la distancia entre los ejes trasero y delantero es $\ell = 0.26m$, las ganancias de control y repulsión fueron seleccionadas $k_x, k_y = 1$ y $\epsilon = 1.2(k\sqrt{2} + \eta)/d = 3.7335$. El obstáculo tiene coordenas $\xi_o = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ y la distancia mínima donde entra en acción el campo vectorial repulsivo es $d = 0.5m$. Se agregó la restricción física del ángulo ϕ del AutoMiny, la cual tiene un valor de ± 0.37 radianes.

La trayectoria deseada es la recta definida por:

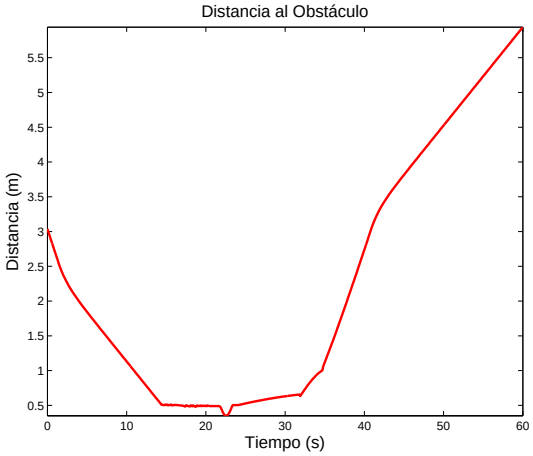
$$m(t) = \begin{bmatrix} 0.1t - 1.8, & 0.1t - 1.8 \end{bmatrix}^T, \tag{3.24}$$

aquí la velocidad máxima de la trayectoria es $\eta = 0.14142$. Los resultados se presentan a continuación. En la Fig. 3.3a se muestra la trayectoria descrita por el robot. El robot sigue la trayectoria, cuando se encuentra a $0.5m$ del

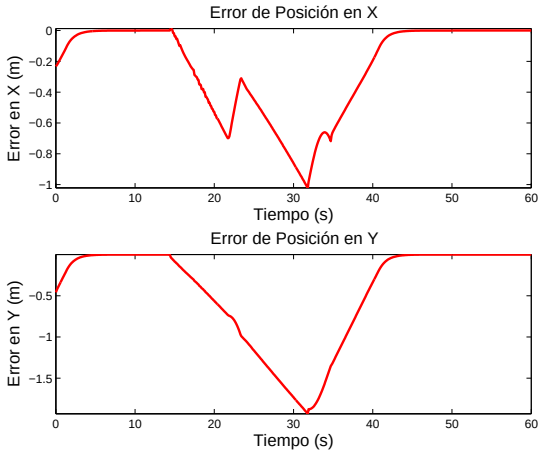
obstáculo entra en funcionamiento el campo vectorial repulsivo expulsando al robot de la zona de colisión. El AutoMiny entra en la zona de colisión, debido a las restricciónes holonomicas del robot, lo cual se puede observar en la Fig. 3.3b. En esta figura se muestra la distancia del punto a controlar P con respecto a la posición del obstáculo. En la Fig. 3.3c se muestran los errores de posición del robot en los ejes coordenados del plano. Se puede observar que entre los segundos 15-45, el robot, se incrementa el error debido a la evasión del obstáculo, sin embargo, vuelve a retomar la trayectoria establecida una vez que logra evadirlo. En la Fig. 3.3d se muestran los ángulos de orientación del robot y de dirección de las ruedas. Por último se muestran las señales de control en la Fig. 3.3e.



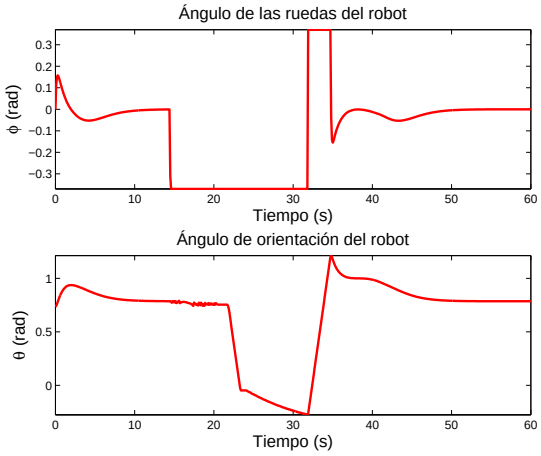
(a) Trayectoria del robot en el plano.



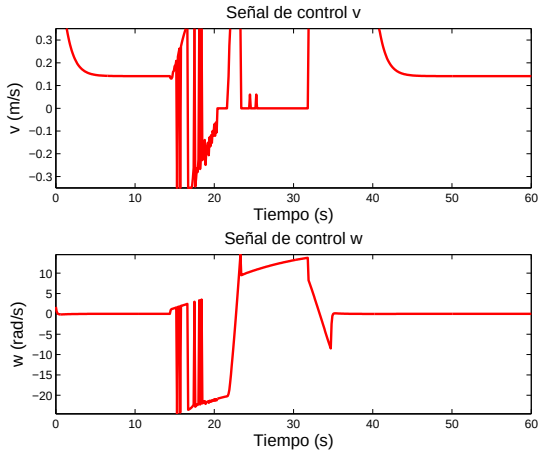
(b) Distancia entre el punto P y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del robot.



(e) Señales de control.

Figura 3.3: Simulación numérica

3.2.2. Análisis de restricción de dirección del robot

Debido a las restricciones no-holonómicas del AutoMiny, este llega a entrar a la zona de repulsión del obstáculo. Con la finalidad de evitar que el AutoMiny se encuentre a una distancia menor a la permitida (d) se realiza el siguiente análisis geométrico.

En la Fig. 3.4 se muestra al AutoMiny en dirección a colisionar con un obstáculo fijo. Se definen ahora dos distancias: d la distancia mínima permitida y d_m la distancia mínima a la cual el AutoMiny debe girar para evitar entrar en la zona de restricción. El análisis consiste en determinar d_m.

$$\begin{aligned} d_r &= \alpha\cos(\phi) \\ \alpha &= \frac{\ell}{\sin(\phi)} \\ \rho &= \sqrt{\alpha^2+\Delta^2} \\ \rho &= \sqrt{\frac{\ell^2}{\sin(\phi)^2}+\Delta^2} \\ H_r &= d+\rho \\ C_r &= \sqrt{H_r^2-d_r^2} \end{aligned}$$

Ahora es posible determinar d_m,

$$d_m = |C_r - (\ell+\Delta)|. \tag{3.25}$$

El valor de d_m utilizado para este caso es $d_m = 0.666m$, tomando en cuenta el ángulo mínimo de giro, $\phi = -\frac{\pi}{2}$, los parámetros reales del AutoMiny, $\ell = 0.26m, \Delta = 0.1m$ y considerando una distancia mínima $d = 0.5m$.

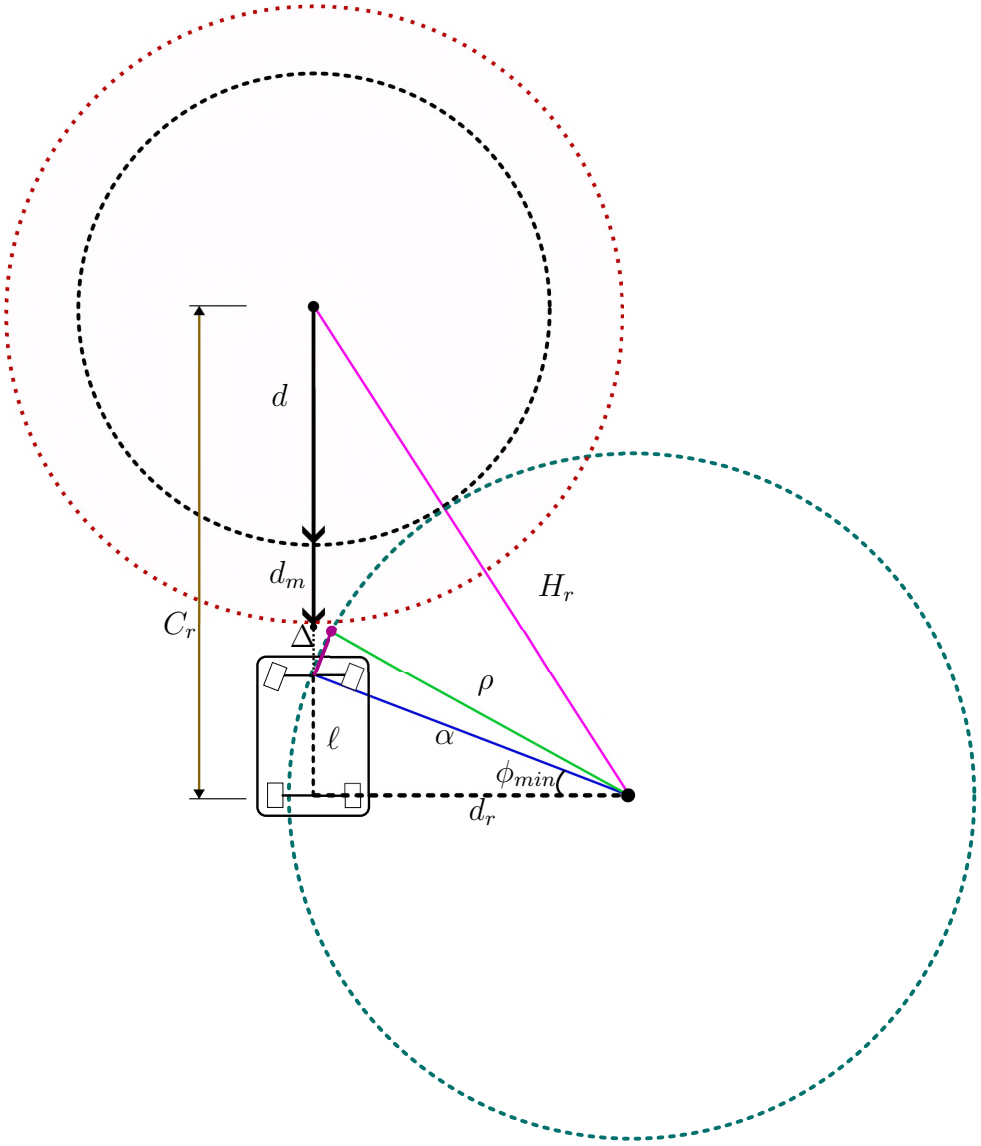


Figura 3.4: Esquema de restricción en el ángulo de dirección del robot.

Resultados de simulación

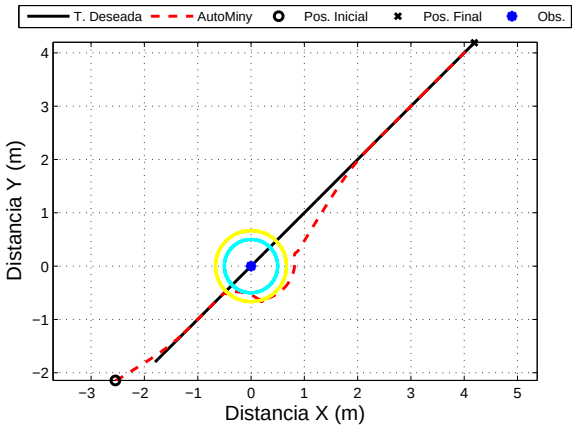
La simulación numérica se realizó utilizando dos trayectorias distintas. Una recta y un círculo. Los parámetros de ambas trayectorias se muestran en la siguiente tabla.

Parámetros de las Trayectorias					
	Ecuación	vel. máx. η	k_x	k_y	T
Línea recta	$m(t) = \begin{bmatrix} 0.1t - 1.8, & 0.1t - 1.8 \end{bmatrix}^T$	0.14142 m/s	1	1	60 s
Círculo	$m(t) = \begin{bmatrix} 0.5 + 1.2cos\left(\frac{2\pi t}{T}\right), 1.2sin\left(\frac{2\pi t}{T}\right) \end{bmatrix}^T$	0.1257 m/s	1	1	60 s

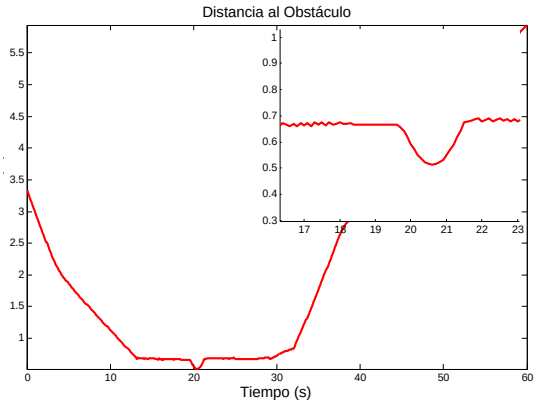
Tabla 3.1: Trayectorias deseadas

1. Para la recta el valor de ϵ es, $\epsilon = 1.2\frac{(k\sqrt{2}+\eta)}{d_m} = 2.80294$ y el obstáculo tiene coordenadas $\xi_o = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$. En la Fig.3.5a se observa que el robot no entra a la zona mínima de colisión lo que asegura que pueda mantenerse a la distancia mínima d designada. Esto es más evidente en la Fig.(3.5b).

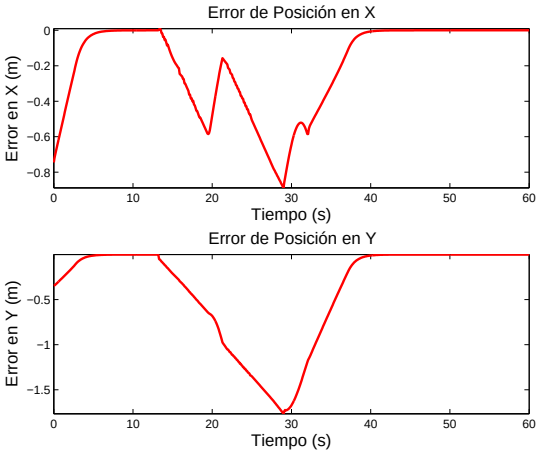
2. Para la trayectoria circular el obstáculo tiene coordenas $\xi_o = \begin{bmatrix} -0.7 & 0 \end{bmatrix}^T$ y la ganancia de repulsión es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta)}{d_m} = 2.774619$. Los resultados se presentan en la Fig. 3.6.



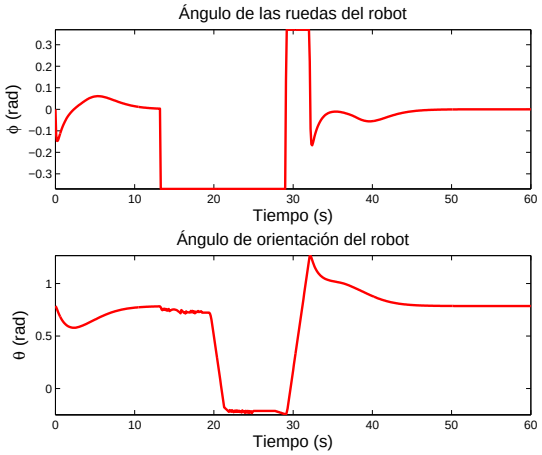
(a) Trayectoria del robot en el plano.



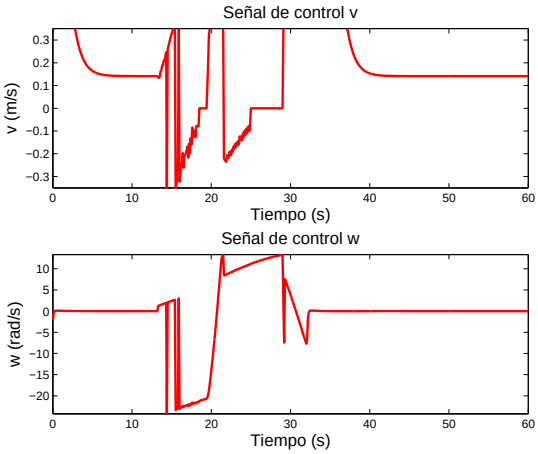
(b) Distancia entre el punto P y el obstáculo.



(c) Errores de posición del robot.

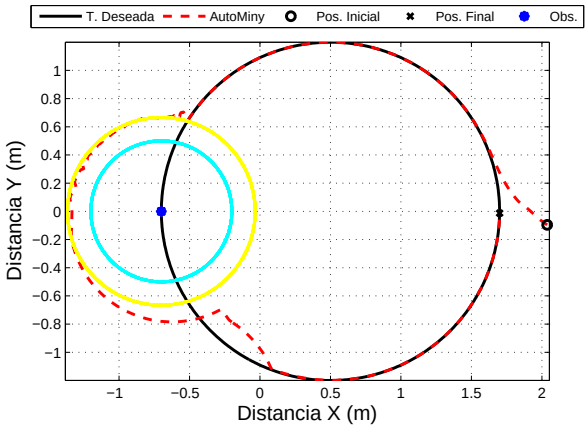


(d) Ángulos de orientación y dirección del robot.

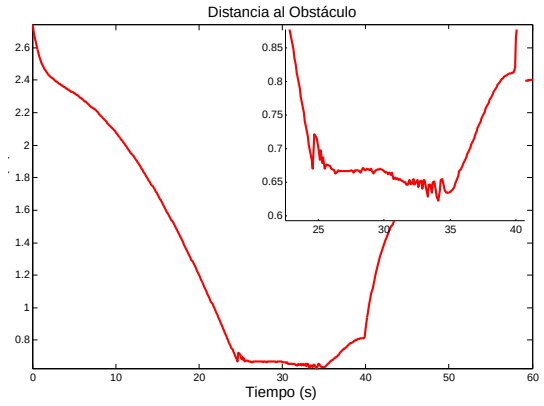


(e) Señales de control.

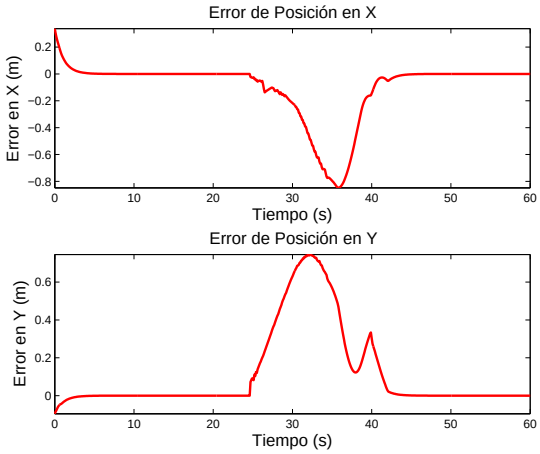
Figura 3.5: Simulación numérica trayectoria recta obstáculo fijo



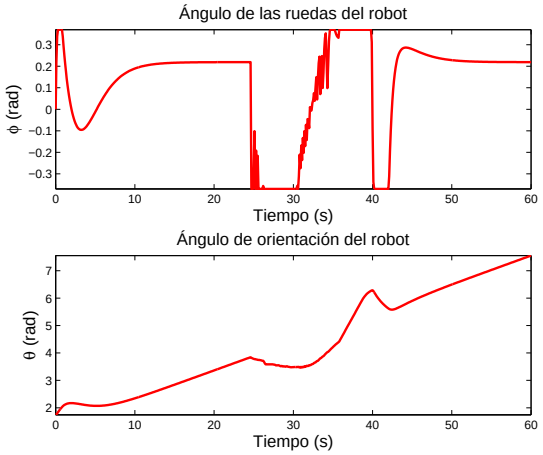
(a) Trayectoria del robot en el plano.



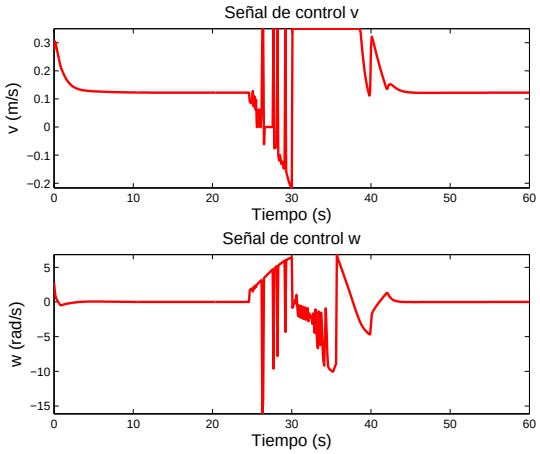
(b) Distancia entre el punto P y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del robot.



(e) Señales de control.

Figura 3.6: Simulación numérica trayectoria circular obstáculo fijo

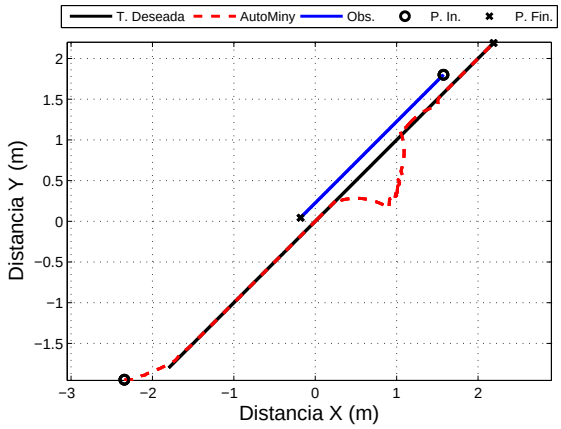
3.2 Estrategia para la evasión de colisiones

Una vez que se ha validado, mediante simulación numérica, la evasión de colisiones contra un obstáculo fijo, se extienden estos resultados al caso cuando el AutoMiny se encuentra con un obstáculo en movimiento durante el seguimiento de su trayectoria. La trayectorias deseadas se describen en la Tabla 3.1.

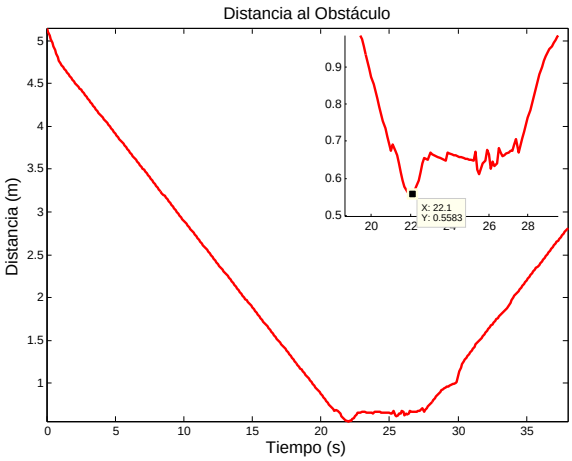
Resultados de Simulación

1. Trayectoria recta: Para la simulaciíon numérica, la velocidad máxima del obstáculo está limitada a $\eta_o = 0.0622$. Además, el valor de la ganancia de repulsión es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta+\eta_o)}{d_m} = 2.91501$. En la Fig.3.7a se observa la trayectoria que sigue el AutoMiny y el obstáculo en movimiento. En las Fig. 3.8 se muestra el momento en que el AutoMiny evade el obstáculo.

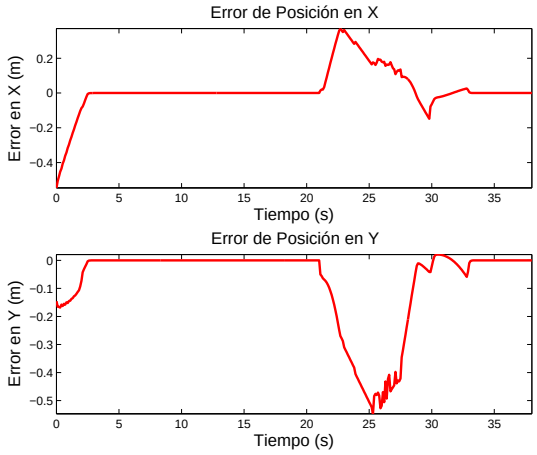
2. Trayectoria circular: Al igual que para el caso anterior, la velocidad máxima del obstáculo es $\eta_o = 0.0622$. La ganancia de repulsión para la trayectoria circular es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta+\eta_o)}{d_m} = 2.88669$. Los resultados se presentan en la Fig.3.9 y 3.10.



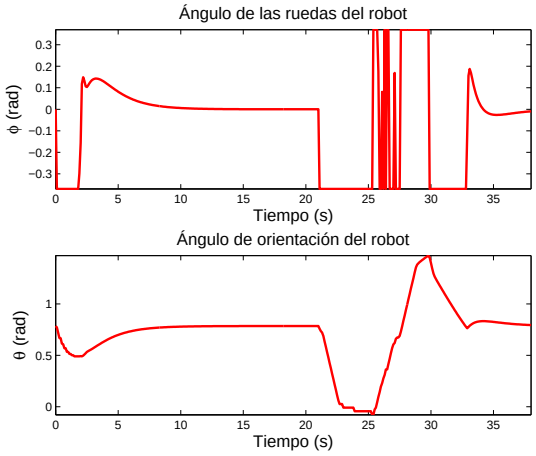
(a) Trayectoria del robot en el plano.



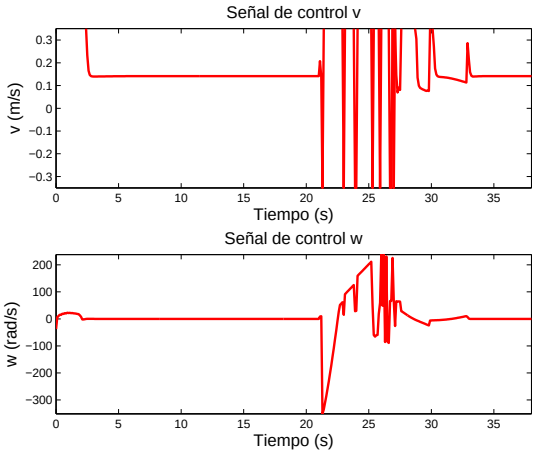
(b) Distancia entre el punto P y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del robot.



(e) Señales de control.

Figura 3.7: Simulación numérica trayectoria recta, obs. en movimiento

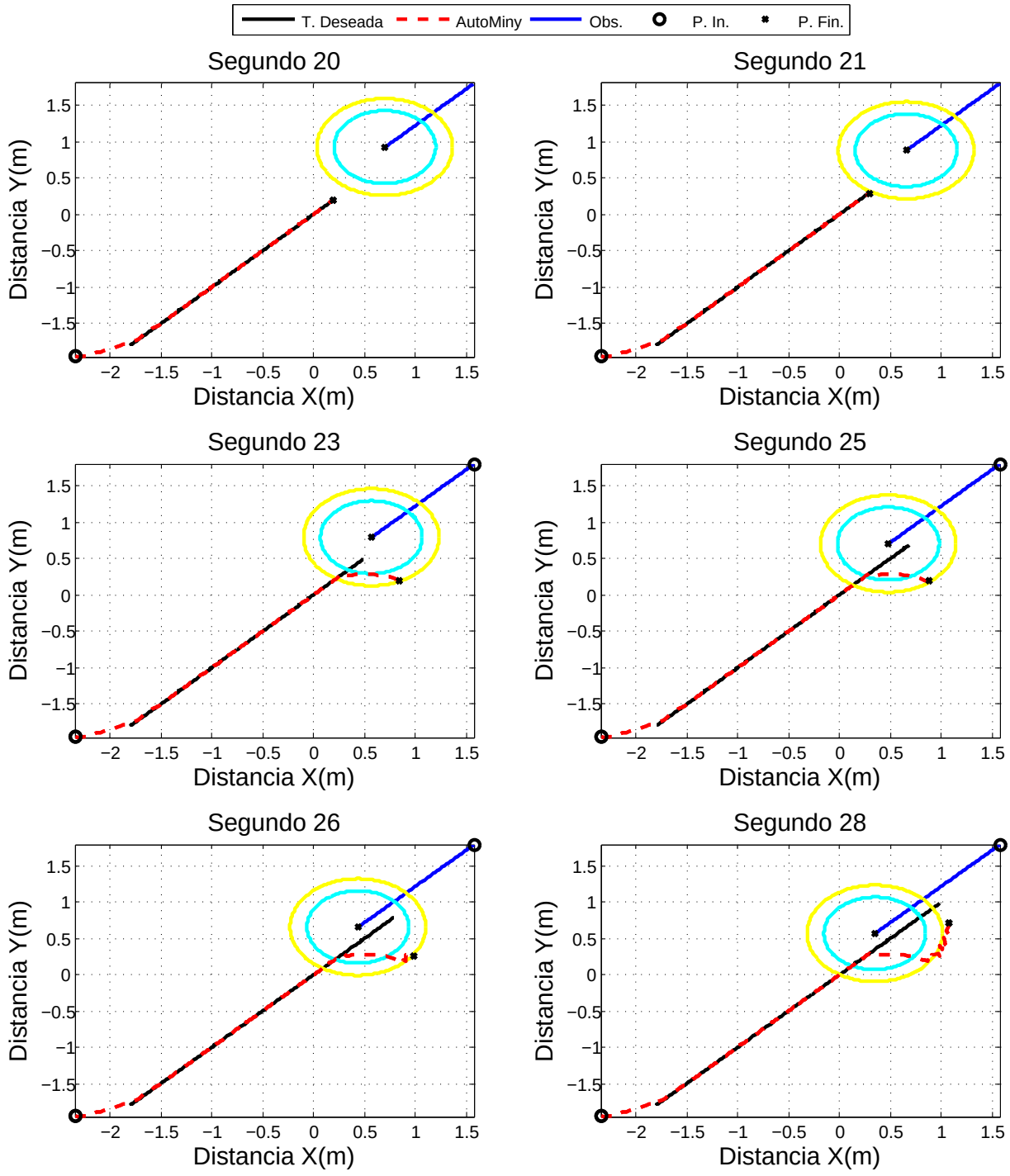
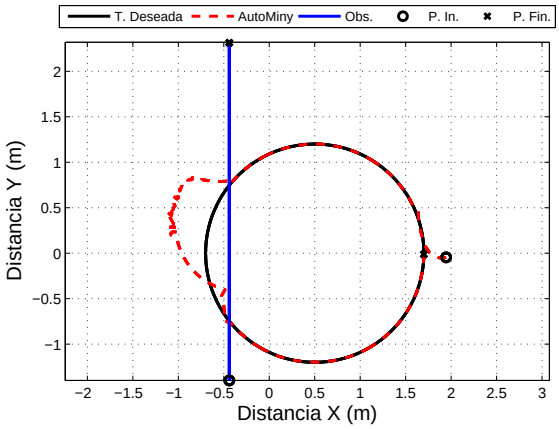
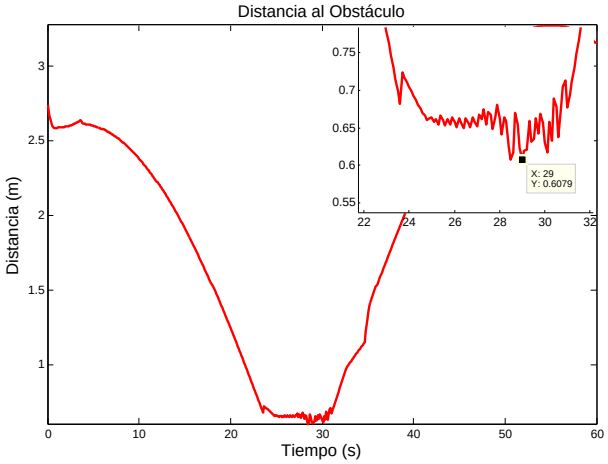


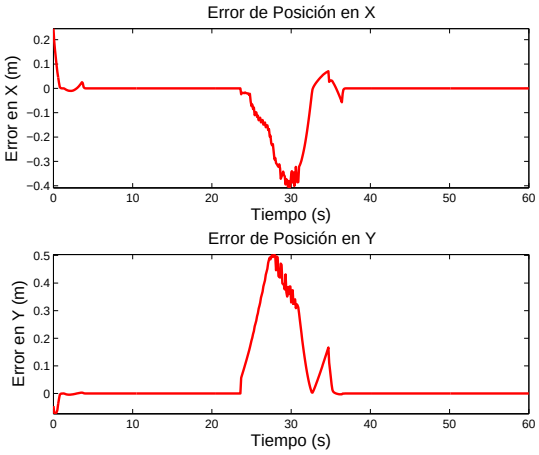
Figura 3.8: Evasión del obs. en movimiento en diferente tiempo, trayectoria recta



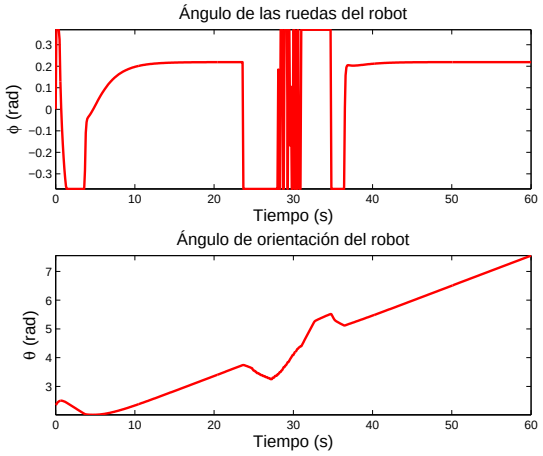
(a) Trayectoria del robot en el plano.



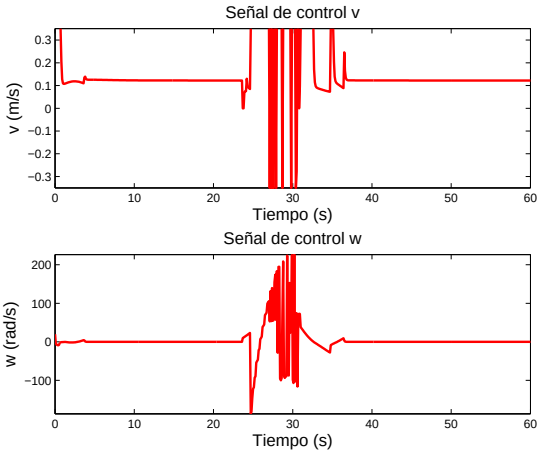
(b) Distancia entre el punto P y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del robot.



(e) Señales de control.

Figura 3.9: Simulación numérica trayectoria circular, obs. en movimiento

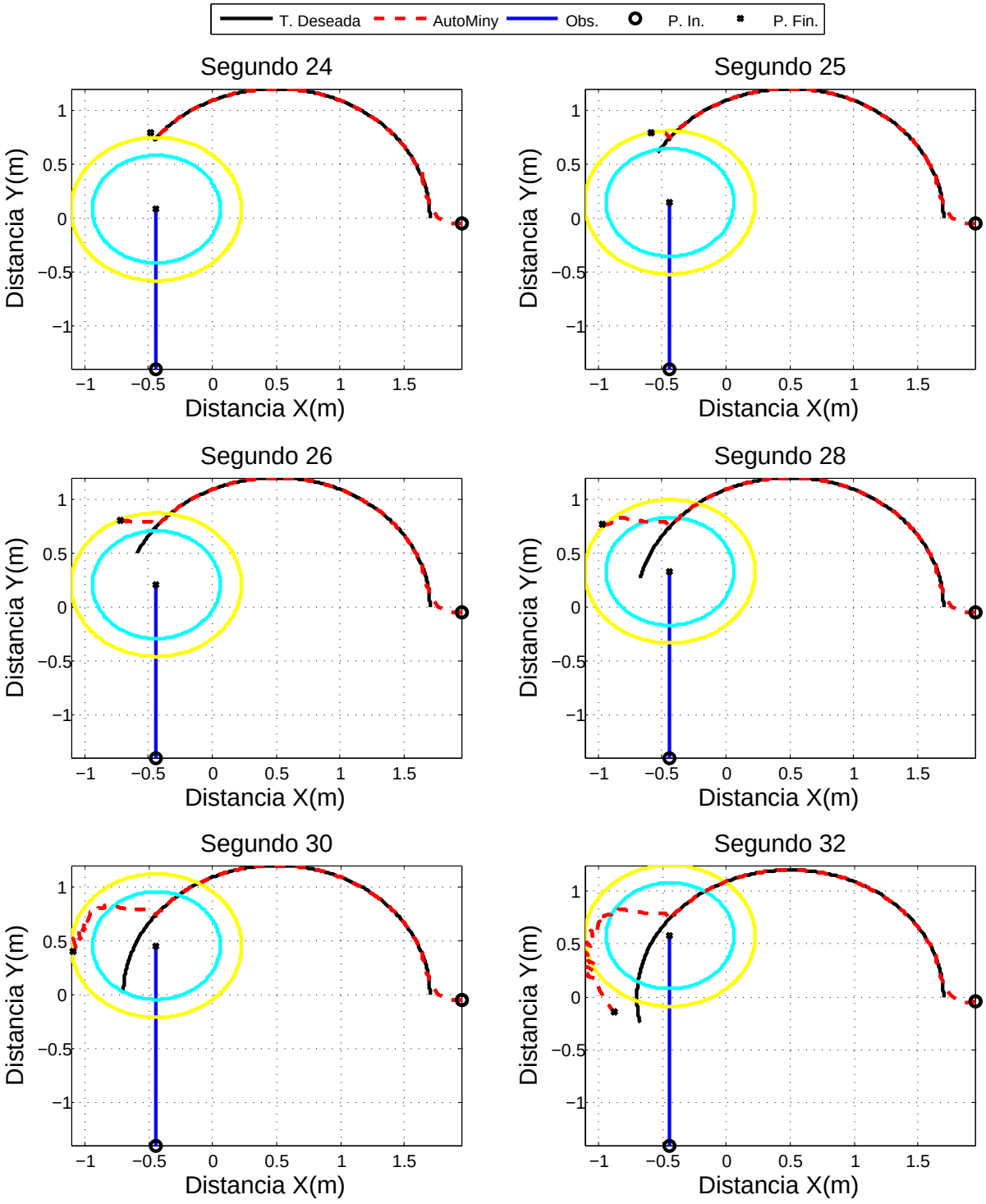


Figura 3.10: Evasión del obs. en movimiento en diferente tiempo, trayectoria circular

3.2.3. Evasión de colisiones contra n obstáculos en movimiento al mismo tiempo

Se procede ahora a analizar la evasión de colisiones con varios obstáculos al mismo tiempo. Para ello primeramente se realiza con dos obstáculos para su posterior extención a 'n' obstáculos. Con este fin se modifica la ley de control

$$\begin{bmatrix} v \\ w \end{bmatrix} = A(\theta,\phi)^{-1}\left(\lambda + \beta_1 + \beta_2\right). \tag{3.26}$$

Proposición 4. Considere el sistema (2.6) y la ley de control (3.26). Suponga que existe riesgo de colisión con dos obstáculos en el instante t, y se satisface $\epsilon > \frac{(k\sqrt{2}+\eta+\eta_o)}{2d}$ donde $\eta_o = max(\eta_{o1}, \eta_{o2})$ y $\epsilon = max(\epsilon_1, \epsilon_2)$. Entonces, en el sistema en lazo cerrado (2.6)-(3.26) el AutoMiny logra mantener una distancia mayor o igual a d, $\forall t \geq 0$ a ambos obstáculos.

Demostración: Para demostrár la proposición anterior se define el vector de superficies:

$$\sigma = \begin{bmatrix} \sigma_{ro1} \\ \sigma_{ro2} \end{bmatrix}, \tag{3.27}$$

donde

$$\begin{aligned} \sigma_{ro1} &= (p_x - x_{o1})^2 + (p_y - y_{o1})^2 - d^2 = 0, \\ \sigma_{ro1} &= X_{ro1}^2 + Y_{ro1}^2 - d^2 = 0, \qquad (3.28) \\ \sigma_{ro2} &= (p_x - x_{o2})^2 + (p_y - y_{o2})^2 - d^2 = 0, \\ \sigma_{ro2} &= X_{ro2}^2 + Y_{ro2}^2 - d^2 = 0. \qquad (3.29) \end{aligned}$$

La derivada de la diferencia de velocidades entre el AutoMiny y los obstáculos, cuando el primero se encuentra en peligro de colisión con ambos

obstáculos a la vez, está definida por:

$$\begin{bmatrix} \dot{X}_{ro1} \\ \dot{Y}_{ro1} \end{bmatrix} = \begin{bmatrix} \dot{p_x} - \dot{x}_{o1} \\ \dot{p_y} - \dot{y}_{o1} \end{bmatrix} = -K\tanh(P-m) + \dot{m} + \epsilon_1 \begin{bmatrix} (p_x - x_{o1}) - (p_y - y_{o1}) \\ (p_x - x_{o1}) + (p_y - y_{o1}) \end{bmatrix} + ... \\ ... + \epsilon_2 \begin{bmatrix} (p_x - x_{o2}) - (p_y - y_{o2}) \\ (p_x - x_{o2}) + (p_y - y_{o2}) \end{bmatrix} - \dot{\xi}_{o1}. \tag{3.30}$$

$$\begin{bmatrix} \dot{X}_{ro2} \\ \dot{Y}_{ro2} \end{bmatrix} = \begin{bmatrix} \dot{p_x} - \dot{x}_{o2} \\ \dot{p_y} - \dot{y}_{o2} \end{bmatrix} = -K\tanh(P-m) + \dot{m} + \epsilon_1 \begin{bmatrix} (p_x - x_{o1}) - (p_y - y_{o1}) \\ (p_x - x_{o1}) + (p_y - y_{o1}) \end{bmatrix} + ... \\ ... + \epsilon_2 \begin{bmatrix} (p_x - x_{o2}) - (p_y - y_{o2}) \\ (p_x - x_{o2}) + (p_y - y_{o2}) \end{bmatrix} - \dot{\xi}_{o2}. \tag{3.31}$$

Para determinar el comportamiento del AutoMiny bajo la acción de los campos vectoriales repulsivos proponemos una función de Lyapunov dada por

$$V = \frac{1}{2}\sigma^T\sigma, \tag{3.32}$$

cuya derivada con respecto al tiempo está dada por $\dot{V} = \sigma^T\dot{\sigma} \leq \sigma^*(\dot{\sigma}_{ro1} + \dot{\sigma}_{ro2}) < 0$ donde $\sigma^* = max(\sigma_{ro1}, \sigma_{ro2})$. Considerando que, cuando existe riesgo de colisión, la trayectoria del AutoMiny se encuentra en la región interior de σ, de modo que $\sigma_{ro1}, \sigma_{ro2} \leq 0$, entonces el análisis para encontrar la ganancia de repulsión ϵ se reduce a determinar el valor de dicha ganancia para el cual $\dot{\sigma}_{ro1} + \dot{\sigma}_{ro2} > 0$. Entonces

$$\begin{aligned} \dot{\sigma}_{ro1} + \dot{\sigma}_{ro2} &= 2X_{ro1}\dot{X}_{ro1} + 2Y_{ro1}\dot{Y}_{ro1} + 2X_{ro2}\dot{X}_{ro2} + 2Y_{ro2}\dot{Y}_{ro2} \\ &= 2\begin{bmatrix} X_{ro1} & Y_{ro1} \end{bmatrix}\begin{bmatrix} \dot{X}_{ro1} \\ \dot{Y}_{ro1} \end{bmatrix} + 2\begin{bmatrix} X_{ro2} & Y_{ro2} \end{bmatrix}\begin{bmatrix} \dot{X}_{ro2} \\ \dot{Y}_{ro2} \end{bmatrix} \end{aligned} \tag{3.33}$$

Sustituyendo (3.30), (3.31) en (3.33) obtenemos:

$$\dot{\sigma}_{ro1} + \dot{\sigma}_{ro2} = 2\begin{bmatrix} X_{ro1} & Y_{ro1} \end{bmatrix}[-K\tanh(P-m) + \dot{m}] + 2\epsilon_1(X_{ro1}^2 + Y_{ro1}^2) + ... \\ ... + 2(X_{ro1}X_{ro2} + Y_{ro1}Y_{ro2})(\epsilon_1 + \epsilon_2) + 2(X_{ro1}Y_{ro2} - X_{ro2}Y_{ro1})(\epsilon_1 - \epsilon_2) - ... \\ ... - 2\begin{bmatrix} X_{ro1} & Y_{ro1} \end{bmatrix}\dot{\xi}_{o1} + 2\begin{bmatrix} X_{ro2} & Y_{ro2} \end{bmatrix}[-K\tanh(P-m) + \dot{m}] + ... \\ ... + 2\epsilon_2(X_{ro2}^2 + Y_{ro2}^2) - 2\begin{bmatrix} X_{ro2} & Y_{ro2} \end{bmatrix}\dot{\xi}_{o2}. \tag{3.34}$$

$$\begin{aligned}\dot{\sigma}_{ro1}+\dot{\sigma}_{ro2} = 2\begin{bmatrix} X_{ro1} & Y_{ro1}\end{bmatrix}[-K\tanh(P-m)+\dot{m}] + 2\epsilon_1(X_{ro1}^2+Y_{ro1}^2)+...\\ ...+2\begin{bmatrix} X_{ro1} & Y_{ro1}\end{bmatrix}\begin{bmatrix} X_{ro2}\\ Y_{ro2}\end{bmatrix}(\epsilon_1+\epsilon_2)+2\begin{bmatrix} X_{ro1} & Y_{ro1}\end{bmatrix}\begin{bmatrix} Y_{ro2}\\ -X_{ro2}\end{bmatrix}(\epsilon_1-\epsilon_2)-...\\ ...-2\begin{bmatrix} X_{ro1} & Y_{ro1}\end{bmatrix}\dot{\xi}_{o1}+2\begin{bmatrix} X_{ro2} & Y_{ro2}\end{bmatrix}[-K\tanh(P-m)+\dot{m}]+...\\ ...+2\epsilon_2(X_{ro2}^2+Y_{ro2}^2)-2\begin{bmatrix} X_{ro2} & Y_{ro2}\end{bmatrix}\dot{\xi}_{o2}. \end{aligned} \tag{3.35}$$

Considerando el peor escenario para cada uno de los obstáculos con respecto al AutoMiny (ver Fig.(3.2)), al igual que en el análisis del capítulo 3 y la definición del producto punto obtenemos:

$$\begin{aligned}\dot{\sigma}_{ro1}+\dot{\sigma}_{ro2} = &-2\sqrt{X_{ro1}^2+Y_{ro1}^2}\left(k\sqrt{2}+\eta\right)+2\epsilon_1(X_{ro1}^2+Y_{ro1}^2)+...\\ &...+2\sqrt{X_{ro1}^2+Y_{ro1}^2}\sqrt{X_{ro2}^2+Y_{ro2}^2}\,(\epsilon_1+\epsilon_2)-...\\ &...-2\sqrt{X_{ro1}^2+Y_{ro1}^2}\eta_{o1}-2\sqrt{X_{ro2}^2+Y_{ro2}^2}\left(k\sqrt{2}+\eta\right)+...\\ &...+2\epsilon_2(X_{ro2}^2+Y_{ro2}^2)-2\sqrt{X_{ro2}^2+Y_{ro2}^2}\eta_{o2}>0. \end{aligned} \tag{3.36}$$

por lo que

$$\begin{aligned}\dot{\sigma}_{ro1}+\dot{\sigma}_{ro2} = &-2d\left(k\sqrt{2}+\eta\right)+2\epsilon_1 d^2+2d^2\,(\epsilon_1+\epsilon_2)-2d\eta_{o1}-...\\ &...-2d\left(k\sqrt{2}+\eta\right)+2\epsilon_2 d^2-2d\eta_{o2}>0, \end{aligned} \tag{3.37}$$

$$= -2d\left[2(k\sqrt{2}+\eta)+\eta_{o1}+\eta_{o2}\right]+4d^2\,(\epsilon_1+\epsilon_2)>0. \tag{3.38}$$

Seleccionando el valor máximo entre las velocidades de los obstáculos y tomando el valor máximo entre las ganancias de repulsión obtenemos lo siguiente

$$\begin{aligned} -2d\left[2(k\sqrt{2}+\eta+\eta_o)\right]+4d^2(2\epsilon) &> 0,\\ 8d^2\epsilon &> 4d\left[(k\sqrt{2}+\eta+\eta_o)\right],\\ \epsilon &> \frac{k\sqrt{2}+\eta+\eta_o}{2d}. \end{aligned} \tag{3.39}$$

Realizando el mismo procedimiento podemos extender el resultado a uno más general para n obstáculos.

$$\begin{bmatrix} v \\ w \end{bmatrix} = A(\theta,\phi)^{-1}\left(\lambda + \beta_1 + \beta_2 + ... + \beta_n\right). \tag{3.40}$$

Proposición 5. Considere el sistema (2.6) y la ley de control (3.40). Suponga que existe riesgo de colisión con n obstáculos en el instante t, y se satisface $\epsilon > \frac{(k\sqrt{2}+\eta+\eta_o)}{nd}$ donde $\eta_o = max(\eta_{o1}, \eta_{o2}, ..., \eta_{on})$ y $\epsilon = max(\epsilon_1, \epsilon_2, ..., \epsilon_n)$. Entonces, en el sistema en lazo cerrado (2.6)-(3.40) el AutoMiny logra mantener una distancia a todos los obstáculos mayor o igual a d, $\forall t \geq 0$.

Demostración: Definimos el vector de superficies:

$$\sigma = \begin{bmatrix} \sigma_{ro1} \\ \sigma_{ro2} \\ \vdots \\ \sigma_{ron} \end{bmatrix}, \tag{3.41}$$

donde

$$\sigma_{ro1} = X_{ro1}^2 + Y_{ro1}^2 - d^2 = 0, \tag{3.42}$$

$$\sigma_{ro2} = X_{ro2}^2 + Y_{ro2}^2 - d^2 = 0, \tag{3.43}$$

$$\vdots$$

$$\sigma_{ron} = X_{ron}^2 + Y_{ron}^2 - d^2 = 0. \tag{3.44}$$

Definimos de la siguiente manera las derivadas de la diferencia de velocidades entre el AutoMiny y cada uno de los obstáculos.

$$\begin{bmatrix} \dot{X}_{roi} \\ \dot{Y}_{roi} \end{bmatrix} = \begin{bmatrix} \dot{p_x} - \dot{x}_{oi} \\ \dot{p_y} - \dot{y}_{oi} \end{bmatrix} = -K \tanh(P - m) + \dot{m} + \dots$$
$$\dots + \sum_{k=1}^{n} \epsilon_k \begin{bmatrix} (p_x - x_{ok}) - (p_y - y_{ok}) \\ (p_x - x_{ok}) + (p_y - y_{ok}) \end{bmatrix} - \dot{\xi}_{oi}. \tag{3.45}$$

para $i = 1, 2, \cdots, n$. La función candidata de Lyapunov se propone en forma similar a (3.32), cuya derivada con respecto al tiempo está dada por $\dot{V} = \sigma^T \dot{\sigma} \leq \sigma^* (\sum_{i=1}^{n} \dot{\sigma}_{roi}) < 0$.

El valor de la ganancia de repulsión debe satisfacer que $\sum_{i=1}^{n} \dot{\sigma}_{roi} > 0$. Este valor puede obtenerse de la siguiente forma.

$$\sum_{i=1}^{n} \dot{\sigma}_{roi} = 2 \sum_{i=1}^{n} (X_{roi} \dot{X}_{roi} + Y_{roi} \dot{Y}_{roi}),$$
$$= 2 \sum_{i=1}^{n} [X_{roi} \; Y_{roi}] \begin{bmatrix} \dot{X}_{roi} \\ \dot{Y}_{roi} \end{bmatrix}. \tag{3.46}$$

Sustituyendo (3.45) en (3.46) obtenemos:

$$\sum_{i=1}^{n} \dot{\sigma}_{roi} = 2 \left\{ \sum_{i=1}^{n} ([X_{roi} \; Y_{roi}] \left[-K tanh(P - m) + \dot{m} \right] + \dots \right.$$
$$\dots + \epsilon_i \left(X_{roi}^2 + Y_{roi}^2 \right) - [X_{roi} \; Y_{roi}] \dot{\xi}_{oi}) + \dots$$
$$\dots + \sum_{i=1}^{n-1} [X_{roi} \; Y_{roi}] \sum_{j=i+1}^{n} \left(\begin{bmatrix} X_{roj} \\ Y_{roj} \end{bmatrix} (\epsilon_i + \epsilon_j) + \dots \right.$$
$$\left. \left. \dots + \begin{bmatrix} X_{roj} \\ -Y_{roj} \end{bmatrix} (\epsilon_i - \epsilon_j) \right) \right\}, \tag{3.47}$$

por lo que

$$\sum_{i=1}^{n} \dot{\sigma}_{roi} = -2nd(k\sqrt{2} + \eta) + 2d^2 \sum_{i=1}^{n} \left(n\epsilon_i - \frac{\eta_{oi}}{d} \right) > 0. \tag{3.48}$$

Se puede garantizar que, al seleccionar el valor máximo entre las velocidades de los obstáculos y el valor máximo entre las ganancias de repulsión, la forma general para el valor ϵ con n obstáculos está descrita por

$$\epsilon > \frac{k\sqrt{2} + \eta + \eta_o}{nd}. \tag{3.49}$$

Observación 1. El valor de ϵ que garantiza la evasión de los obstáculos está sujeto tanto a las restricciones no holónomas del AutoMiny como a las del área de trabajo.

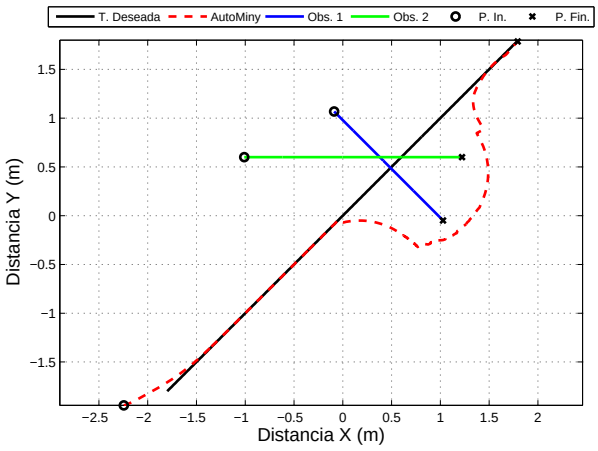
Resultados de Simulación

La simulación para la extención a más de un obstáculo se realizó usando dos trayectorias, la recta y una circunferencia descritas en la Tabla 3.1.

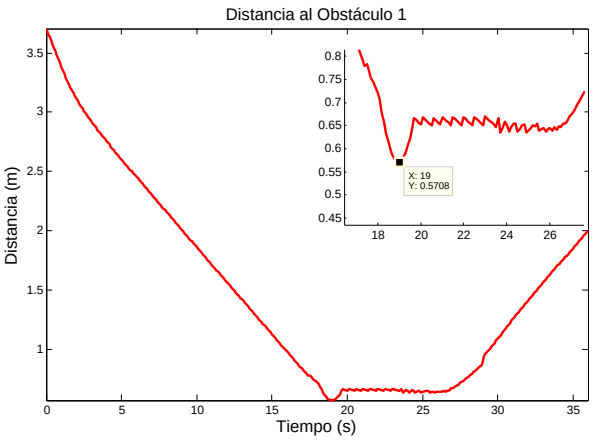
1. Trayectoria recta: La velocidad máxima de los obstáculos 1 y 2 son $\eta_{o1} = 0.0622m/s$ y $\eta_{o2} = 0.044m/s$ respectivamente. La ganancia de repulsión es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta+\eta_o)}{2d_m} = 1.457507714$. Es importante mencionar que la ganancia de repulsión anterior entra en acción sólo cuando el robot detecta a ambos obstáculos al mismo tiempo, en cualquier otro instante que sólo esté en peligro de colisionar con un sólo obstáculo, la ganancia de repulsión será $\epsilon = 1.2\frac{(k\sqrt{2}+\eta+\eta_{on})}{d_m}$, según corresponda a un obsáculo o a otro.

   En la Figura 3.11a se observan las trayectorias deseadas tanto para el robot como los obstáculos. En las Figuras 3.11b, 3.11c se grafican las distancias entre el robot y los obstáculos. En estas figuras se puede observar que entre los segundos 16 y el 20, el robot se encuentra en la zona de repulsión de ambos obstáculos al mismo tiempo. Lo anterior se puede observar con mayor detalle en la Figura 3.12, por lo que se interpreta que es en este instante es cuando entra en acción la ganancia de repulsión calculada. También se observa como el AutoMiny entra a la región amarilla debido a la restricción de giro de sus ruedas delanteras pero no entra en la zona de colisión, por lo que el análisis realizado es válido. El robot sigue de forma aceptable la trayectoria propuesta, como consecuencia los errores convergen a cero cuando el robot no se encuentra en riesgo de colisionar.

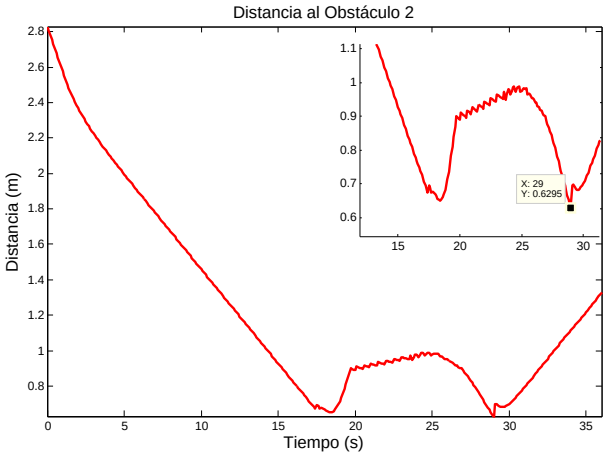
2. Trayectoria circular: La velocidad máxima de los obstáculos 1 y 2 son $\eta_{o1} = 0.044m/s$ y $\eta_{o2} = 0.0622m/s$ respectivamente. La ganancia de repulsión es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta+\eta_o)}{2d_m} = 1.443345$. Los resultados son presentados en las Figuras 3.13 y 3.14. Podemos observar, en la Figura 3.14, que es entre los segundos 25 al 28 que el robot se encuentra dentro de la zona amarilla de ambos obstáculos, por lo que es en este intervalo de tiempo en donde entra en acción la ley de control (3.26) junto con el valor de ϵ. Se observa también que el robot logra evadir los obstáculos y mantenerse a la distancia mínima permitida (ver Figuras 3.13b y 3.13c). Por último, se puede ver que los errores de seguimiento convergen a cero cuando el robot no se encuentra en peligro de colisionar.



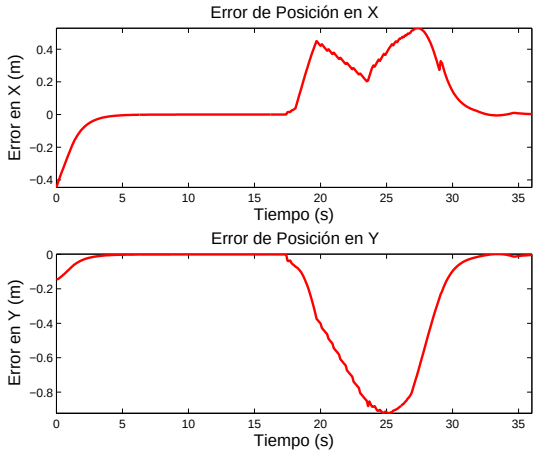
(a) Trayectoria del robot en el plano.



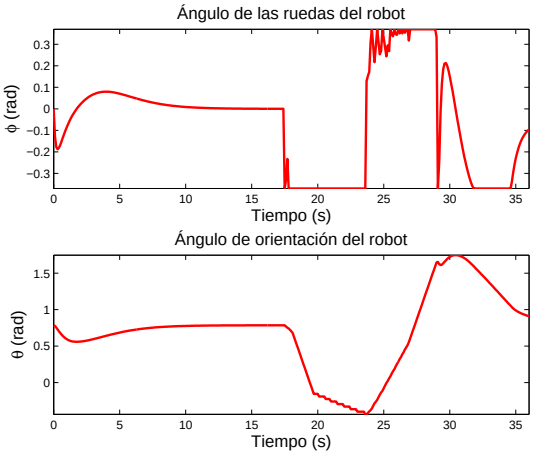
(b) Distancia entre el punto P y el obstáculo 1.



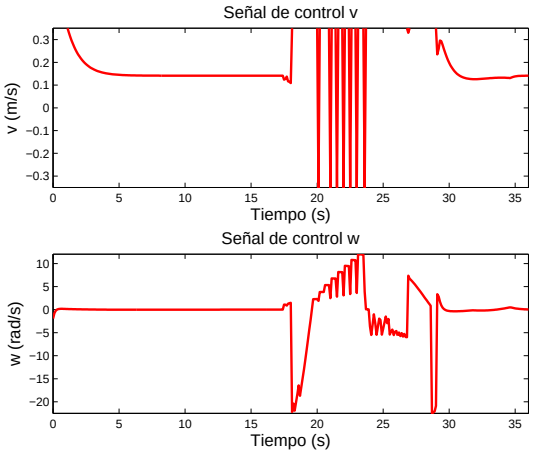
(c) Distancia entre el punto P y el obstáculo 2.



(d) Errores de posición del robot.



(e) Ángulos de orientación y dirección del robot.



(f) Señales de control.

Figura 3.11: Simulación numérica trayectoria recta, dos obs. en movimiento

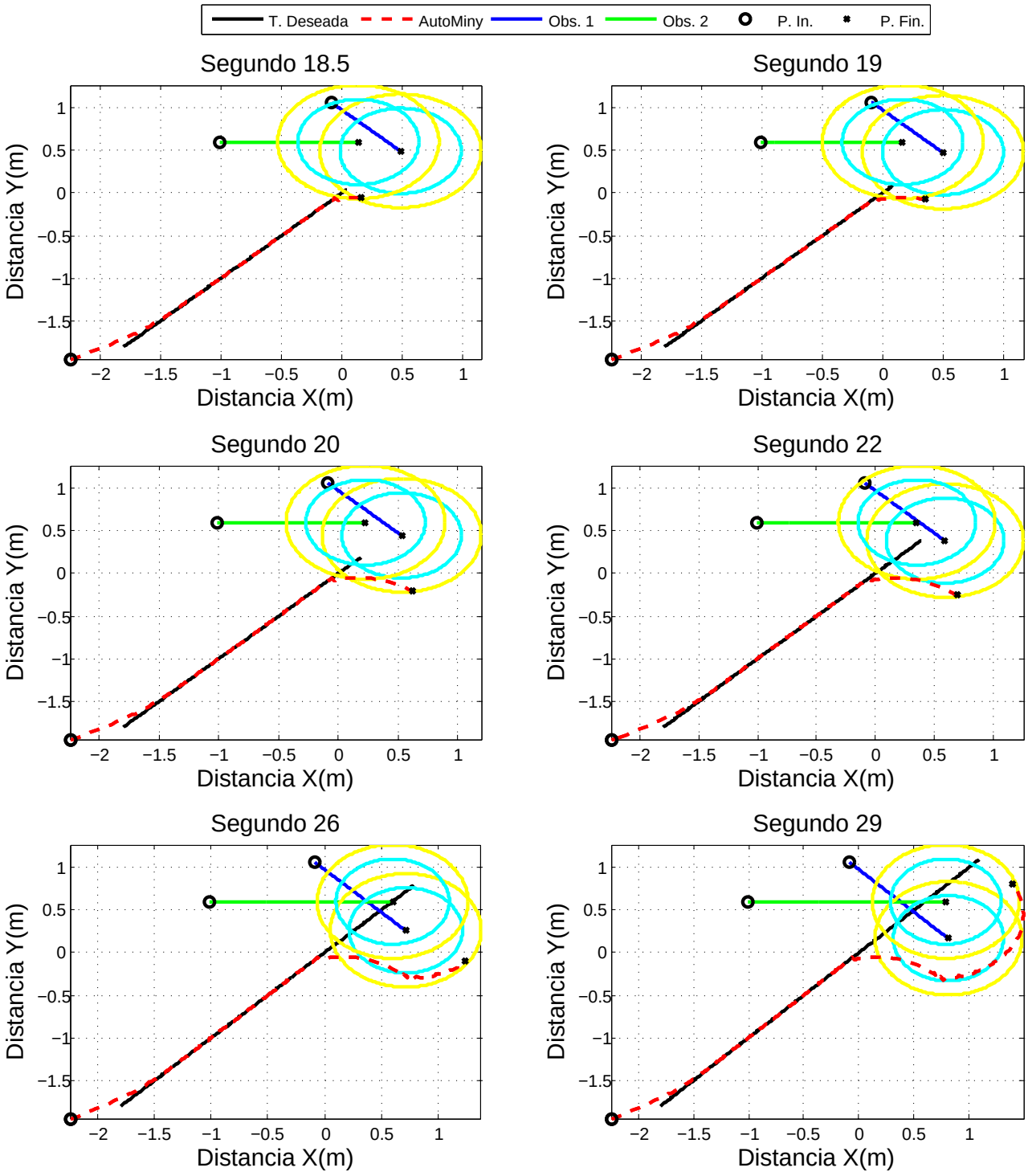
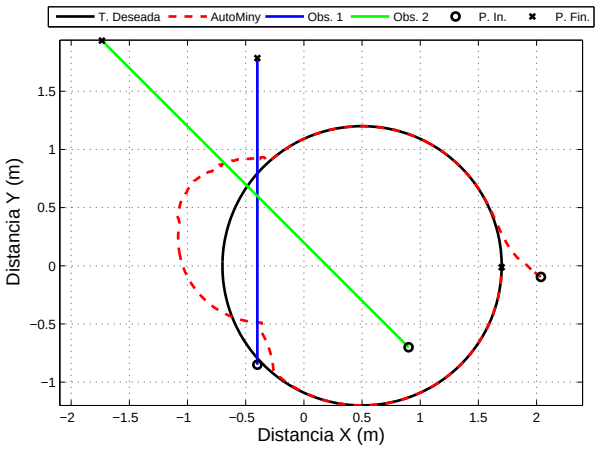
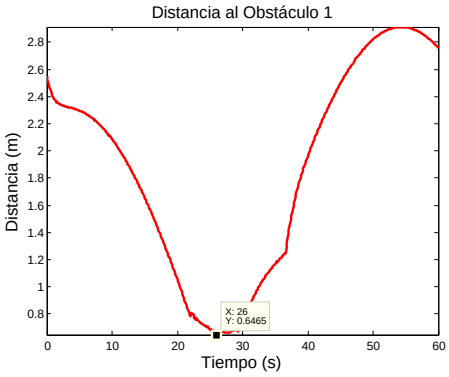


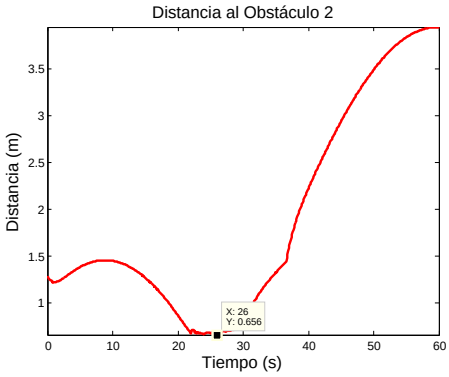
Figura 3.12: Evasión de dos obs. en movimiento, trayectoria recta



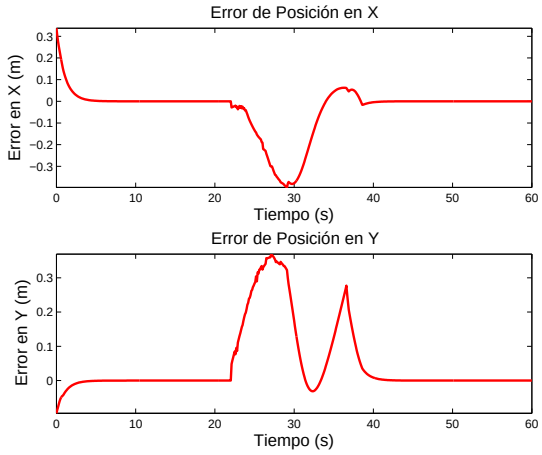
(a) Trayectoria del robot en el plano.



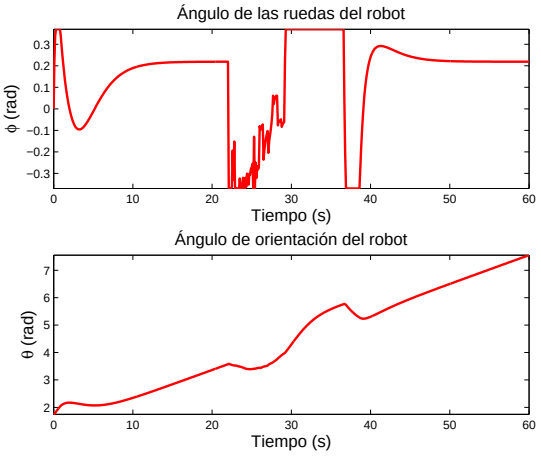
(b) Distancia entre el punto P y el obstáculo 1.



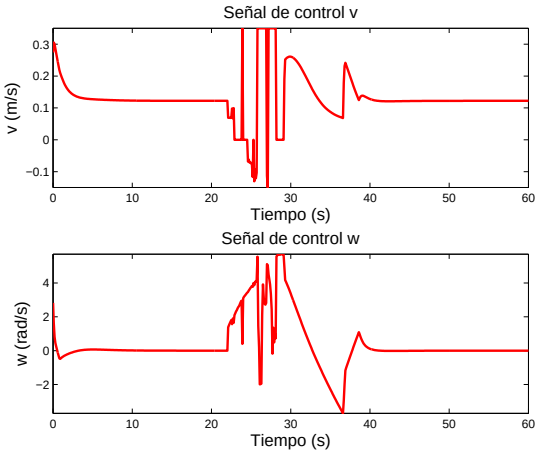
(c) Distancia entre el punto P y el obstáculo 1.



(d) Errores de posición del robot.



(e) Ángulos de orientación y dirección del robot.



(f) Señales de control.

Figura 3.13: Simulación numérica trayectoria circular, dos obs. en movimiento

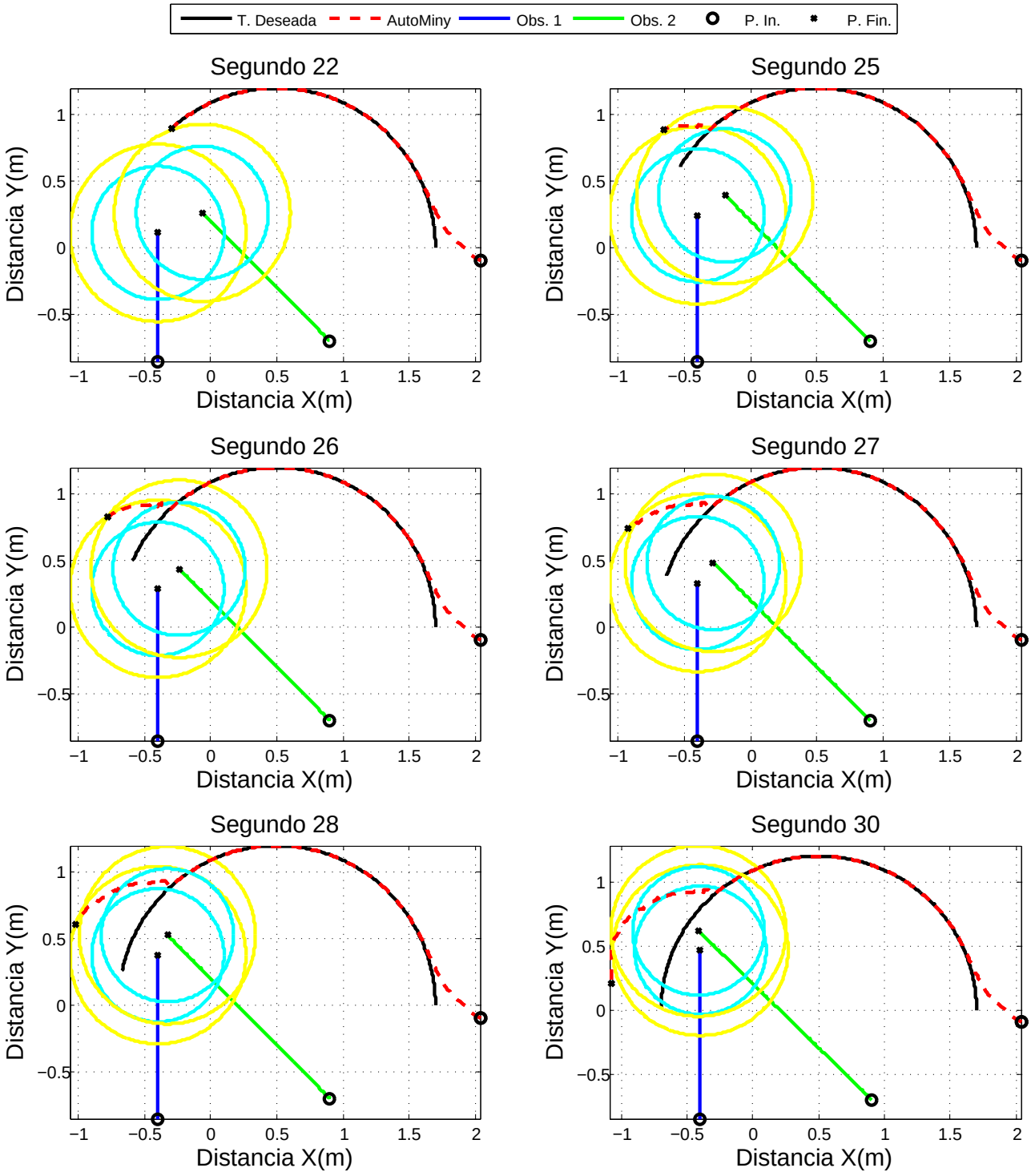


Figura 3.14: Evasión de dos obs. en movimiento, trayectoria circular

Capítulo 4

Experimentación

Para la etapa de experimentación, se utiliza la plataforma experimental descrita a continuación.

4.1. Plataforma Experimental

4.1.1. AutoMiny 4.0

El AutoMiny 4.0 (Fig. 4.1) es un modelo de vehículo autónomo (escala 1:10) desarrollado en la Freie Universität Berlin con fines educativos y de investigación. Es un robot clasificado como de tipo Ackerman el cual puede direccionar sus llantas delanteras y las traseras permanecen fijas en una misma dirección. Además se clasifica como un robot no-holonomico, es decir, que para que pueda cambiar de dirección es necesario que el robot gire. El AutoMiny es un sistema completo montado sobre un chasis, provisto de sensores de percepción (una cámara estéreo infrarroja y un LiDAR), sensores de retroalimentación (sensor de posición del ángulo de dirección y codificador del motor) con alto poder de cómputo (CPU y GPU), así como LED's

para emular las luces de los automóviles. Funciona con ROS Melodic Morenia en Ubuntu 18.04. El Autominy se basa principalmente en dos módulos de procesamiento separados: una placa controladora con un microprocesador (Arduino nano) y una computadora Intel NUC. La placa controladora es una PCB de cuatro capas donde se instala el controlador Arduino y una Unidad de Medición Inercial (IMU) adicional. El Arduino lee la información dada por ambos motores: Motor de CD (corriente directa) para la tracción y Servomotor para la dirección. El motor CD tiene un codificador incremental que nos permite conocer la velocidad del robot. Los pulsos de interrupcion son leídos por el Arduino, este los procesa y calcula la velocidad. El servomotor tiene una salida de voltaje que es leída por una entrada analógica y transformada a valores normalizados entre -1 y 1 con una calibración previamente calculada. La información se envía a través de una comunicación serial a la computadora NUC para publicar topics en el entorno ROS [1]. Realizando pruebas de funcionamiento de velocidad de tracción y dirección previas a los experimentos, se obtuvo que el robot tiene una velocidad máxima/minima de ± 2.5m/s. El ángulo máximo de giro de las ruedas 0.37 radianes y el mínimo -0.37 radianes. Además, se tiene que a velocidades entre -0.06m/s y 0.06m/s se entra en la zona muerta del motor de tracción, por lo que en este rango de velocidades el robot no se desplaza.

4.1.2. Robot Operating System (ROS)

ROS (Robot Operating System) es un paquete de desarrollo de software de código abierto para aplicaciones de robótica. ROS ofrece una plataforma de

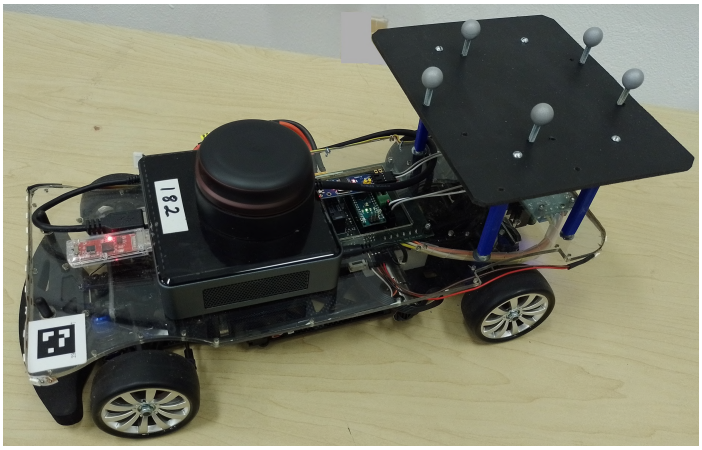


Figura 4.1: AutoMiny 4.0.

software estándar para desarrolladores, que les provee de herramientas para la investigación y creación de prototipos para su posterior implementación y producción. [2] Fue desarrollado en Willow Garage, Laboratorio de Inteligencia Artificial de Stanford. ROS provee abstracción de hardware, controladores de dispositivos, librerías, herramientas de visualización, comunicación por mensajes, administración de paquetes y más.

ROS no es un sistema operativo hablando en el sentido tradicional, en el cual se puedan de administrar y planificar procesos, sino que provee una capa de comunicaciones estructuradas por encima del sistema operativo anfitrión. Provee la funcionalidad básica de cualquier sistema operativo tradicional como: abstracción del hardware, acceso a dispositivos de bajo nivel, implementación de funcionalidad de uso común, comunicación entre procesos, mantenimiento de paquetes, entre otros, además proporciona librerías, y todo tipo de herramientas (visualizadores, etc.) para el desarrollo de robots y sus aplicaciones. ROS organiza el software según una arquitectura de grafos en la que en el más bajo nivel se tienen los llamados nodos (porción de código ejecutable) y

éstos se comunican entre sí a través de tópicos o topics (interfaz de ROS para el paso de mensajes entre nodos). A su vez los nodos se agrupan en paquetes (colecciones mínimas de código reusable), y éstos en stacks que permiten compartir el código fácilmente. Una de las ventajas de ROS es que es de código abierto y tiene una comunidad creciente que lo apoya y da mantenimiento, así como gran actividad respecto de módulos disponibles para controladores de dispositivos y algoritmos de robótica móvil.
A nivel de software los paquetes se organizan con los elementos listados a continuación:

- CMakeFile: Es el archivo que se encarga de compilar el paquete. En él están los códigos que el paquete utiliza, librerías y las dependencias del mismo.
- Package.xml: Contiene las dependencias del paquete, y se define como serán construidos y ejecutados.
- Src: es una carpeta que contiene el código que se convierte en ejecutables o nodos.
- Launch: contiene los lanzadores .launch que permiten iniciar y configurar varios nodos a la vez.

ROS provee herramientas para visualización de información en tiempo real como es el caso de Rviz. Por medio de ella pueden visualizarse imágenes, mediciones de odometría, nubes de puntos, entre otras salidas de sensores [11].

4.1.3. OptiTrack

El sistema utilizado para la retroalimentación de la posición y orientación del robot es el sistema OptiTrack [3]. Este es un sistema de captura de movimiento desarrollado por la compañia Natural Point [4]. Consiste en un conjunto de ocho cámaras infrarrojas modelo "*Flex 13*" (Fig. 4.2). Cuya resolución es de 1280 x 1024 [px] y una velocidad de captura de 120 cuadros por segundo (FPS).

Figura 4.2: Cámara modelo "*Flex 13*".

Entre otras formas de trabajo, estas cámaras pueden ser configuradas para la detección de marcadores reflejantes (Rigid Bodies, marcadores individuales o esqueletos de varios marcadores) que se colocan en forma de un patrón geométrico irregular sobre el robot y cuyo propósito es trasladar el movimiento real a un modelo virtual con el cual se pueda trabajar. La información generada por las cámaras es procesada en el software Motive. Este transforma dicha información a valores de posición y orientación en el espacio de trabajo. Por medio de este sistema y colocando marcadores sobre el AutoMiny (ver

Fig. 4.1) es posible obtener la posición y orientación del eje trasero del robot y así determinar la posición el punto frontal P.

4.1.4. Integración de la plataforma experimental

La plataforma experimental utilizada para validar los resultados teóricos se presenta en la Fig. 4.3. En esta plataforma ocho cámaras son interconectadas por medio de dos OptiHubs (ver Fig. 4.4). Cada Optihub conecta cuatro cámaras. Los Optihubs se conectan a una computadora con sistema operativo Windows 11 y el software Motive. Este software procesa la información recibida de las cámaras y obtiene la posición del robot. Dicho equipo envía la información a travéz de una red wifi a otra computadora en la cual se tiene instalado un sistema operativo Ubuntu 20.04 y ROS Noetic Ninjemys. En la computadora con Ubuntu se desarrolla el programa de control, escrito en C/C++, en dicho programa se solicitan información de la posición y orientación del robot, así como la posición del obstáculo. La posición del obstáculo es determinada por medio de un sensor LiDAR ubicado sobre el robot. Dicho sensor entrega un vector de tamaño 360, es decir una componente del vector por cada grado alrededor del robot. En cada componente se almacena la distancia de un objeto en esa posición angular respecto al robot, como se muestra en la Fig.4.5 . Con la información recibida del LiDAR y las cámaras se implementan las distintas estrategias de control que, en este trabajo, se diseñan para la evasión de colisiones. La estrategia de control calcula las señales de control (velocidad de las ruedas traseras y ángulo de dirección de las delanteras) que se envían al robot a travéz de una red wifi. En la Fig.4.6

se observa un diagrama de conección de la plataforma experimental.

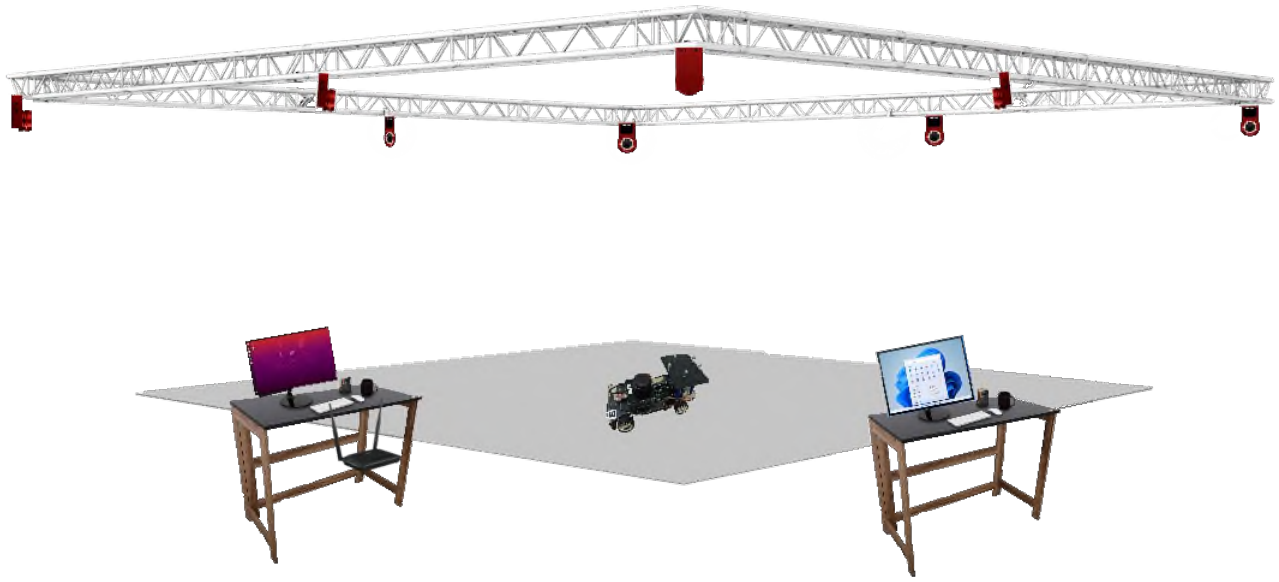


Figura 4.3: Plataforma Experimental.

Figura 4.4: OptiHub.

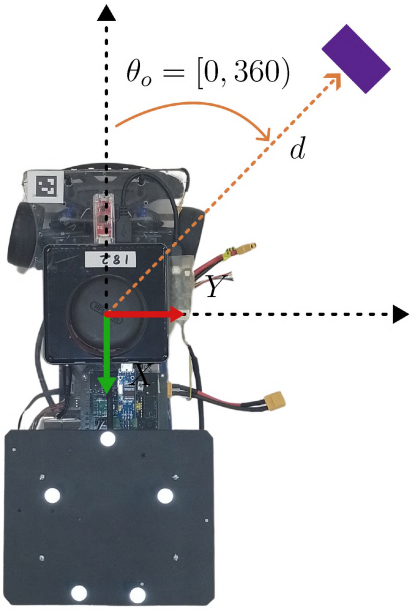


Figura 4.5: Marco de referencia del LIDAR.

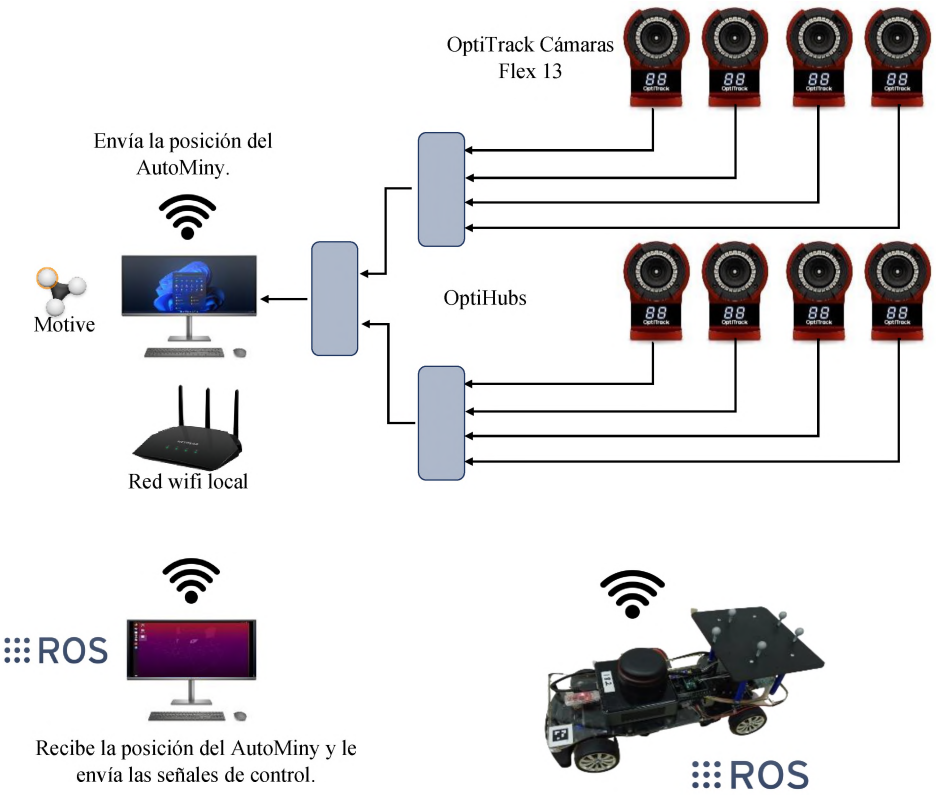


Figura 4.6: Diagrama de conexión de la plataforma experimental.

4.2. Resultados de experimentación

A continuación se presentan los resultados experimentales para cada simulación presentada en el capítulo anterior, haciendo uso de la estrategia de control (3.12). El punto P que se desea controlar se encuentra a una distancia $\Delta = 0.1m$ al frente del punto medio del eje de las ruedas delanteras y la distancia entre los ejes trasero y delantero es $\ell = 0.26m$, la distancia mínima permitida entre el punto frontal y el obstáculo es $d = 0.5m$.

4.2.1. Evasión de colisiones entre el robot y un obstáculo fijo

Primero se muestra el caso cuando no se considera la restricción en el ángulo de dirección de las ruedas delanteras.

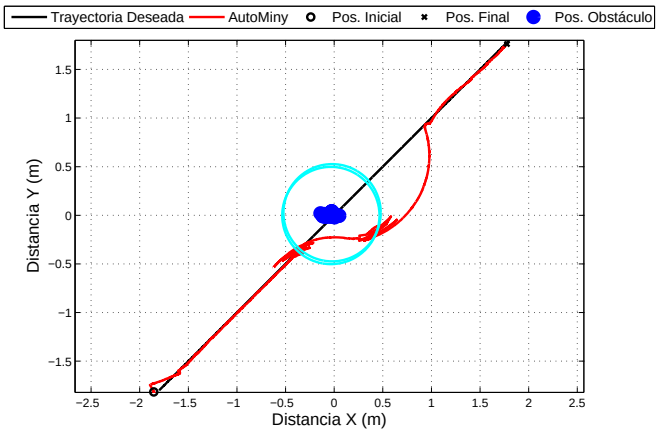
Resultados experimentales sin considerar el ángulo de restricción

La trayectoria deseada es la recta definida por

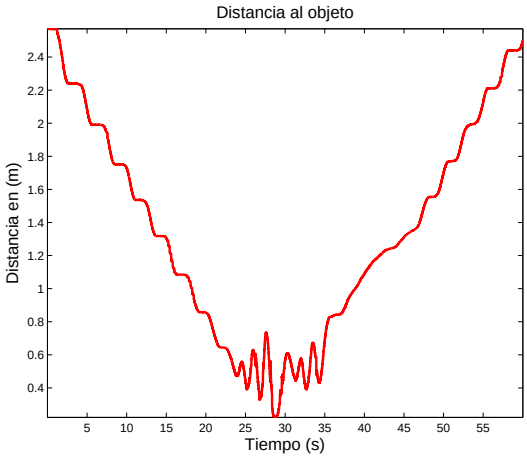
$$m(t) = \left[\; 0.06t - 1.8, \;\; 0.06t - 1.8 \; \right].^{T} \tag{4.1}$$

Aquí la velocidad máxima de la trayectoria es $\eta = 0.084852813$. Las ganancias de control y repulsión son $k_x, k_y = 0.8$ y $\epsilon = 1.2\frac{(k\sqrt{2}+\eta)}{d} = 2.918936$. La Fig. 4.7a muestra la trayectoria descrita por el robot en el plano. De forma similar a la simulación numérica, el AutoMiny entra en la zona de colisión debido a sus restricciones de giro. También se puede observar como el AutoMiny sigue la trayectoria deseada una vez que sale de la zona de repulsión. En la Fig. 4.7b se muestra la distancia del punto frontal con respecto al obstáculo, la cual llega a ser menor que $0.5m$. En la Fig. 4.7c se presentan los

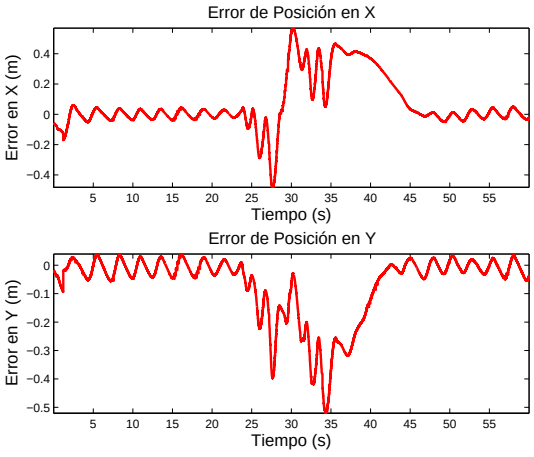
errores de posición del AutoMiny. Estos oscilan alrededor a cero, mientras no encuentre un obstáculo en la trayectoria. En la Fig. 4.7d podemos ver el ángulo de orientación θ del robot y el ángulo ϕ de dirección de las ruedas delanteras durante el seguimiento de la trayectoria, tanto el medido en los sensores, como el utilizado en la entrada de control (integral). Por último, en la Fig. 4.7e, se muestran las señales de control que se envían al robot.



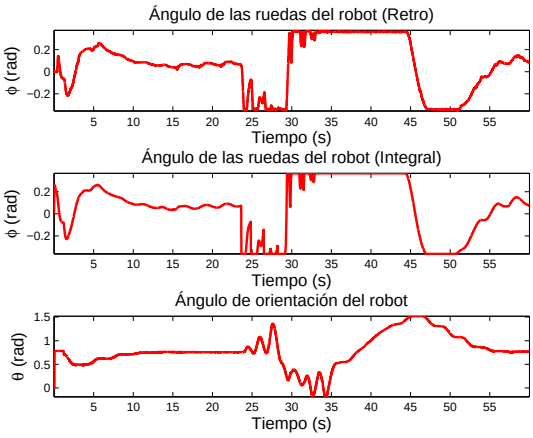
(a) Trayectoria del AutoMiny en el plano.



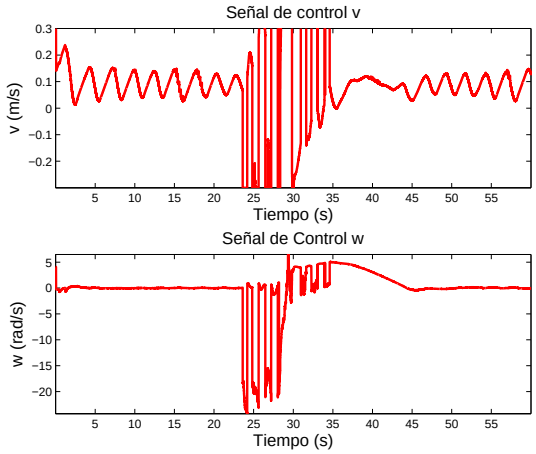
(b) Distancia entre el punto frontal y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del AutoMiny.



(e) Señales de control.

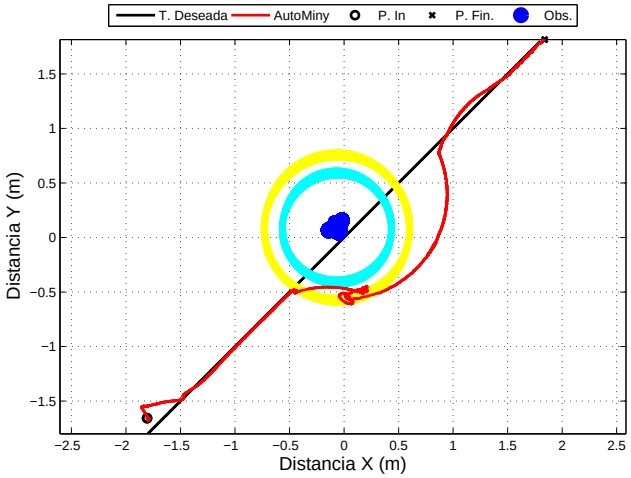
Figura 4.7: Resultados experimentales sin considerar restricción en ángulo de giro

Resultados experimentales considerando el ángulo de restricción

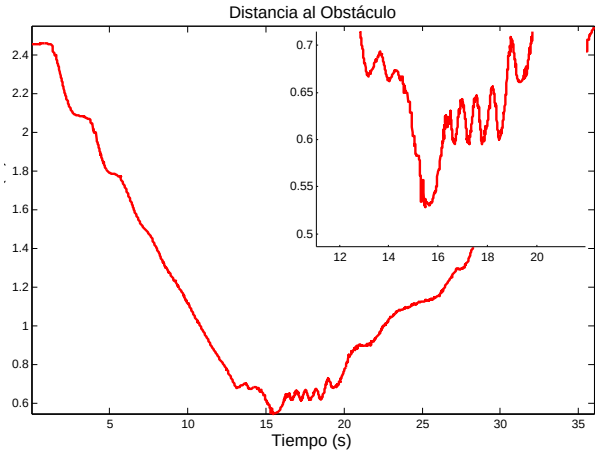
Para evitar entrar en la zona de colisión se necesita considerar el ángulo de restricción, de manera que el robot se mantenga siempre a una distancia mayor a la mínima permitida al obstáculo. Para la experimentación se usaron las trayectorias descritas en la Tabla 3.1.

1. Trayectoria recta: La ganancia de repulsión para la trayectoria es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta)}{d_m} = 2.8029$. Como se observa en las Figuras(4.8a, 4.8b), el AutoMiny entra en el círculo exterior del campo repulsivo (círculo amarillo), pero logra mantenerse fuera de la zona de colisión del obstáculo.

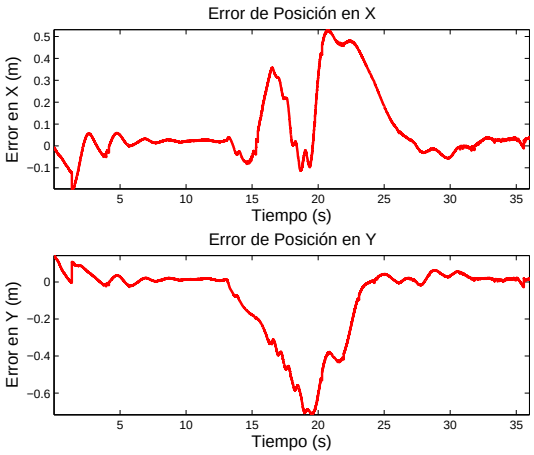
2. Trayectoria circular: La ganancia de repulsión para la trayectoria circular es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta)}{d_m} = 2.774619$. Los resultados se presentan en la Fig.(4.9). Al igual que en el caso anterior, el AutoMiny se mantiene siempre a una distancia mayor a la mínima permitida, además que retoma la trayectoria deseada al salir de la zona de repulsión.



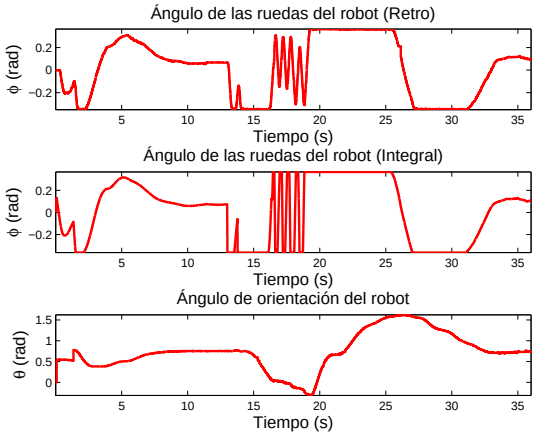
(a) Trayectoria del AutoMiny en el plano.



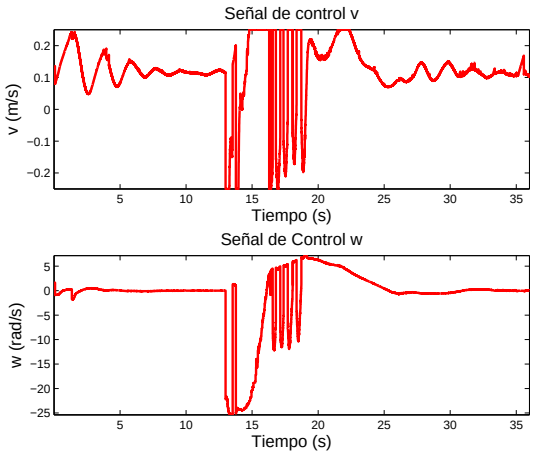
(b) Distancia entre el punto frontal y el obstáculo.



(c) Errores de posición del robot.



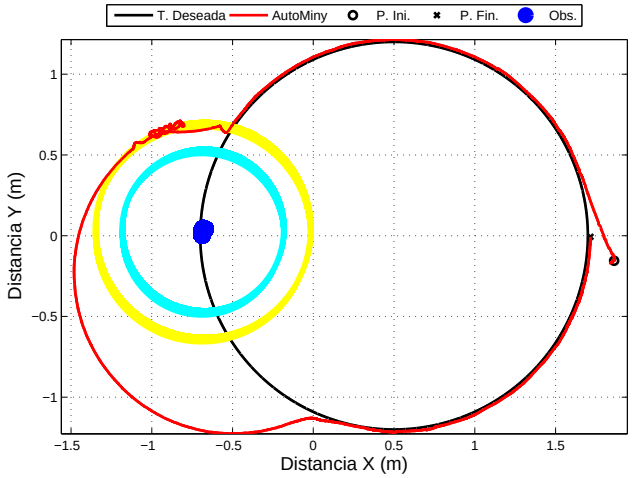
(d) Ángulos de orientación y dirección del AutoMiny.



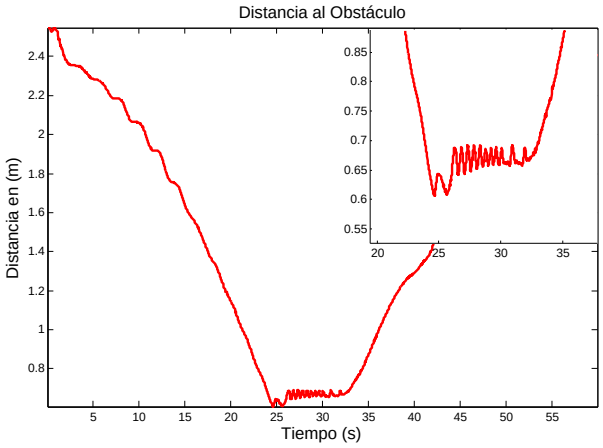
(e) Señales de control.

Figura 4.8: Resultados experimentales trayectoria recta, obs. fijo

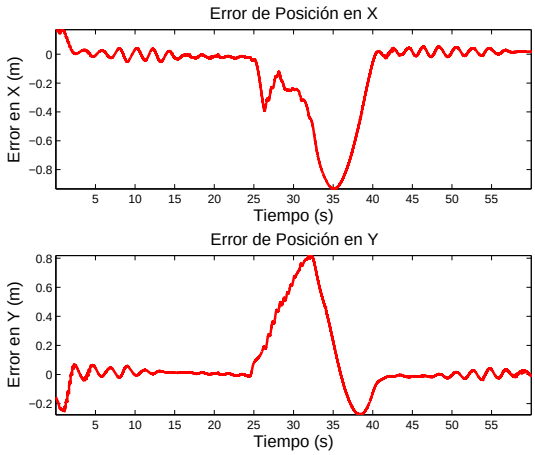
4.2 Resultados de experimentación



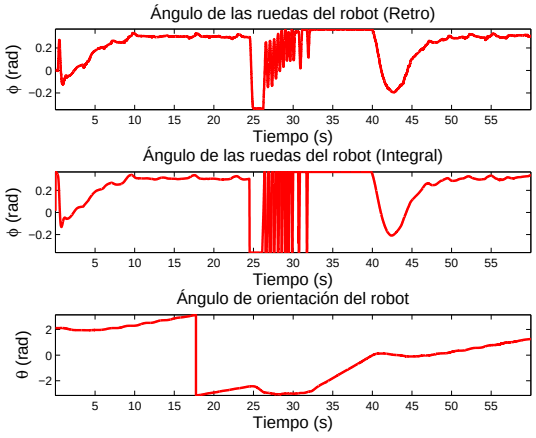
(a) Trayectoria del AutoMiny en el plano.



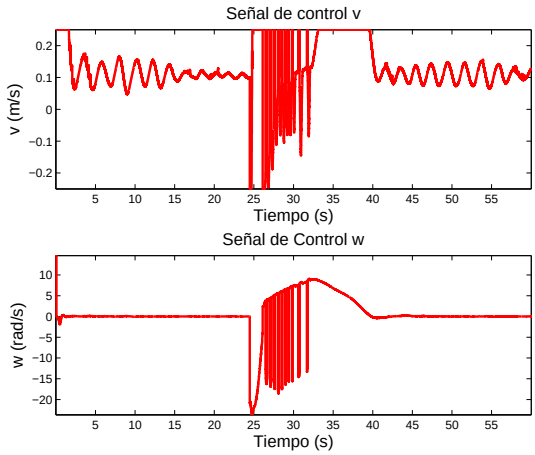
(b) Distancia entre el punto frontal y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del AutoMiny.



(e) Señales de control.

Figura 4.9: Resultados experimentales trayectoria circular, obs. fijo

4.2.2. Evasión de colisiones entre el robot y un obstáculo en movimiento

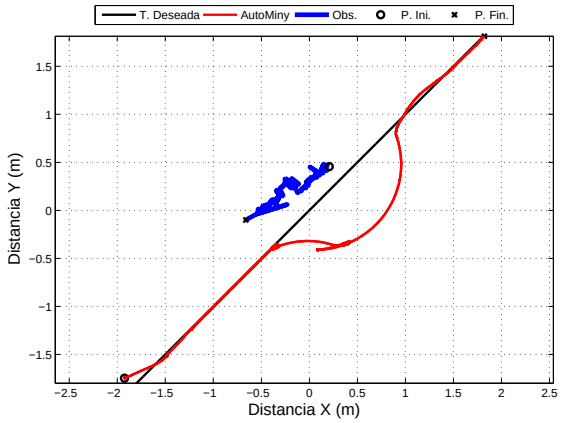
Los experimentos se realizaron con las mismas trayectorias y parámetros utilizados para la simulación numérica (Tabla 3.1). El obstáculo en movimiento es un robot diferencial Lego Mindstorms EV3 (Fig.4.10). Este robot fue configurado para tener una velocidad máxima $\eta_o = 0.0622\ m/s$.

Figura 4.10: Lego Mindstorms EV3.

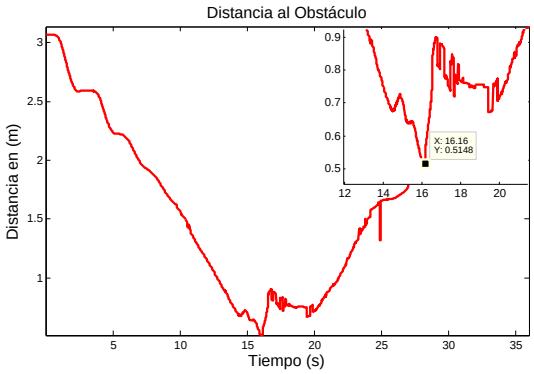
1. Trayectoria recta: Los resultados del seguimiento para la trayectoria recta se muestran en las Figuras 4.11 y 4.12. En la Fig. 4.11b se observa que la distancia entre el AutoMiny y el obstáculo es en todo tiempo mayor a $d = 0.5$m. En la Fig. 4.11c se muestran las señales del error, en las cuales se observa que el error se encuentra en una vecindad a cero, cuando el AutoMiny está fuera de peligro de colisionar. Por último, se muestran el ángulo de dirección del robot, el ángulo de orientación (ver Fig.4.11d) y

las señales de control en la Fig. 4.11e.

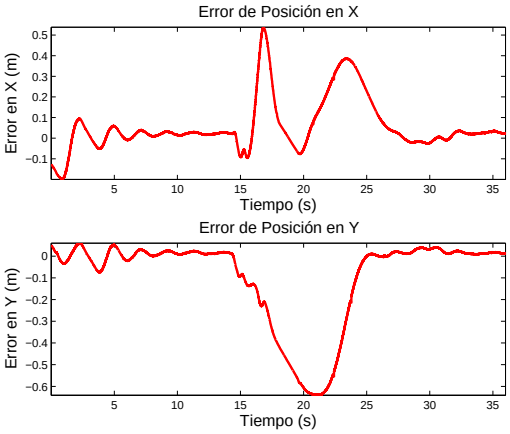
2. Trayectoria circular: Los resultados experimentales para la trayectoria circular se muestran en las Figuras 4.13 y 4.14. Al igual que la simulación numérica el AutoMiny logra mantener la distancia mínima d, además de seguir la trayectoria deseada (Fig. 4.13a). En la Fig. 4.14 se muestra detalladamente la posición del robot y del obstáculo en diferentes instante de tiempo. Como se puede observar, es en el segundo 30 cuando detecta la presencia del obstáculo y entra en acción el campo vectorial repulsivo.



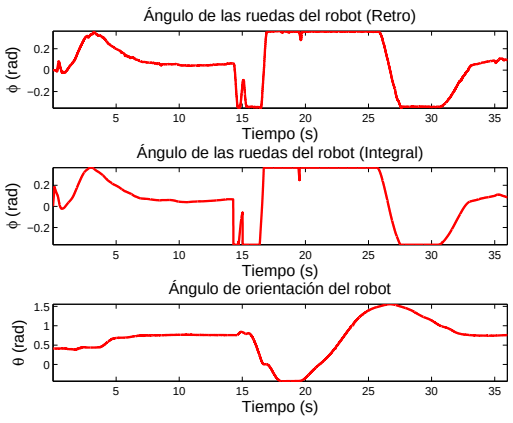
(a) Trayectoria del AutoMiny en el plano.



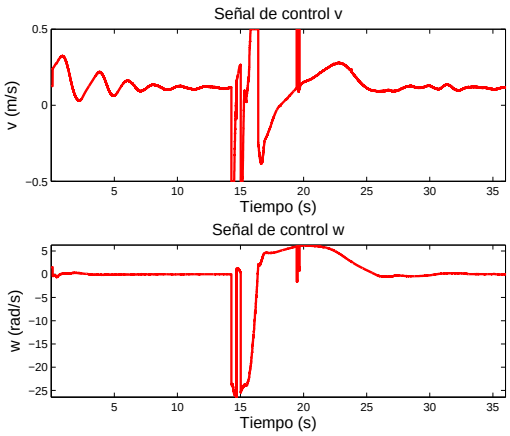
(b) Distancia entre el punto frontal y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del AutoMiny.



(e) Señales de control.

Figura 4.11: Resultados experimentales trayectoria recta, obs. en movimiento

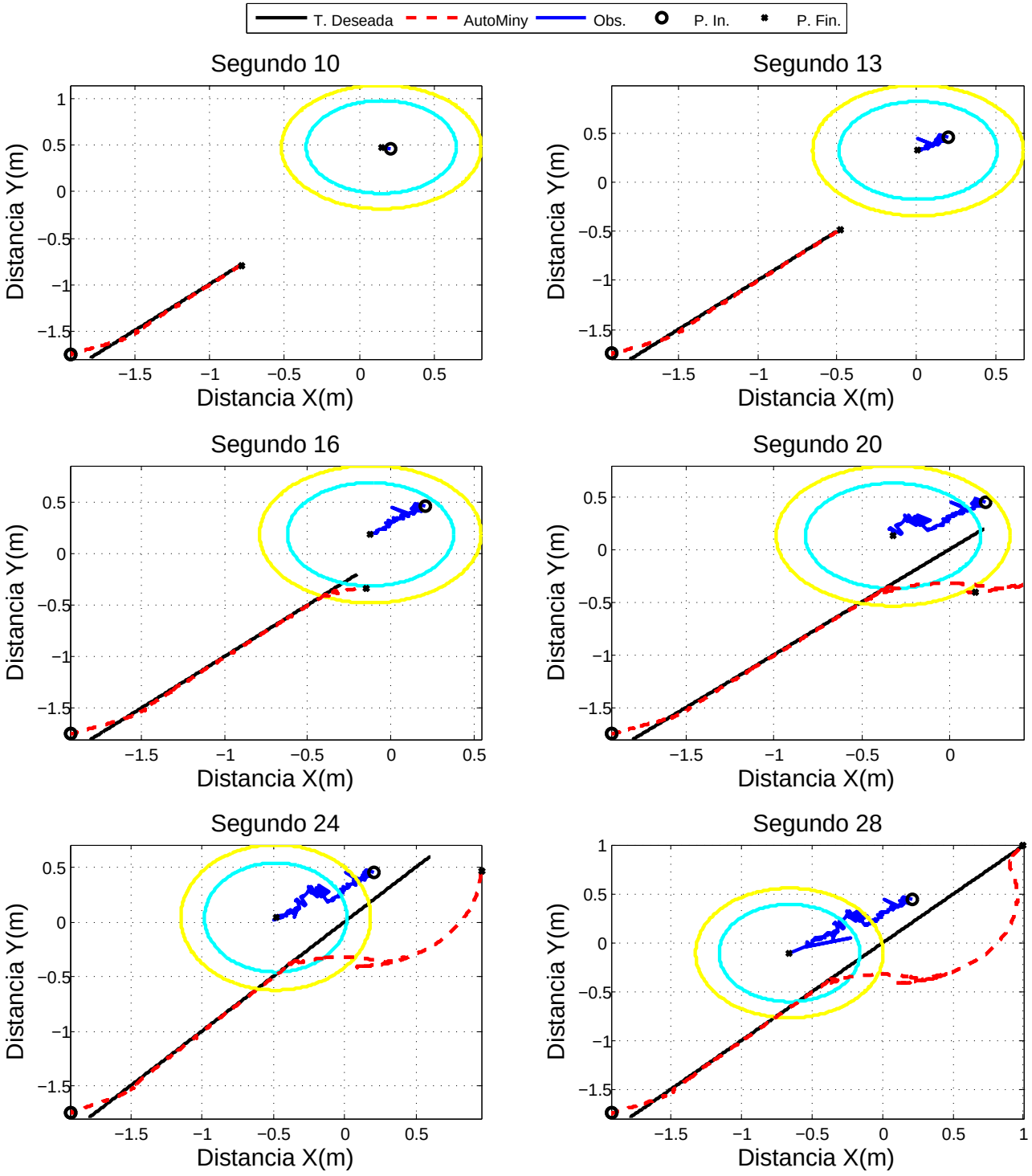
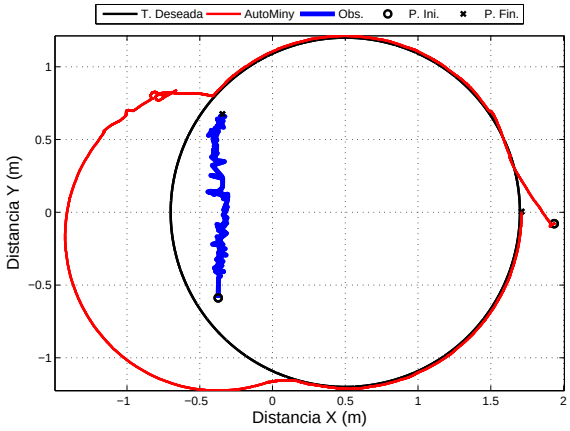
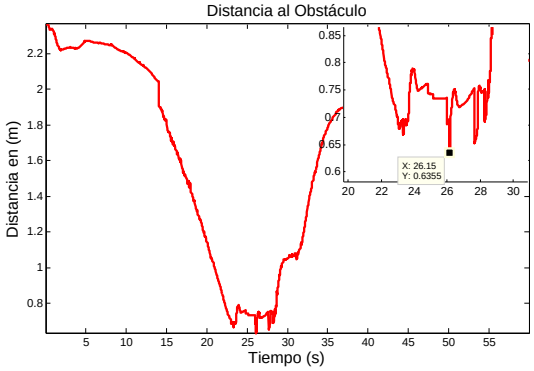


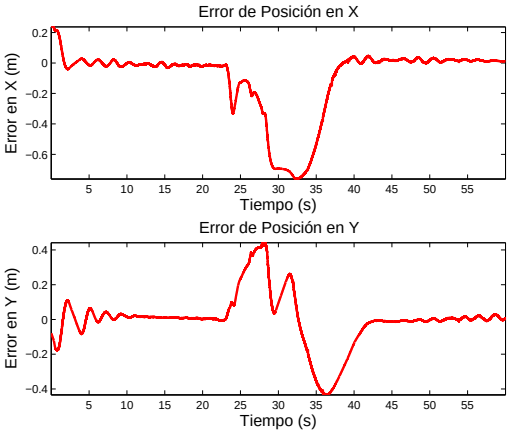
Figura 4.12: Evasión del obstáculo en diferente instante, trayectoria recta



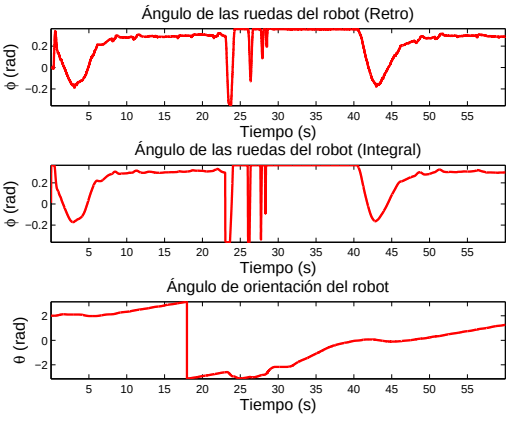
(a) Trayectoria del AutoMiny en el plano.



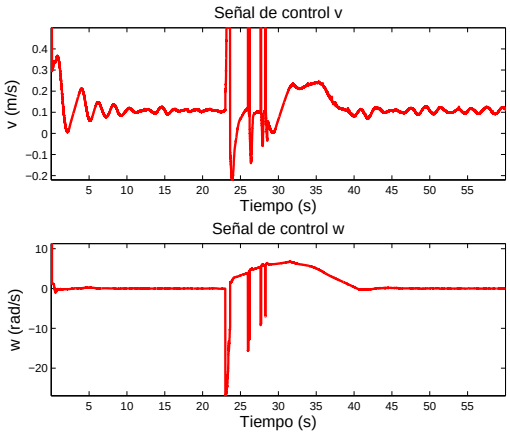
(b) Distancia entre el punto frontal y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del AutoMiny.



(e) Señales de control.

Figura 4.13: Resultados experimentales trayectoria circular, obs. en movimiento

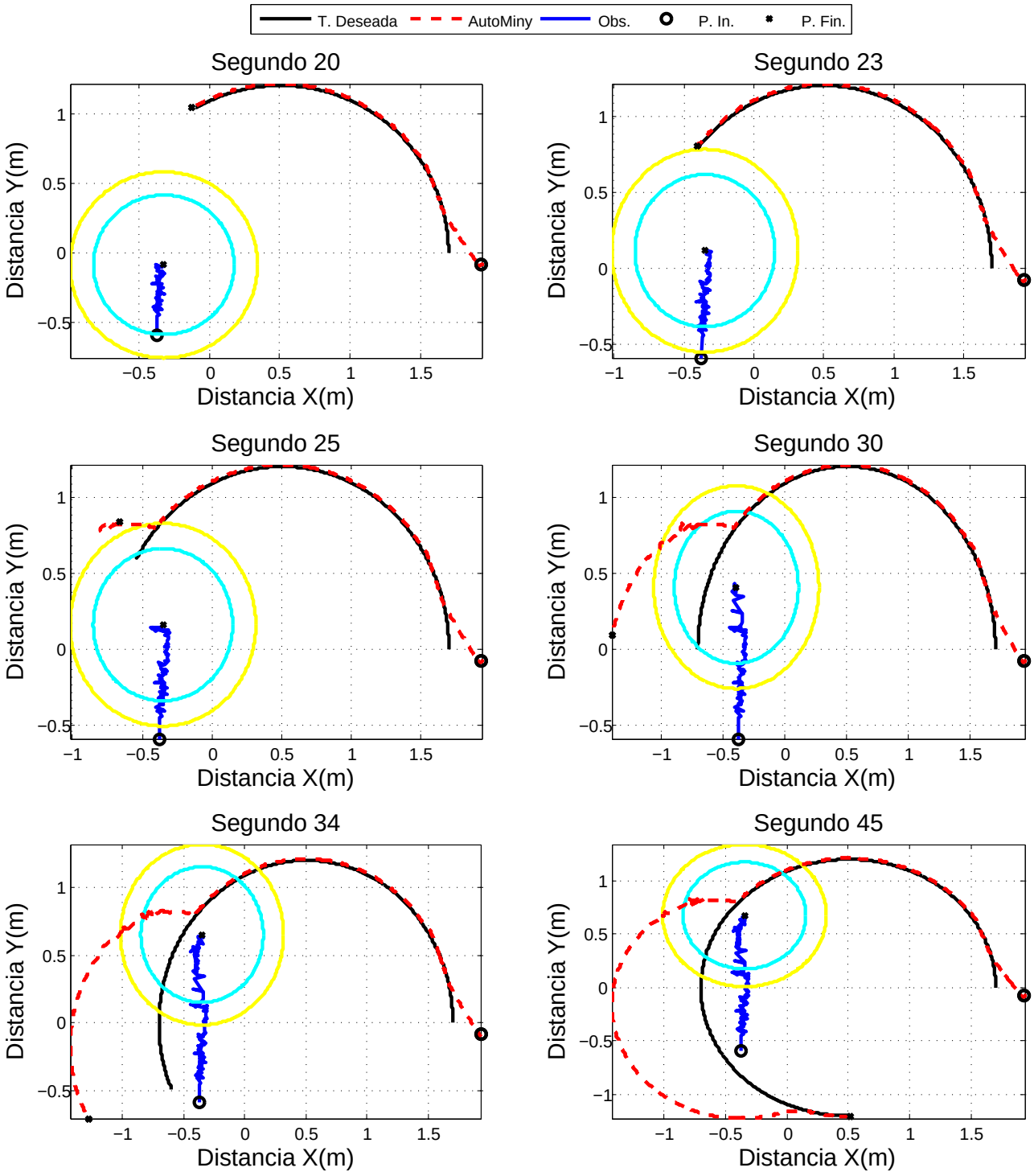


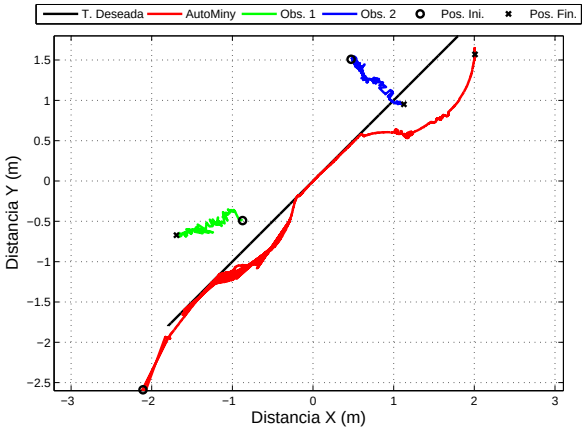
Figura 4.14: Evasión del obstáculo en diferente instante, trayectoria circular

4.2.3. Evasión de colisiones contra más de un obstáculo en movimiento en diferente tiempo.

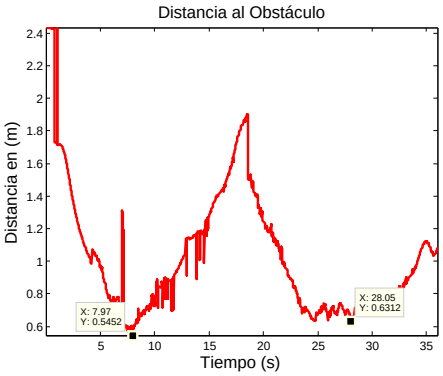
Complementando los resultados de la subsección anterior, se procede a mostrar los resultados experimentales cuando el AutoMiny encuentra en su camino a más de un obstáculo en diferente instante de tiempo. Las trayectorias y parámetros a utilizar se muestran en la Tabla 3.1.

1. Trayectoria recta: Los resultados se muestran en las Figrs. 4.15 y 4.16. Como se puede observar en la Fig. 4.15b, el AutoMiny siempre se mantiene a una distancia mayor a la mínima permitida. Además que cuando no se encuentra en peligro de colision sigue la trayectoria deseada como se ve en la Fig. 4.15a.

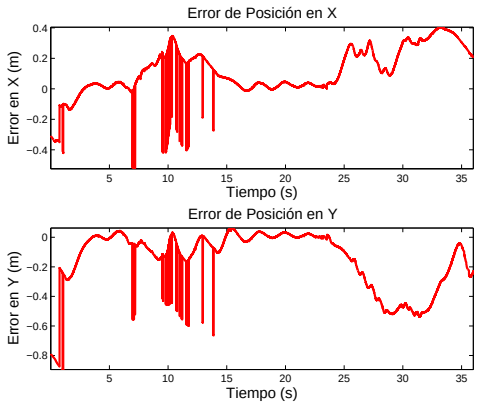
2. Trayectoria circular: Los resultados experimentales cuando el AutoMiny sigue la trayectoria circular se muestran en las Figrs. 4.17 y 4.18. Al igual que en el caso anterior el AutoMiny se mantiene siempre a una distancia mayor a la mínima perimitida, Figrs. 4.17b y 4.17c. Además como se puede ver en la Fig. 4.18, una vez que el AutoMiny sale de la zona de colisión, retoma la trayectoria deseada.



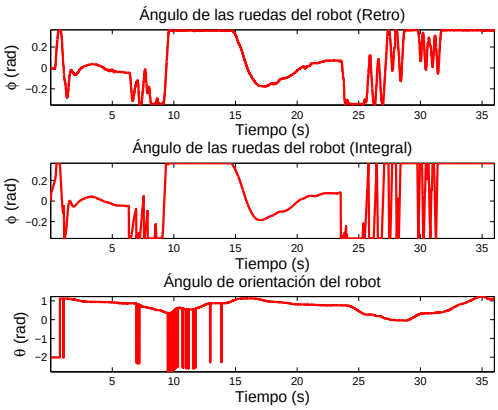
(a) Trayectoria del AutoMiny en el plano.



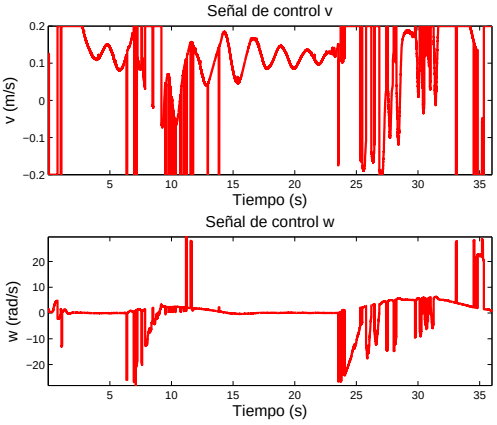
(b) Distancia entre el punto frontal y el obstáculo.



(c) Errores de posición del robot.



(d) Ángulos de orientación y dirección del AutoMiny.



(e) Señales de control.

Figura 4.15: Resultados experimentales trayectoria recta, dos obs. en diferente tiempo

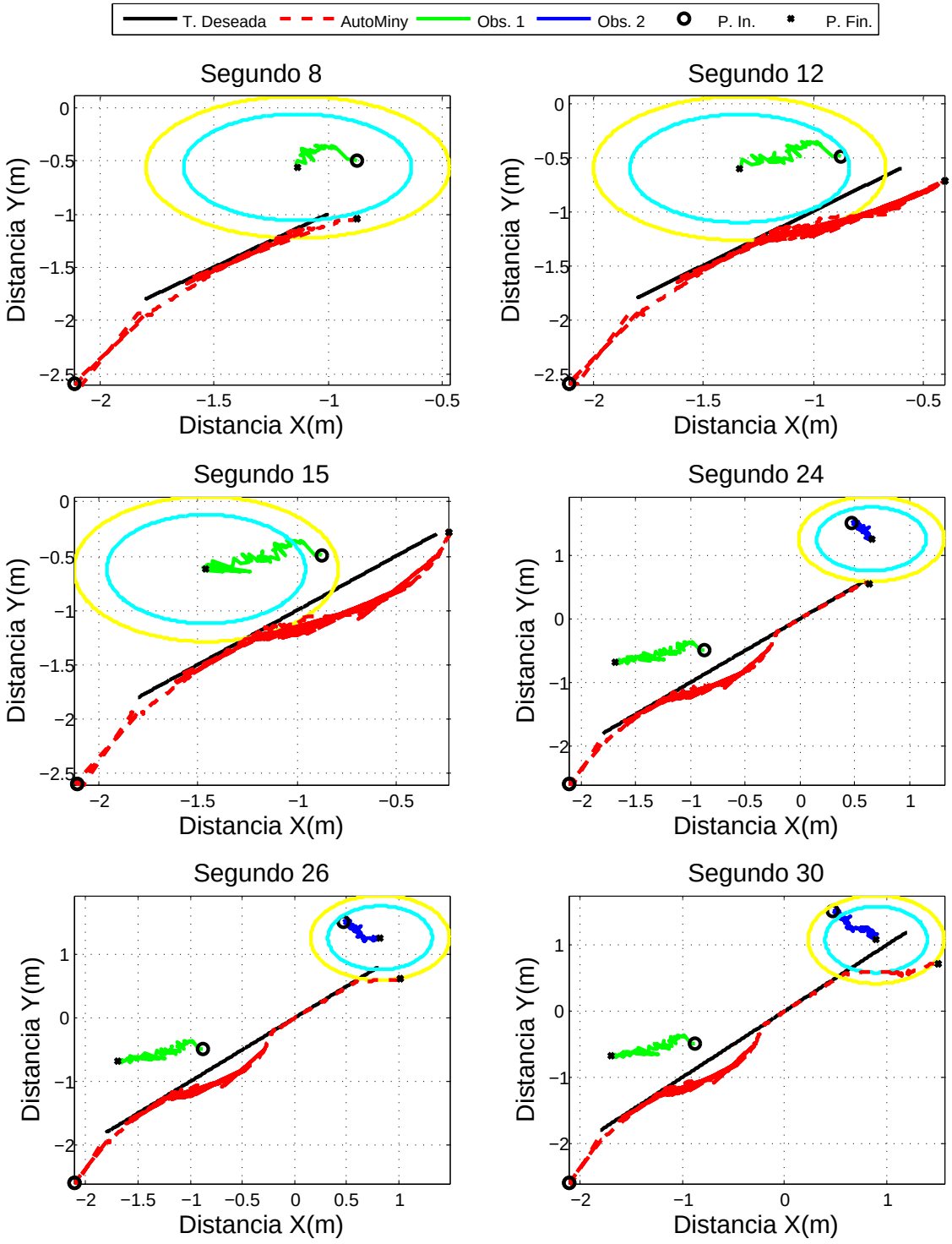
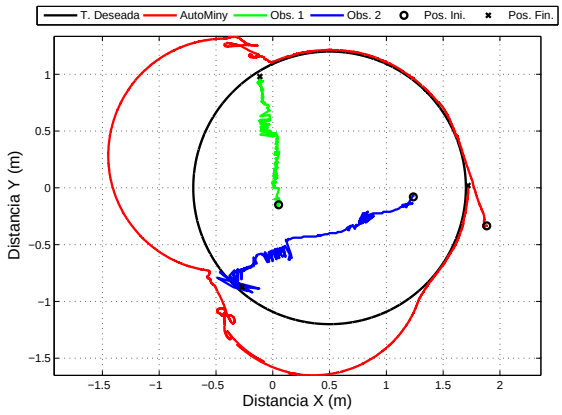
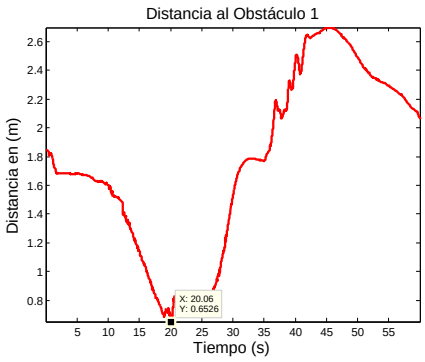


Figura 4.16: Evasión de los obs. en diferente tiempo, trayectoria recta

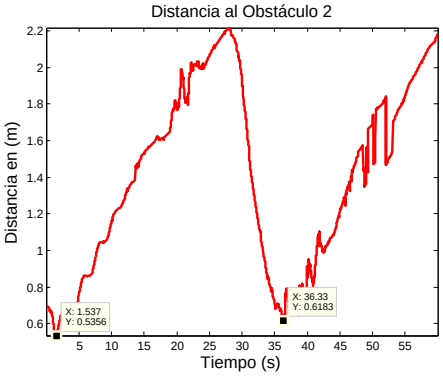
4.2 Resultados de experimentación



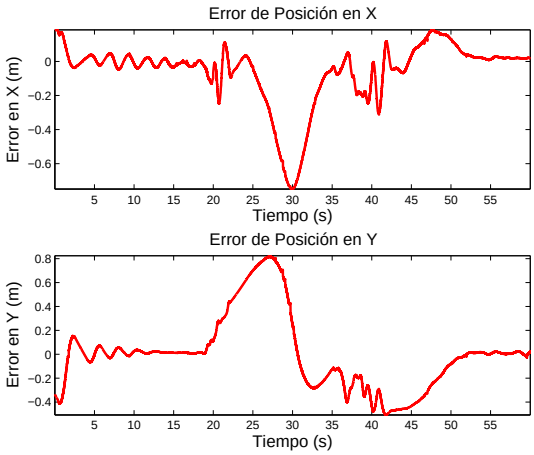
(a) Trayectoria del AutoMiny en el plano.



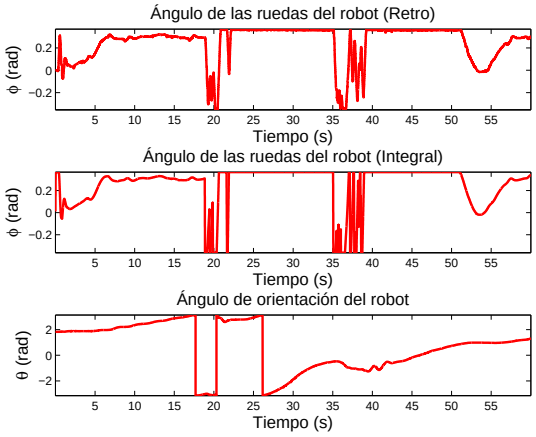
(b) Distancia entre el punto frontal y el obstáculo 1.



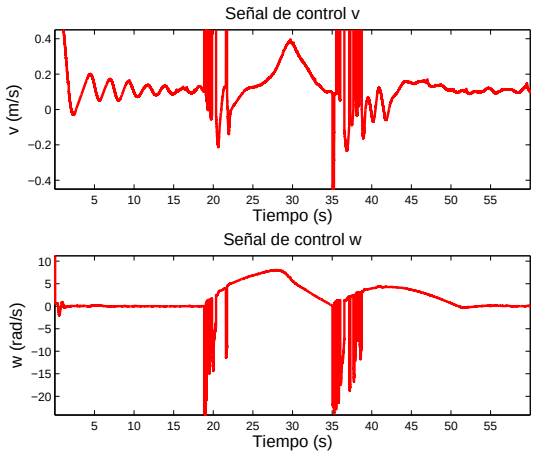
(c) Distancia entre el punto frontal y el obstáculo 2.



(d) Errores de posición del robot.



(e) Ángulos de orientación y dirección del AutoMiny.



(f) Señales de control.

Figura 4.17: Resultados experimentales trayectoria circular, dos obs. en diferente tiempo

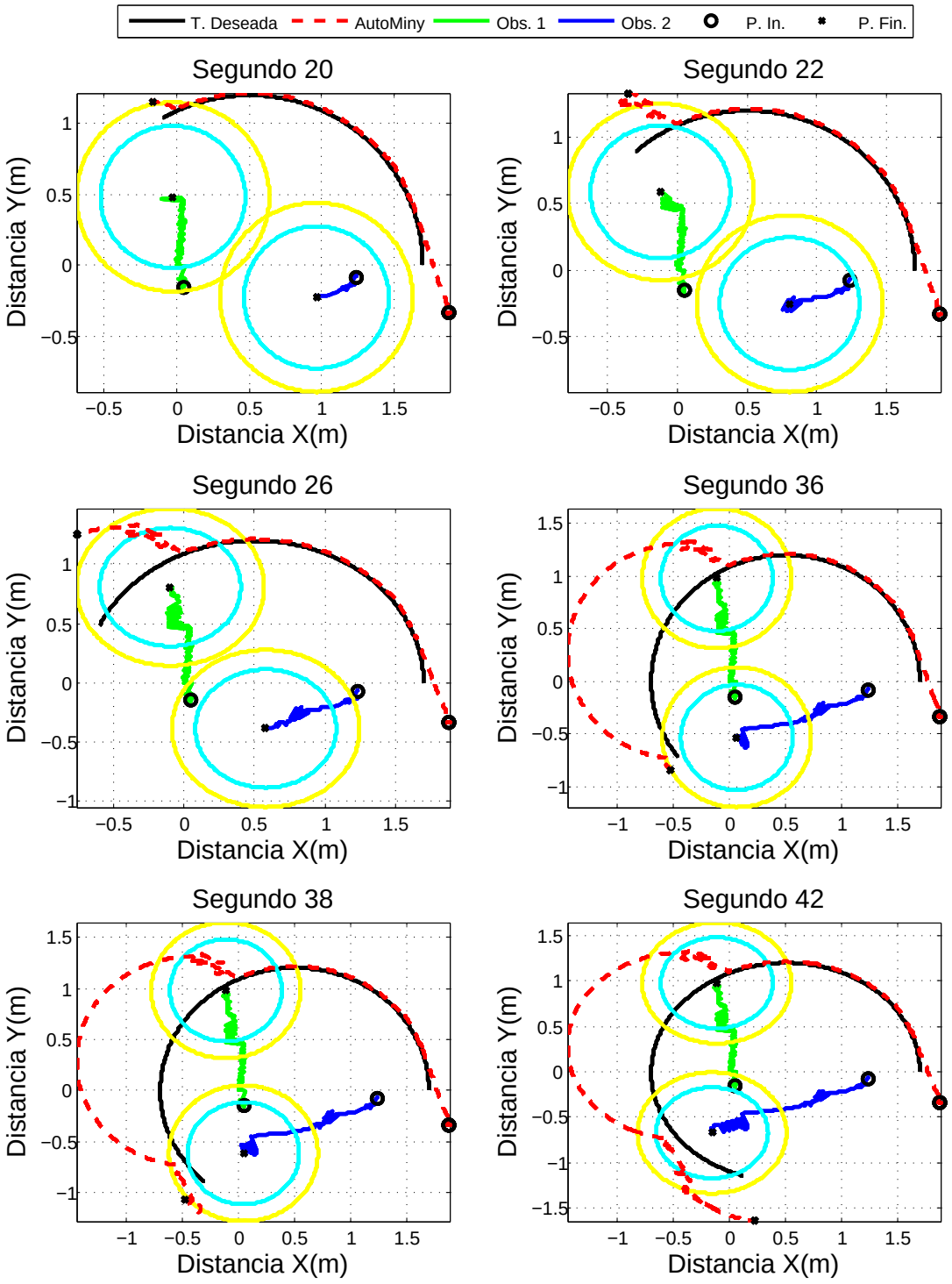


Figura 4.18: Evasión de los obs. en diferente tiempo, trayectoria circular

4.2.4. Evasión de colisiones contra más de un obstáculo en movimiento al mismo tiempo.

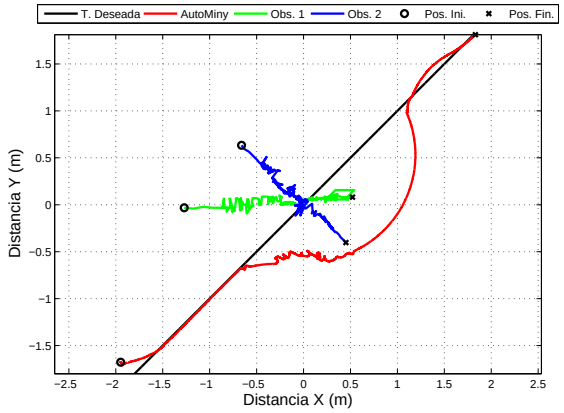
Para validar los resultados teóricos se realiza la siguiente experimentación. Los obstáculos en movimiento, al igual que en los casos anteriores, son dos robots Lego Mindstorms EV3 (ver Fig. 4.10) que se mueven a velocidades distintas. Las trayectorias y parámetros se describen en la tabla 3.1.

1. Trayectoria recta: La velocidad máxima de los obstáculos 1 y 2 son $\eta_{o1} = 0.044m/s$ y $\eta_{o2} = 0.0622m/s$ respectivamente. La ganancia de repulsión es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta+\eta_o)}{2d_m} = 1.457507714$. Los resultados son mostrados en las Figuras 4.19 y 4.20. En la Fig. 4.19a se observan las trayectorias descritas por el AutoMiny y de los obstáculos. Los resultados obtenidos muestran que el AutoMiny se mantiene siempre a una distancia mayor a la distancia mínima permitida d, (Figuras 4.19b y 4.19c). Además, en la Figura 4.20, se observa el momento en el que el AutoMiny se encuentra en la zona de repulsión de ambos obstáculos.

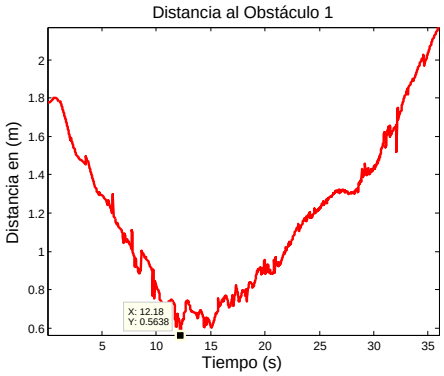
2. Trayectoria círcular: La velocidad máxima de los obstáculos 1 y 2 son $\eta_{o1} = 0.044m/s$ y $\eta_{o2} = 0.0622m/s$ respectivamente. La ganancia de repulsión es $\epsilon = 1.2\frac{(k\sqrt{2}+\eta+\eta_o)}{2d_m} = 1.443345$. Los resultados son presentados en las Figuras 4.21 y 4.22. En la Fig. 4.21a se muestran las trayectorias del AutoMiny y de los obstáculos. En estas figuras se puede ver como el AutoMiny sigue la trayectoria deseada al mismo tiempo que evita colisionar con los obstáculos. Se puede observar, en la Figura 4.22, que entre los segundos 26 al 27 el robot se encuentra dentro de la zona amarilla

de ambos obstáculos, por lo que es en este intervalo de tiempo cuando entra en acción la ley de control (3.26) con el valor de ϵ calculado. Se observa también que el robot logra evadir los obstáculos y mantenerse a la distancia mínima permitida (ver Figuras 4.21b y 4.21c).

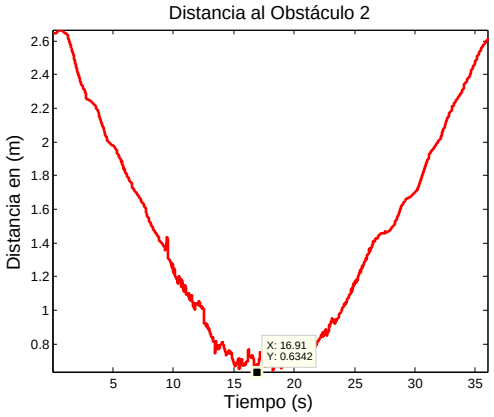
4.2 Resultados de experimentación



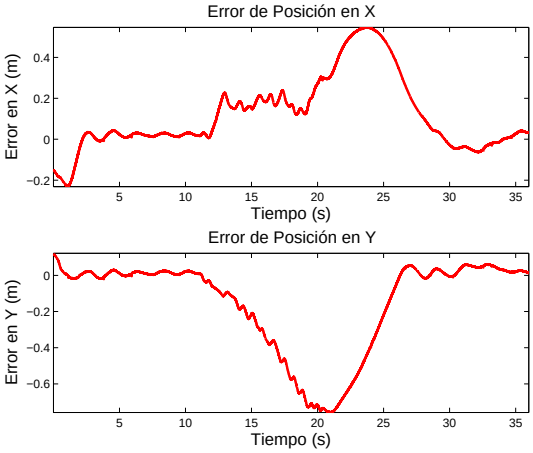
(a) Trayectoria del AutoMiny en el plano.



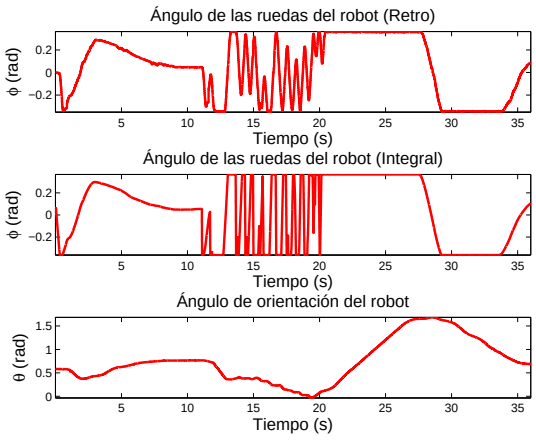
(b) Distancia entre el punto frontal y el obstáculo 1.



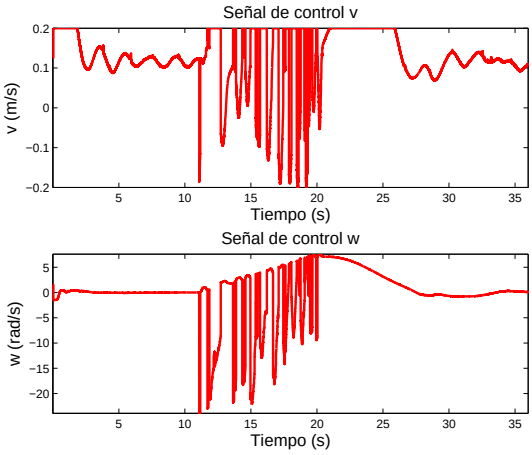
(c) Distancia entre el punto frontal y el obstáculo 2.



(d) Errores de posición del robot.



(e) Ángulos de orientación y dirección del AutoMiny.



(f) Señales de control.

Figura 4.19: Resultados experimentales trayectoria recta, dos obs. en movimiento

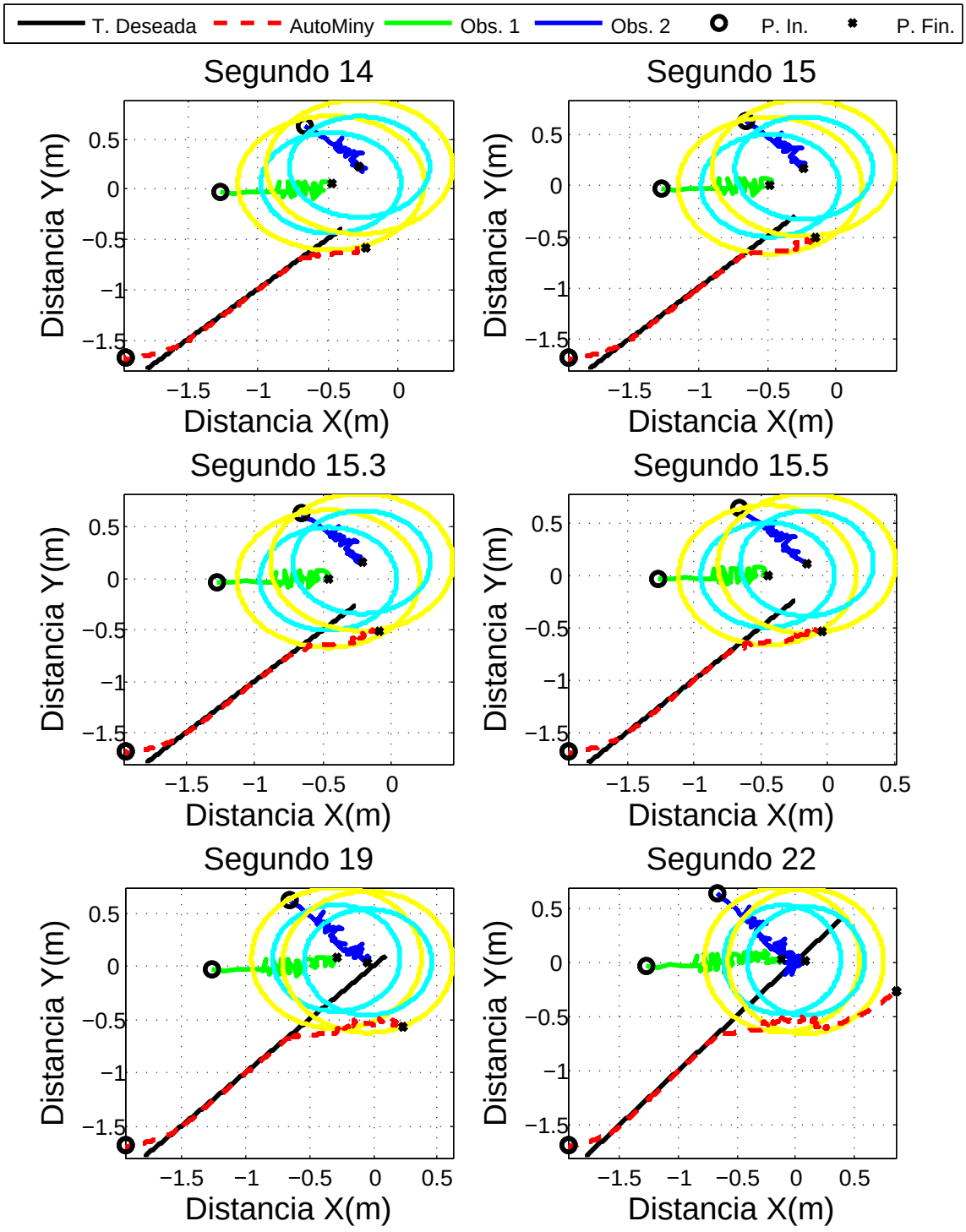
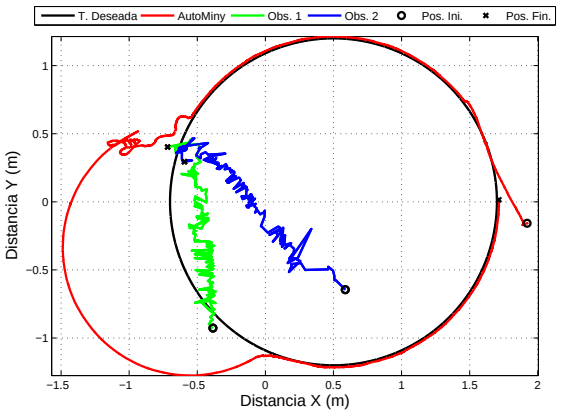
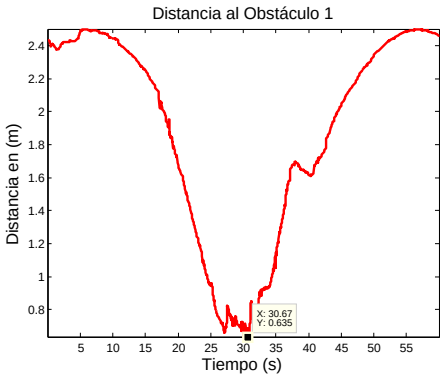


Figura 4.20: Evasión de los obs. en diferente instante de tiempo, trayectoria recta

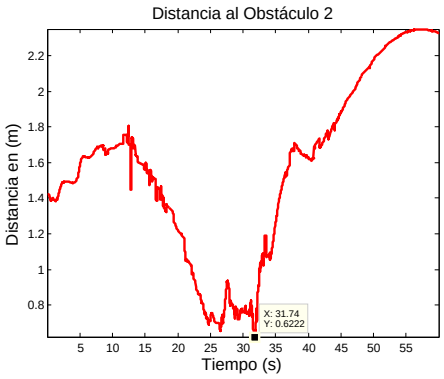
4.2 Resultados de experimentación



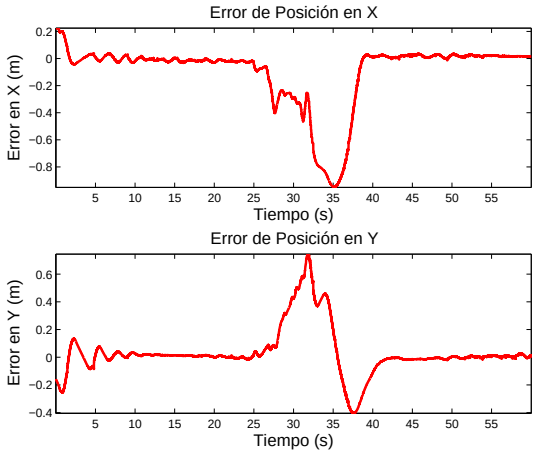
(a) Trayectoria del AutoMiny en el plano.



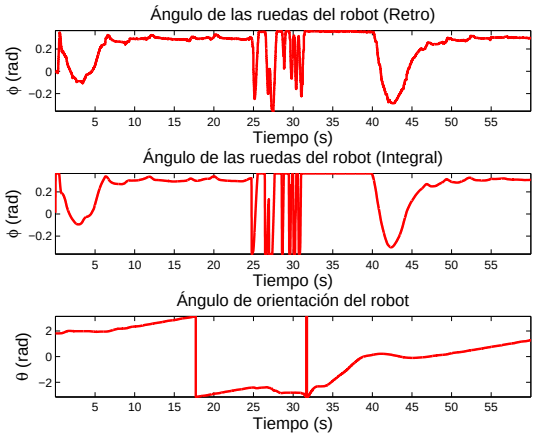
(b) Distancia entre el punto frontal y el obstáculo 1.



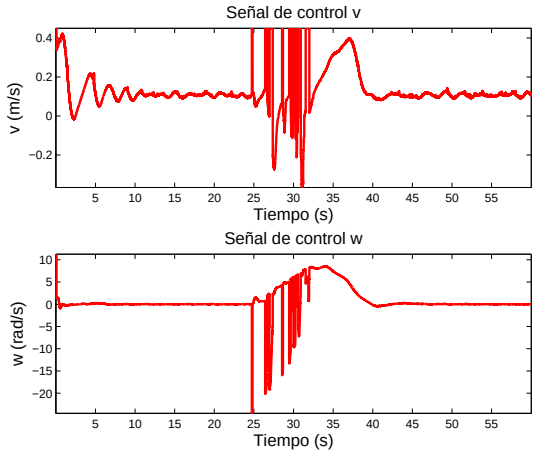
(c) Distancia entre el punto frontal y el obstáculo 2.



(d) Errores de posición del robot.



(e) Ángulos de orientación y dirección del AutoMiny.



(f) Señales de control.

Figura 4.21: Resultados experimentales trayectoria circular, dos obs. en movimiento

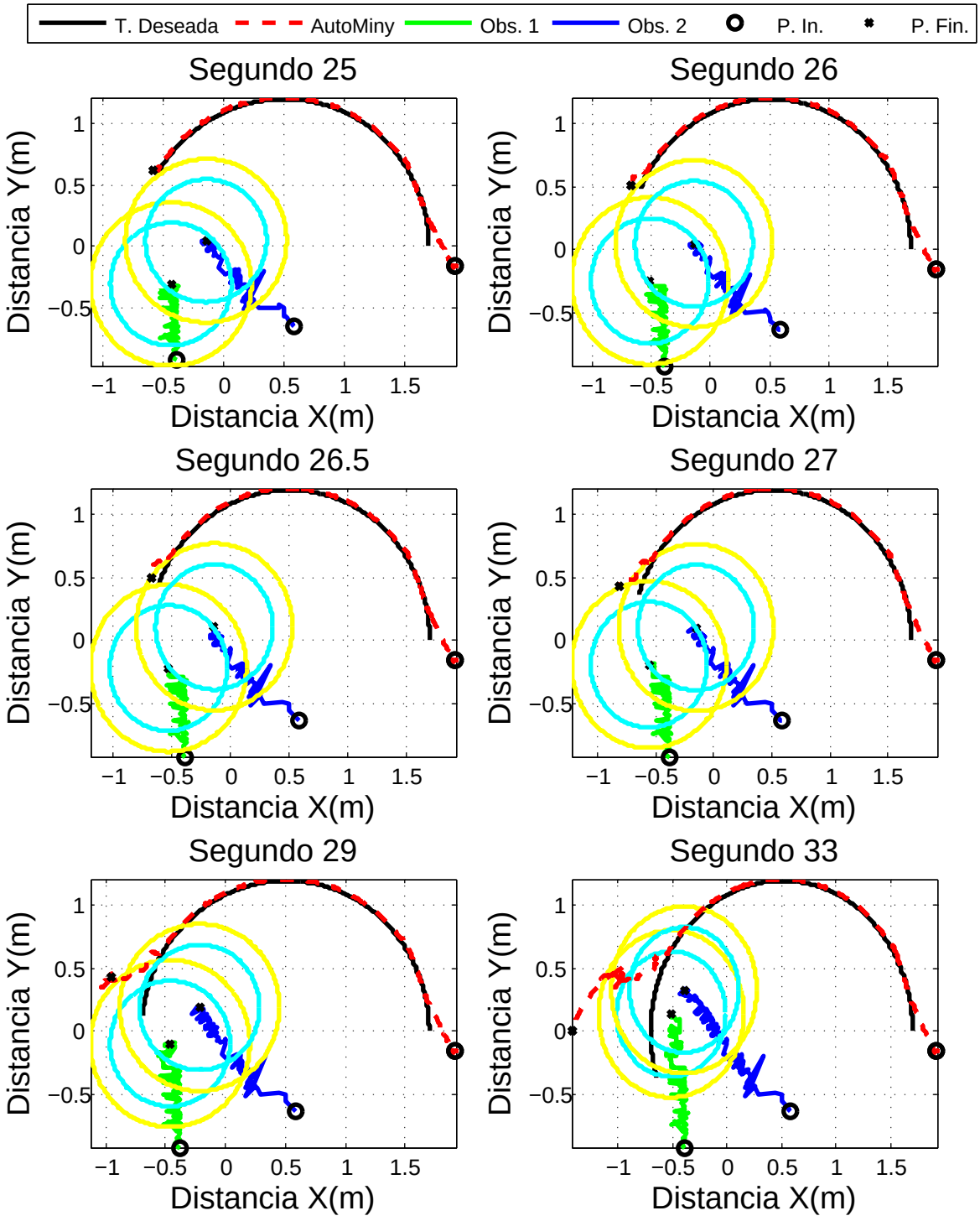


Figura 4.22: Evasión de los obs. en diferente instante de tiempo, trayectoria circular

Capítulo 5

Conclusiones

En este trabajo de tesis se presentó una estrategia de control para el seguimiento de trayectorias con evasión de colisiones contra obstáculos fijos y en movimiento, usando campos vectoriales repulsivos del tipo foco inestable en un robot AutoMiny 4.0. Se utiliza el modelo matemático del AutoMiny basado en el modelo cinemático de un robot en configuración Ackerman, considerando tracción trasera y como salida a controlar un punto P ubicado en la parte frontal del robot. Se presentó una estrategia de control, acotada mediante la función tangente hiperbólico, que garantiza la convergencia del error a cero, entre el robot y la trayectoria deseada. Se diseñó una estrategia para la evasión de obstáculos utilizando campos vectoriales repulsivos donde se garantiza la evasión de "n" obstáculos sujeto a restricciones de maniobrabilidad y espacio. Se realizó la validación numérica y experimental de los resultados teóricos. La implementación experimental se realizó mediante ROS y un sistema de cámaras Optitrack. Como se puede observar, los resultados teóricos y experimentales son consecuentes con los resultados obtenidos en la simulación numérica.

Bibliografía

[1] Quick-start guide. https://autominy.github.io/AutoMiny/docs/quickstart-guide/.

[2] Sitio web:. https://www.ros.org/blog/why-ros/.

[3] Sitio web:. Sitio web: https://optitrack.com.

[4] Sitio web:. https://www.naturalpoint.com.

[5] K. Alomari, R. Mendoza, S. Sundermann, D. Goehring, and R. Rojas. Fuzzy logic-based adaptive cruise control for autonomous model car. *In Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems (ROBOVIS 2020)*, pages 121–130, 2020.

[6] J. Andreesen. *Development of an image based traffic light detection for model cars.* PhD thesis, Freie Universität Berlin, 2020.

[7] F. P. Beer, E. R. Johnston, and P. J. Cornwell. *Mecánica Vectorial para Ingenieros. Dinámica.*, chapter Centro instantáneo de rotación en el movimiento plano, page 950. McGRAW-HILL/INTERAMERICANA, 9 edition, 2010.

[8] G. Borges-Monsreal. Autonomous navigation of an autominy based on the "sliding window technique.ªnd 2d detection. In *International Multidisciplinary Congress of Engineering*, pages 1–6. ECORFAN, jun 2021.

[9] I. Cárdenas-Millán. Programaciíon de robot móvil con prevención de colisiones coppeliasim. *Departamento de ingeniería mecánica*, 2021.

[10] J. T. Catalá. *Presente y futuro de las tecnologías verdes: Contribuciones desde la Universitat de València*, chapter El coche eléctrico: pasado, presente y futuro., pages 95–116. 2020.

[11] E. G. Diaz-Sarmiento. Generación del mapa métrico de una pista de competencias para la navegación del robot móvil autonomos mini v2. Master's thesis, Universidad Tecnológica de la Mixteca, October 2020.

[12] J. F. Flores-Resendiz and et al. Formation control with collision avoidance for first-order multi-agent systems: Experimental results. *IFAC-PapersOnLine*, 2019.

[13] L. A. García-Delgado, D. Berman-Mendozaand R. Gómez-Fuentes, B. Noriega, A. García-Juárez, A. Leal-Cruz, A. Vera-Marquina, and A. Rojas-Hernández. Función potencial repulsiva con ganancias de controlador variables. *Memorias del XVI Congreso Latinoamericano de Control Automático, CLCA 2014*, pages 1289–1294, October 2014.

[14] INEGI. Presenta inegi la georreferenciación de accidentes de tránsito en zonas urbanas, November 2021.

https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2021/accidentes/ACCIDENTES_2021.pdf.

[15] S. Jain, N. J. Ahuja, P. Srikanth, K. V. Bhadane, B. Nagaiah, A. Kumar, and C. Konstantinou. Blockchain and autonomous vehicles: Recent advances and future directions. *IEEE Access*, 9:130264–130328, 2021.

[16] R. Kelly and V. Santibáñes. *Control de Movimiento de Robots Manipuladores*. Pearson Educación. S.A., 2003. pp. 34-48.

[17] C. Kornuta, M. Cichanowski, and M. Marinelli. Inteligencia artificial aplicada a la navegación autónoma de robots móviles. *En XVII Workshop de Investigadores en Ciencias de la Computación (Salta, 2015)*, 2015.

[18] Yohana Li, Marielis Díaz, Shantall Morantes, and Yazmín Dorati. Vehículos autónomos: Innovación en la logística urbana. *Revista de Iniciación Científica*, 4(1):34–39, oct 2018.

[19] D. A. López-García. *Nuevas aportaciones en algoritmos de planificación parala ejecución de maniobras en robots autónomos no holónomos*. phdthesis, Departamento de Ingeniería Electrónica, de Sistemas Informáticos y Automática. Universidad de Huelva., July 2011.

[20] J. M. Naranjo and R. V. Angulo. *Análisis de los sistemas de protección activa y pasiva del automovil*. PhD thesis, Facultad de Mecánica Automotriz. UIDE. Quito, 2016.

[21] M. Noroña and M. Gómez. Desarrollo e innovación de los sistemas mecatrónicos en un automóvil: una revisión. *Enfoque UTE*, 10(1):117–127, March 2019.

[22] D. G. Pérez-Darquea. Evolución de los dispositivos electrónicos en un automovil. *INNOVA Research Journal*, 3(2):1–7, February 2018.

[23] H. J. Ramírez-García. Algoritmo reactivo aplicado en la evitación de obstáculos. Master's thesis, Benemérita Universidad Autónoma de Puebla, 2018.

[24] J. Santiaguillo-Salinas. *Coordinación de Movimiento con No Colisión para Sistemas Multi-agente.* phdthesis, Centro de investigación y de estudios avanzados del Instituto Politécnico Nacional, January 2017.

[25] J. Santiaguillo-Salinas and E. Aranda-Bricaire. Containment problem with time-varying formation and collision avoidance for multiagent systems. *International Journal of Advanced Robotic Systems*, June 2017.

[26] J. Santiaguillo-Salinas and E. Aranda-Bricaire. Time-varying formation tracking with collision avoidance for multi-agent systems. *IFAC-PapersOnLine*, 50:309–314, 2017.

[27] J. Santiaguillo-Salinas, H.N. Garcá-Lozano, and G. Cruz-Herrera. Modelado y control para el seguimiento de trayectorias de un robot móvil autominy 4.0. *Memorias del XXIII Congreso Mexicano de Robótica 2021*, pages 77–82, October 2021.

[28] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Dynamics and Control*. Fourth edition, January 2004.

[29] J. Stewart. *Cálculo de varias variables. Trascendentes tempranas*. Cengage Learning, octava edition, 2016.

[30] J. M. Villagrán-Escobar. Evasión de colisión para conducción autónoma en un ambiente urbano usando aprendizaje reforzado profundo y aprendizaje por representación. Master's thesis, Universidad de Chile, 2022.

Apéndice A

Instalación de software y puesta en marcha del AutoMiny

A.1. Instalación del software Motive

Para el uso de la plataforma experimental se utiliza una computadora con Windows 11 donde se intala el software Motive, el cual puede ser descargado directamente de la página de Optitrack: https://optitrack.com/support/downloads/motive.html

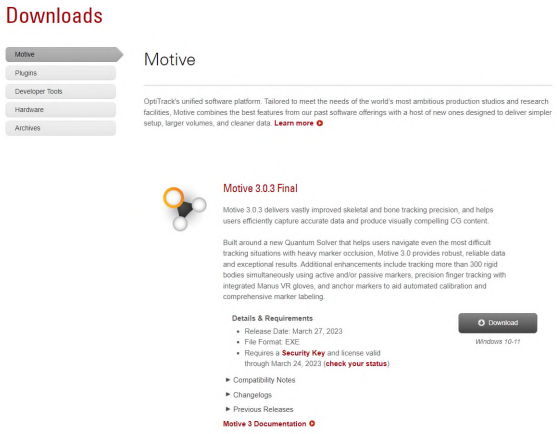


Figura A.1: Instalación del software Motive

A.2. Instalación de ROS

Para realizar la instalación de ROS en la computadora de control, se consultó la página de internet https://www.ros.org en donde seleccionamos la opción de getting-started y buscamos la distribución de ROS recomendada para nuestro sistema operativo, en nuestro caso es ROS Noetic Ninjemys; posteriomente se siguen los pasos de instalación y tutoriales para su correcta ejecución.

Figura A.2: Instalación del software ROS

A.3. Instalación y compilación del proyecto autominy

Para poder hacer uso de los paquetes y nodos con los que cuenta el autominy se descarga el proyecto a través de su página de github. Los pasos para la instalación, compilación y puesta en marcha son descritos en el enlace: https://autominy.github.io/AutoMiny/docs/installation/

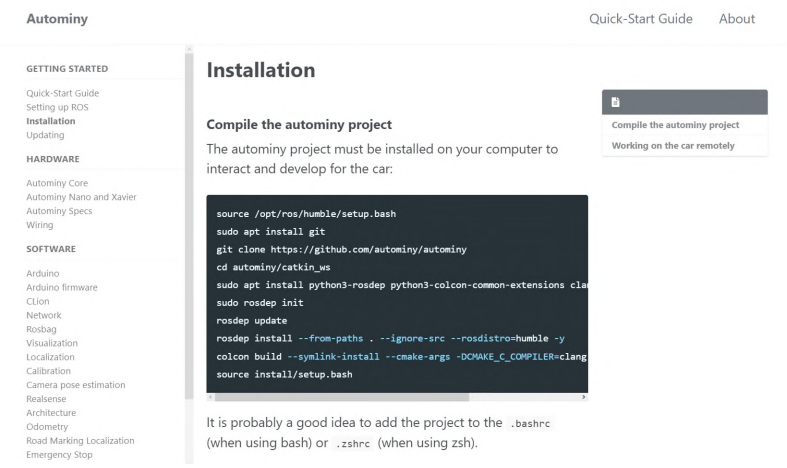


Figura A.3: Instalación del proyecto autominy

A.4. Comunicación entre ROS y las cámaras Optitrack

Para realizar la conexión entre la computadora de control y la computadora donde se recibe la información de las cámaras se hace uso de los paquetes y nodos disponibles en la wiki de ROS, llamado mocap_optitrack: http://wiki.ros.org/mocap_optitrack.

Este paquete contiene un nodo que traduce los datos de captura de movimiento de un equipo OptiTrack a tf transforms, poses y/o 2D poses. El nodo recibe paquetes que son transmitidos por una fuente compatible con NatNet, los decodifica y transmite las poses de los cuerpos rígidos configurados.

Una vez instalado el paquete, se realiza la configuración para transmitir lo datos, tanto en la computadora con Motive como en el propio paquete a travéz de un archivo .yaml donde se definen los RigidBodies que se quieren capturar.

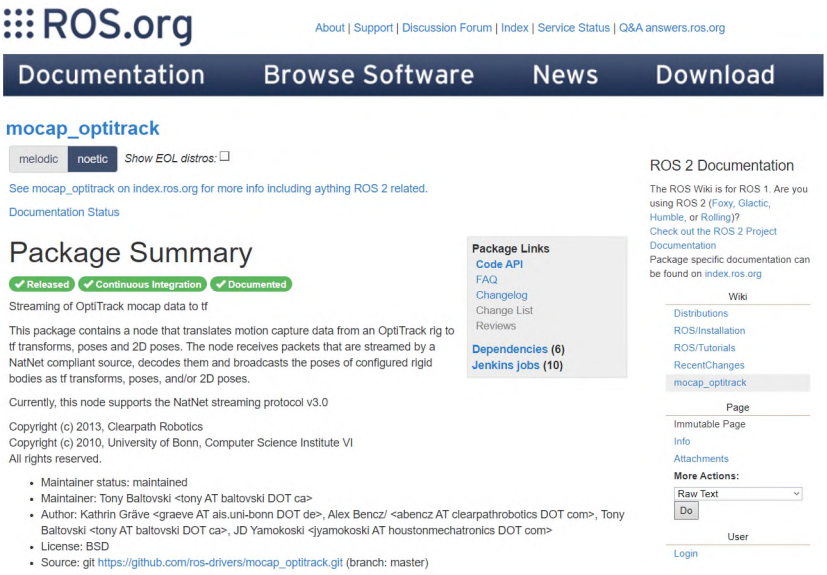


Figura A.4: Instalación del paquete mocap_optitrack

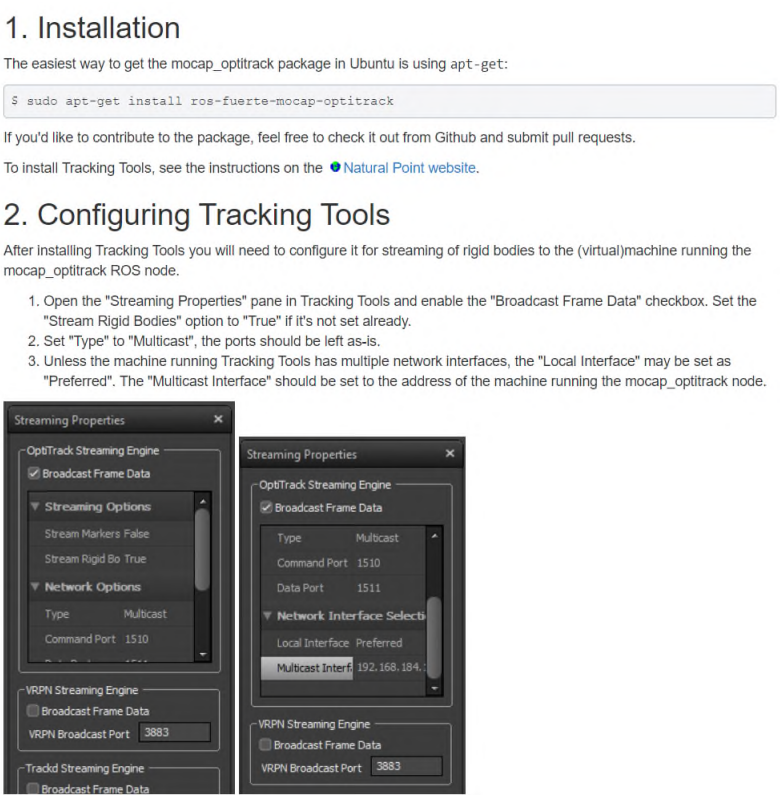


Figura A.5: Configuración del Motive

3. Configuring mocap_optitrack

Once motion capture data is being streamed to the mocap_optitrack node, the mapping of trackables to ROS topics must be defined. A sample configuration file called "mocap.yaml" is included with the package, you can find it in the config directory of the package.

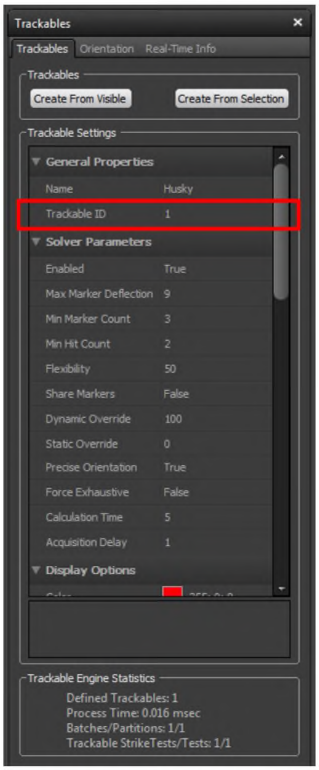
```
$ roscd mocap_optitrack/config
$ vi mocap.yaml
```

3.1 Selecting Rigid Bodies to Publish

Since multiple robots may be tracked by the OptiTrack system, the mocap_optitrack configuration allows for publishing poses and transforms for each robot separately.

First, the robot must be set up as a trackable in Tracking Tools - see the Tracking Tools documentation for how to do this.

Next, note the Trackable ID for each trackable in Tracking Tools.



The published topics for each trackable can now be configured in mocap_optitrack. Each trackable that should be published in ROS can be specified by its Trackable ID - the desired topics for the published Pose, 2D Pose and the frame_id of the transform in tf can be configured.

```
rigid_bodies:
    '1':
        pose: Robot_1/pose
        pose2d: Robot_1/ground_pose
        frame_id: Robot_1/base_link
    '2':
        pose: Robot_2/pose
```

Figura A.6: Configuración del paquete mocap_optitrack

A.5. Nodos de Control del Autominy

A continuación se muestran los nodos de control desarrollados para realizar la etapa experimentación.

1. Nodo del sensor LIDAR: En este nodo se procesa la información recibida del sensor Lidar, obteniendo así el número de obstáculos que hay en la vecindad cercana al robot, además de también determinar la posición en la que se encuentra cada uno de ellos, por último publíca la información de cada obstáculo en un topic.

```
#include "ros/ros.h"
#include "sensor_msgs/LaserScan.h"
#include <geometry_msgs/Twist.h>
#include <stdio.h>
#define RAD2DEG(x) ((x)*180./M_PI)

ros::Subscriber sub_lidar;
ros::Publisher pub_lidar;
ros::Subscriber sub_aminy;

geometry_msgs::Twist PosObs_msg;
geometry_msgs::Twist aminy_msg;

double pi = 3.141592;

//Parametros de la posicion del robot
double a1x = 0.0;
double a1y = 0.0;
double a1z = 0.0;
double a1psi = 0.0;

void aminyCallback(const geometry_msgs::Twist& msg){
  a1x = msg.linear.x;
```

```
24    a1y = msg.linear.y;
25    a1psi = msg.angular.z;
26  }
27
28  //Parametros para deteccion del obstaculo
29  double dist_lidar = 0.17;
30  double start_time = 0.0;
31  double t = 0.0;
32  int count = 0;
33  float degree = 0.0;
34  float angle_obs1 = 0.0;
35  double dist_obs1 = 0.0;
36  float angle_obs2 = 0.0;
37  double dist_obs2 = 0.0;
38  double xobs1 = 0.0;
39  double yobs1 = 0.0;
40  double xobs2 = 0.0;
41  double yobs2 = 0.0;
42  int gb1 = 0;
43  int gb2 = 0;
44
45  //Variable para encontrar la distancia minima y el angulo
46  int flag1 = 0;
47  int flag2 = 0;
48  double dmin1 = 0.0;
49  double dmin2 = 0.0;
50  double angle_t = 0.0;
51
52  FILE *destino;
53
54  void scanCallback(const sensor_msgs::LaserScan::ConstPtr& scan){
55      count = scan->scan_time / scan->time_increment;
56      ROS_INFO("I heard a laser scan %s[%d]:", scan->header.frame_id.
    c_str(), count);
57      ROS_INFO("angle_range, %f, %f", RAD2DEG(scan->angle_min),
    RAD2DEG(scan->angle_max));
```

```
58
59      dmin1 = scan->ranges[0];
60
61      for(int i = 1; i < count; i++) {
62          degree = RAD2DEG(scan->angle_min + scan->angle_increment * i
    );
63          if(scan->ranges[i]<=7 && scan->ranges[i]>=0.25){
64              flag1 = 1;
65              if(scan->ranges[i]<dmin1){
66                  dmin1 = scan->ranges[i];
67                  gb1 = i;
68                  angle_obs1 = i * pi / 180;
69              }
70          }
71      }
72
73      dist_obs1 = dmin1;
74      if (gb1==0){
75        dmin2 = scan->ranges[12];
76        gb2 = 12;
77      }else{
78        dmin2 = scan->ranges[0];
79      }
80
81      for(int u = 0; u < count; u++) {
82          degree = RAD2DEG(scan->angle_min + scan->angle_increment * u
    );
83          if(scan->ranges[u]<=7 && scan->ranges[u]>=0.25){
84              flag2 = 1;
85              if(scan->ranges[u]>dist_obs1 && scan->ranges[u]<=dmin2){
86                if(u>=gb1+12 || u<=gb1-12){
87                      dmin2 = scan->ranges[u];
88                      angle_obs2 = u * pi / 180;
89                 }
90              }
91          }
```

```
92      }
93    dist_obs2 = dmin2;
94
95      //Posicion del obstaculo 1
96      if (flag1 == 1){
97        xobs1 = a1x+dist_lidar*cos(a1psi)+dist_obs1*cos(a1psi+
      angle_obs1);
98        yobs1 = a1y+dist_lidar*sin(a1psi)+dist_obs1*sin(a1psi+
      angle_obs1);
99        flag1 = 0;
100     }
101     //Posicion del obstaculo 2
102     if (flag2 == 1){
103       xobs2 = a1x+dist_lidar*cos(a1psi)+dist_obs2*cos(a1psi+
      angle_obs2);
104       yobs2 = a1y+dist_lidar*sin(a1psi)+dist_obs2*sin(a1psi+
      angle_obs2);
105       flag2 = 0;
106     }
107 }
108
109 int main(int argc, char **argv){
110 destino = fopen("obstaculos.txt","w");
111     ROS_INFO("AutoMiny. Prueba lidar");
112     ros::init(argc, argv, "lidar_data");
113     ros::NodeHandle n;
114
115     //suscribers
116     sub_aminy = n.subscribe("/optiR1", 1, aminyCallback);
117     sub_lidar = n.subscribe<sensor_msgs::LaserScan>("/sensors/
      rplidar/scan", 1000, scanCallback);
118     //publishers
119     pub_lidar = n.advertise<geometry_msgs::Twist>("/PosObs", 1);
120
121     ros::Rate loop_rate(100);
122     while(ros::ok()){
```

```
123         start_time =(double)ros::Time::now().toSec();
124         PosObs_msg.linear.x = xobs1;
125         PosObs_msg.linear.y = yobs1;
126         PosObs_msg.linear.z = angle_obs1;
127         PosObs_msg.angular.x = xobs2;
128         PosObs_msg.angular.y = yobs2;
129         PosObs_msg.angular.z = angle_obs2;
130         pub_lidar.publish(PosObs_msg);
131         t = (double)ros::Time::now().toSec() - start_time;
132         ROS_INFO("Posicion Obs 1:[%f, %f]", xobs1, yobs1);
133         ROS_INFO("Posicion Obs 2:[%f, %f]", xobs2, yobs2);
134         fprintf(destino,"%lf %lf %lf %lf\n", xobs1, yobs1, xobs2,
    yobs2);
135         ros::spinOnce();
136         loop_rate.sleep();
137     }
138
139     return 0;
140 }
```

2. Nodo de Optitrack: Recibe la información obtenida del sistema de cámaras optitrack, la cual es la posición y orientación del punto medio del eje de las ruedas traseras del robot. Posteriormente es publicada la información en un topic.

```
1  #include "ros/ros.h"
2  #include "geometry_msgs/PoseStamped.h"
3  #include "tf/transform_datatypes.h"
4  #include <geometry_msgs/Twist.h>
5
6  #define RAD2DEG(x) ((x)*180./M_PI)
7
8  double start_time = 0.0;
9  double t = 0.0;
10
```

```
double r1x = 0.0;
double r1y = 0.0;
double r1z = 0.0;
double r1qx = 0.0;
double r1qy = 0.0;
double r1qz = 0.0;
double r1qw = 0.0;

tf::Quaternion r1q;
double r1roll = 0.0;
double r1pitch = 0.0;
double r1yaw = 0.0;

ros::Subscriber sub_optir1;
ros::Publisher pub_optir1;

geometry_msgs::Twist optir1_msg;

void optiCallback(const geometry_msgs::PoseStamped msg){

    r1x = -msg.pose.position.y;
    r1y = msg.pose.position.x;
    r1z = msg.pose.position.z;
    r1qx = -msg.pose.orientation.x;
    r1qy = -msg.pose.orientation.y;
    r1qz = msg.pose.orientation.z;
    r1qw = msg.pose.orientation.w;

    r1q = tf::Quaternion(r1qx,r1qy,r1qz,r1qw);
    tf::Matrix3x3(r1q).getRPY(r1roll, r1pitch, r1yaw);
}

int main(int argc, char **argv){

    ROS_INFO("AutoMiny. Prueba Optitrack");
    ros::init(argc, argv, "optitrack_data");
```

```
47      ros::NodeHandle n;
48
49      sub_optir1 = n.subscribe<geometry_msgs::PoseStamped>("/
        mocap_node/Autominy/pose", 1000, optiCallback);
50      pub_optir1 = n.advertise<geometry_msgs::Twist>("/optiR1", 1);
51
52      ROS_INFO("Iniciando prueba Optitrack...");
53
54      ros::Rate loop_rate(100);
55      while(ros::ok()){
56          start_time =(double)ros::Time::now().toSec();
57          optir1_msg.linear.x = r1x;
58          optir1_msg.linear.y = r1y;
59          optir1_msg.linear.z = r1z;
60          optir1_msg.angular.x = r1roll;
61          optir1_msg.angular.y = r1pitch;
62          optir1_msg.angular.z = r1yaw;
63          pub_optir1.publish(optir1_msg);
64
65          ROS_INFO("Posicion y orientacion del autominy");
66          ROS_INFO("x: %lf", r1x);
67          ROS_INFO("y: %lf", r1y);
68          ROS_INFO("z: %lf", r1z);
69          ROS_INFO("roll: %lf", RAD2DEG(r1roll));
70          ROS_INFO("pich: %lf", RAD2DEG(r1pitch));
71          ROS_INFO("yaw: %lf", RAD2DEG(r1yaw));
72          t = (double)ros::Time::now().toSec() - start_time;
73          ROS_INFO("tiempo de muestreo, %lf", t);
74
75          ros::spinOnce();
76          loop_rate.sleep();
77      }
78
79      ROS_INFO("Prueba Optitrack finalizada.");
80      return 0;
81  }
```

3. Nodo de Control: En este nodo se procesa la información obtenida de los dos nodos anteriores para así desarrollar la estrategia de control y obtener las señales de control (velocidad lineal y ángulo de dirección) para así enviarlas al robot.

```
#include "ros/ros.h"
#include "autominy_msgs/SpeedCommand.h"
#include "autominy_msgs/SteeringCommand.h"
#include "autominy_msgs/SteeringAngle.h"
#include <geometry_msgs/Twist.h>
#include <stdio.h>
#include "sensor_msgs/LaserScan.h"

ros::Subscriber sub_aminy;
ros::Subscriber sub_angle;
ros::Publisher pub_speed;
ros::Publisher pub_steering;
ros::Subscriber sub_lidar;

autominy_msgs::SpeedCommand speed_cmd;
autominy_msgs::SteeringCommand steering_cmd;
geometry_msgs::Twist aminy_msg;

// Variables de posicion del robot
double a1x = 0.0;
double a1y = 0.0;
double a1z = 0.0;
double a1phi = 0.0;
double a1psi = 0.0;

//Variables de posicion del obstaculo 1
double xobs1 = 0.0;
double yobs1 = 0.0;
double angle_obs1 =0.0;

```

```
31 //Variables de posicion del obstaculo 2
32 double xobs2 = 0.0;
33 double yobs2 = 0.0;
34 double angle_obs2 =0.0;
35
36 void aminyCallback(const geometry_msgs::Twist& msg){
37   a1x = msg.linear.x;
38   a1y = msg.linear.y;
39   a1z = msg.linear.z;
40   a1psi = msg.angular.z;
41 }
42
43 void angleCallback(const autominy_msgs::SteeringAngle angle){
44     a1phi = angle.value;
45     ROS_INFO("Angulo de las ruedas: %lf", a1phi);
46 }
47
48 void lidarCallback(const geometry_msgs::Twist& msg){
49   xobs1 = msg.linear.x;
50   yobs1 = msg.linear.y;
51   angle_obs1 = msg.linear.z;
52   xobs2 = msg.angular.x;
53   yobs2 = msg.angular.y;
54   angle_obs2 = msg.angular.z;
55 }
56
57 int main(int argc, char **argv){
58
59     double pi = 3.141592;
60
61     //trayectoria
62     double r = 1.2;
63     double Ti = 60.0;
64     double vel = 0.166;
65     double xd = 0.0;
66     double yd = 0.0;
```

```
67      double xdp = 0.0;
68      double ydp = 0.0;
69
70      //parametros para la evasion
71      double deltamin1 = 0.0;
72      double dist_obs1 = 0.0;
73      double deltamin2 = 0.0;
74      double dist_obs2 = 0.0;
75      double dist_min = 0.7;
76      double tam = 0.0;
77      double epsi1 = 0.0;
78      double epsi2 = 0.0;
79      double beta_x1 = 0.0;
80      double beta_y1 = 0.0;
81      double beta_x2 = 0.0;
82      double beta_y2 = 0.0;
83      double eta = 0.1257;
84      double etao1 = 0.063;
85      double etao2 = 0.05;
86
87      //parametros del robot
88      double px = 0.0;
89      double py = 0.0;
90      double delta = 0.1;
91      double ell = 0.26;
92      double anglei = 0.0;
93      double velmax = 0.45;//2.5;
94      double velmin = -0.45;//-2.5;
95      double anglewmax = 0.367;
96      double anglewmin = -0.362;
97
98      //parametros del control
99      double rx = 0.0;
100     double ry = 0.0;
101     double k = 1.0;
102     //double ky = 1.0;
```

```
103     double a1v = 0.0;
104     double a1w = 0.0;
105 
106     //variables del experimento
107     double start_time = 0.0;
108     double t = 0.0;
109     double periodo = 0.001;
110     double other_time = 0.0;
111     double muestreo = 0.0;
112     int count = 0;
113     float degree1 = 0.0;
114     float degree2 = 0.0;
115     //se crea archivo txt
116     FILE *destino;
117 
118     ROS_INFO("AutoMiny. Prueba Control");
119     ros::init(argc, argv, "control_data");
120     ros::NodeHandle n;
121 
122     //subscribers
123     sub_aminy = n.subscribe("/optiR1", 1, aminyCallback);
124     sub_angle = n.subscribe<autominy_msgs::SteeringAngle>("/sensors/
    steering", 1000, angleCallback);
125     sub_lidar = n.subscribe("/PosObs", 1, lidarCallback);
126     //publish
127     pub_speed = n.advertise<autominy_msgs::SpeedCommand>("/actuators
    /speed", 1000);
128     pub_steering = n.advertise<autominy_msgs::SteeringCommand>("/
    actuators/steering", 1000);
129 
130     //abrimos archivo txt
131     destino = fopen("dataminy.txt","w");
132 
133     ROS_INFO("Iniciando programa Autominy...");
134 
135     //se inicia el angulo en cero y velocidad cero
```

```
136     speed_cmd.value = 0.0;
137     pub_speed.publish(speed_cmd);
138     steering_cmd.value = 0.0;
139     pub_steering.publish(steering_cmd);
140 
141     //se inicia el experimento
142     start_time =(double)ros::Time::now().toSec();
143     ros::Rate loop_rate(1000);
144     while(ros::ok()){
145         t = (double)ros::Time::now().toSec() - start_time;
146         other_time =(double)ros::Time::now().toSec();
147 
148         if(t <=60 ){
149 
150             //Inicial
151             deltamin1 = 0.0;
152             deltamin2 = 0.0;
153 
154       //Trayectoria
155       //circulo
156       xd = r*cos((2*pi*t)/Ti) + 0.5;
157       yd = r*sin((2*pi*t)/Ti);
158       xdp = -r*((2*pi)/Ti)*sin((2*pi*t)/Ti);
159       ydp = r*((2*pi)/Ti)*cos((2*pi*t)/Ti);
160       //recta
161       //xd = 0.1*t-1.8;
162       //yd = xd;
163      // xdp = 0.1;
164      // ydp = 0.1;
165       //Punto P
166       px = a1x + ell*cos(a1psi) + delta*cos(a1psi + a1phi);
167       py = a1y + ell*sin(a1psi) + delta*sin(a1psi + a1phi);
168 
169       //distancia a los obstaculo
170       dist_obs1 = sqrt(((px-xobs1)*(px-xobs1))+((py-yobs1)*(py-yobs1
    )));
```

```
171     dist_obs2 = sqrt(((px-xobs2)*(px-xobs2))+((py-yobs2)*(py-yobs2
    )));
172
173     //calculo de epsilon
174     epsi1 = 1.1*(k*sqrt(2)+eta+etao1)/dist_min;
175     epsi2 = 1.1*(k*sqrt(2)+eta+etao2)/dist_min;
176
177     //calculo de beta
178     if(dist_obs1 <= dist_min){
179       deltamin1 = 1.0;
180     }
181     if(dist_obs2 <= dist_min){
182       deltamin2 = 1.0;
183     }
184
185     //calculo de epsilon si hay dos obstaculos
186     if(deltamin1 == 1.0 && deltamin2 == 1.0){
187       epsi1 = 1.2*(k*sqrt(2)+eta+etao1)/(2*dist_min);
188       epsi2 = 1.2*(k*sqrt(2)+eta+etao1)/(2*dist_min);
189     }
190
191     //calculo de beta
192           beta_x1 = epsi1*deltamin1*((px-xobs1)-(py-yobs1));
193           beta_y1 = epsi1*deltamin1*((px-xobs1)+(py-yobs1));
194     beta_x2 = epsi2*deltamin2*((px-xobs2)-(py-yobs2));
195           beta_y2 = epsi2*deltamin2*((px-xobs2)+(py-yobs2));
196
197     //control auxiliar
198     rx = -k*tanh(px - xd) + xdp + beta_x1 + beta_x2;
199     ry = -k*tanh(py - yd) + ydp + beta_y1 + beta_y2;
200
201     //control
202     a1v = (cos(a1phi)/delta)*(delta*cos(a1psi+a1phi))*rx +  (cos(
    a1phi)/delta)*(delta*sin(a1psi+a1phi))*ry;
```

```
203     a1w = (cos(a1phi)/delta)*(-sin(a1psi)-tan(a1phi)*(cos(a1psi)+
  delta*cos(a1psi+a1phi)/ell))*rx + (cos(a1phi)/delta)*(cos(a1psi)
  -tan(a1phi)*(sin(a1psi)+delta*sin(a1psi+a1phi)/ell))*ry;
204     anglei = anglei + (periodo*a1w);
205
206     //saturacion
207     if(a1v >= velmax){a1v = velmax;}
208     if(a1v <= velmin){a1v = velmin;}
209     if(anglei >= anglewmax){anglei = anglewmax;}
210     if(anglei <= anglewmin){anglei = anglewmin;}
211
212     //envio de datos
213     speed_cmd.value = a1v;
214     pub_speed.publish(speed_cmd);
215     steering_cmd.value = anglei;
216     pub_steering.publish(steering_cmd);
217
218     //guardar en archivo
219     fprintf(destino,"%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %
  lf %lf %lf %lf %lf\n", xd, yd, a1x, a1y, px, py, a1v, a1w, a1phi
  , anglei, a1psi, t, xobs1, yobs1, xobs2, yobs2);
220   }
221
222   if(t > 60){
223     break;
224   }
225
226       ROS_INFO("px: %lf", px);
227       ROS_INFO("py: %lf", py);
228       ROS_INFO("phi: %lf", a1phi);
229       ROS_INFO("psi: %lf", a1psi);
230       ROS_INFO("tiempo: %lf", t);
231
232       muestreo = (double)ros::Time::now().toSec() - other_time;
233       ROS_INFO("muestreo: %lf", muestreo);
234
```

```
235         ros::spinOnce(); //refresca los datos
236         loop_rate.sleep(); //espera a que se cumpla la frecuencia
237     }
238 
239     speed_cmd.value = 0.0;
240     pub_speed.publish(speed_cmd);
241     steering_cmd.value = 0.0;
242     pub_steering.publish(steering_cmd);
243 
244     fclose(destino);
245 
246     ROS_INFO("Programa Autominy finalizado");
247 
248     return 0;
249 }
```

A.6. Puesta en marcha del robot

Para poner en marcha el robot se escriben las lineas que aparecen en la siguiente imagen. Donde en las primeras dos líneas se configuran el ROS Master que se encuentra en el robot y la IP de la computadora de control. Posteriormente abrimos la carpeta del espacio de trabajo donde se encuentran los paquetes del proyecto y se identifica la carpeta con el "setup.bash". Por último se lanzan los nodos a usar, para el mocap se utiliza "roslaunch" como se muestra en la Fig. A.7, mientras que para los demás se hace por medio de "rosrun" seguido del nombre del nodo. Lo anterior se repite en cada una de las terminales abiertas para cada nodo.

A.6 Puesta en marcha del robot

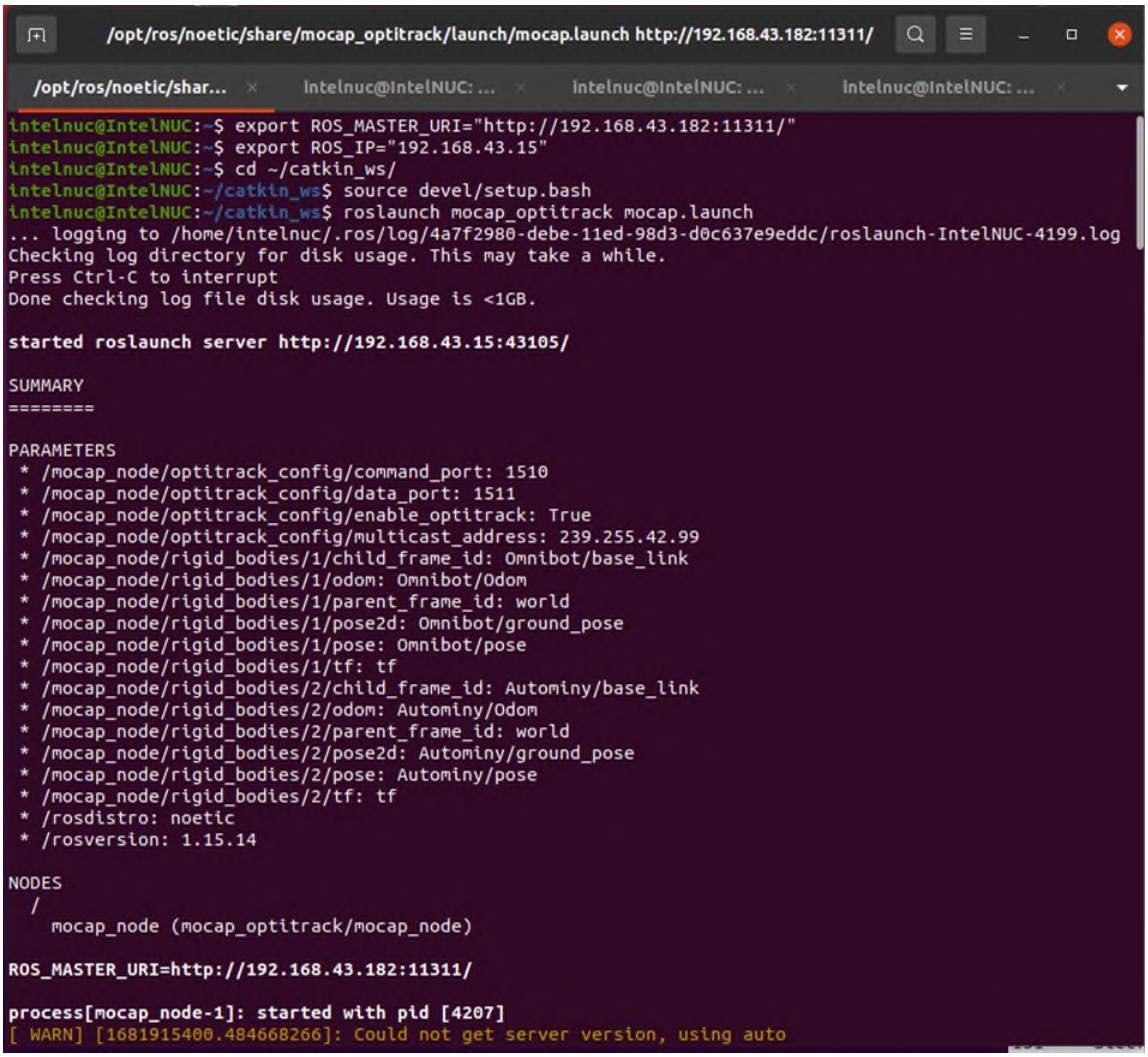


Figura A.7: Ejecución de los nodos de ROS

Apéndice B

Artículos publicados

A continuación se presentan los artículos de investigación elaborados a partir del desarrollo del presente trabajo de tesis.

- Hernández-Montalvo, D. G., Santiaguillo-Salinas, J. (2022) Diseño e implementación de una estrategia de control acotada para el seguimiento de trayectorias considerando evasión de colisiones utilizando RVF para el AutoMiny 4.0. 22 Congreso Internacional de Ciencias de la Computación CORE 2022.

- Santiaguillo-Salinas, J., García-Lozano, H. N., González-Zárate, R. F., & Hernández-Montalvo, D. G. (2022). Seguimiento de trayectorias con evasión de colisiones para un robot autominy 4.0. Pädi Boletín Científico De Ciencias Básicas E Ingenierías Del ICBI, 10(Especial5), 8-14. https://doi.org/10.29057/icbi.v10iEspecial5.10125

- Santiaguillo-Salinas, J., Hernández-Montalvo, D. G., García-Lozano, H.,

González-Zárate, R. F. 2020. Non-collision strategy with RVF for n moving obstacles with LIDAR detection in trajectory tracking for an AutoMiny 4.0. Artículo terminado, por someter.