UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO Y GENERACIÓN DE TRAYECTORIAS AVANZADAS PARA EL CONTROL DE ROBOTS MÓVILES

AUTOR **JORGE HERRERA CRESPO**

TUTOR: ÁNGEL VALERA FERNÁNDEZ

Curso Académico: 2018-2019

ÍNDICE

Documento 1: MEMORIA 7
Documento 2: PRESUPUESTO 79

ÍNDICE DE LA MEMORIA

1. INTRODUCCIÓN 9
1.1. Motivación 9
1.2. Descripción de los Objetivos del Proyecto 10
2. DESARROLLO TEÓRICO 12
2.1. Introducción a la Robótica 12
2.2. Tipos de Robots 13
2.2.1. Robots Industriales 13
2.2.2. Robots de Servicio 14
2.2.3. Robots Manipuladores 14
2.2.4. Robots Móviles 15
2.3. Robótica Móvil 16
2.4. Sistemas de Control 22
2.5. Curvas de Transición 30
2.5.1. Curvas Compuestas 31
2.5.2. Parábola Cúbica 31
2.5.3. Espiral Cúbica 31
2.5.4. Curva de Transición de Schramm 32
2.5.5. Curva Elástica 32
2.5.6. Lemniscata de Bernoulli 33
2.5.7. Clotoide. 34
2.6. Lego NXT 43
2.7. Robot C 44
2.8. Navegación 38
2.8.1. Hoja de Ruta 38
2.8.2. Descomposición en Celdas 39
2.8.3. Algoritmo Insecto 40

2.8.4. Campos Potenciales Artificiales 41
3. DESARROLLO APLICADO 45
3.1. Soluciones Elegidas 45
3.2. Diseño del Sistema de Control 46
3.3. Diseño y Generación de Clotoides 50
3.3.1. Clotoide Recta-Círculo 50
3.3.2. Clotoide Círculo Recta 55
3.3.3. Caso Aplicado a una Situación de la Vida Real. Cambio de Sentido en Rotonda. 56
3.3.4. Generalización de las Clotoides 60
3.4. Desarrollo de Trayectorias mediante Campos Potenciales 63
3.4.1. Definición de los Elementos del Entorno. 63
3.4.2. Construcción de los Campos Potenciales 66
3.4.3. Simulación. 70
3.4.4. Generalización del Algoritmo de Campos Potenciales Artificiales. 72
3.4.5. Problemas de los Campos Potenciales Artificiales. 74
4. CONCLUSIONES 76
5. REFERENCIAS 78

ÍNDICE DEL PRESUPUESTO

1. PROGRAMACIÓN 81
2. SIMULACIÓN 82
3. REDACCIÓN 82
4. PRESUPUESTO PARCIAL 83
5. PRESUPUESTO BASE 83

ÍNDICE DE FIGURAS

Figura 2.1 Robot Industrial. 13
Figura 2.2 Robot de Servicio 14
Figura 2.3 Robot Manipulador 15
Figura 2.4 Robot Móvil 15
Figura 2.5. Machina Speculatrix 16
Figura 2.6. Bestia 16
Figura 2.7 Stanford Cart. 17
Figura 2.8 Robot Doméstico RB5X 17
Figura 2.9. Dante II 18
Figura 2.10. Vehículo VaMP 18
Figura 2.11 Roomba 19
Figura 2.12 Robot Teleoperado ANATROLLER ARI-100. 20
Figura 2.13 AGV Egemin 21
Figura 2.14 Robots Guiados de Forma Autónoma TUG 21
Figura 2.15. Autonomía Deslizante PatrolBot 22
Figura 2.16 Configuración Diferencial 23
Figura 2.17 Configuración de Oruga 23
Figura 2.18 Configuración Triciclo. 24
Figura 2.19 Configuración Ackerman 24
Figura 2.20. Bucle de Control. 25
Figura 2.21. Curvas Compuestas 31
Figura 2.22 Curva Elástica 32
Figura 2.23. Lemniscata de Bernoulli 33
Figura 2.24. Clotoide 36
Figura 2.25. Proporcionalidad de Clotoides 37
Figura 2.26. Hoja de Ruta 39
Figura 2.27 Descomposición en Celdas 40
Figura 2.28 Algoritmo Insecto1 (izq.) Algoritmo Insecto2 (der.) 40
Figura 2.29 Campos Potenciales Artificiales 42
Figura 2.30 Lego NXT 43
Figura 2.31. Robot C. 44
Figura 3.1. Control Cinemático Directo 46
Figura 3.2 Control Cinemático Inverso. 47
Figura 3.3 Simulación Kmp=1, Krp=1 Y Krv=0.1. 48
Figura 3.4 Determinación de Kmp. 48
Figura 3.5 Determinación de Krp. 48
Figura 3.6 Parámetros de la clotoide 51
Figura 3.7. Primera Referencia Clotoide 53
Figura 3.8 Generación Trayectoria Clotoide. 54
Figura 3.9 Clotoide que une un tramo con otro de mayor radio. 55
Figura 3.10. Clotoides en Rotonda 57
Figura 3.11 Señal del robot en X (rojo) vs. referencia en X (azul) 58

Figura 3.12 Señal del robot en Y (rojo) vs. referencia en Y (azul) 58
Figura 3.13 Error en X 58
Figura 3.14 Error en Y 58
Figura 3.15 Acción de Control de la Rueda Derecha. 59
Figura 3.16. Acción de Control de la Rueda Izquierda 59
Figura 3.17 Definición del Entorno. 64
Figura 3.18 Definición Obstáculos Fijos. 64
Figura 3.19 Definición de Punto Inicial y Meta. 65
Figura 3.20 Definición Obstáculos Móviles. 65
Figura 3.21 Campos Potenciales 4 Obstáculos. 68
Figura 3.22 Campos Potenciales Trayectoria en S. 68
Figura 3.23 Obstáculos Móviles. 70
Figura 3.24. Representación del Campo Potencial. 72
Figura 3.25 Representación de las Fuerzas Potenciales. 73
Figura 3.26 Representación en 3D. 73
Figura 3.27 Acción de Control de la Rueda Derecha. 74
Figura 3.28 Acción de Control de la Rueda Izquierda. 74
Figura 3.29 Robot Estancado. 75
Figura 3.30 Solución al Estancamiento. 75

ÍNDICE DE TABLAS

Tabla 1 Parámetros de control. 49
Tabla 2 Costes Programación. 81
Tabla 3 Costes Simulación. 82
Tabla 4. Costes Redacción. 82
Tabla 5 Presupuesto Parcial. 83
Tabla 6 Presupuesto Total. 83

Documento 1:
MEMORIA

1. INTRODUCCIÓN

1.1. Motivación

En estos días, en los que la huella digital está presente en casi todos los ámbitos de la vida, y día a día se convive con *máquinas programadas* que permiten a las personas disfrutar de una vida más fácil y cómoda, resulta muy necesario ser capaz de conseguir que dichas máquinas puedan realizar con éxito la tarea para la cual fueron diseñadas y construidas.

Dentro del grupo de las máquinas programadas, este Trabajo Fin de Grado se va a centrar en ***los robots móviles.*** Ideados en su momento con el fin de poder transportar, tanto personas como materiales, y que cada vez son más importantes en el ámbito industrial, aumentando el rendimiento de la producción, y proporcionando un movimiento óptimo del producto por los distintos sectores del proceso productivo. Además de dicho uso, han sido también capaces de lograr una gran popularidad gracias a sus aplicaciones en la exploración espacial, los vehículos autónomos o los vehículos cuya aplicación ha trascendido a múltiples utilidades económicas, empresariales y de ocio, como los drones.

Conseguir que los robots móviles sean capaces de realizar sus tareas de forma autónoma, es un nuevo, pero gran, avance de cara al constante progreso en la automatización industrial. Produciendo grandes beneficios para la industria, tanto económicos, como de calidad en los productos elaborados.

Si se pretende que los robots autónomos circulen por recintos cerrados, donde el entorno está prácticamente controlado, e incluso que más adelante circulen por espacios abiertos, en ambientes totalmente impredecibles, estos deben ser capaces de generar trayectorias tales que garanticen su propia seguridad, la de aquello que transportan, y de todo aquello que pueda encontrarse en su entorno.

De este modo, las trayectorias que siga el robot deben ser lo más suaves posibles, haciendo así que el viaje sea más cómodo y seguro para los ocupantes del vehículo, o para los materiales que transporta y contiene. Además, las trayectorias deben ser lo más flexibles posible, para poder evitar cualquier obstáculo que se interponga en el camino del robot.

La finalidad de este proyecto es académica. Se pretende llevar a cabo un análisis del problema a resolver, y una propuesta de solución a dicho problema, a través de un Trabajo Fin de Grado que cumple las indicaciones de las “Directrices sobre la presentación de Trabajos Final de Grado en la Escuela Técnica Superior de Ingenieros Industriales de Valencia”.

1.2. Descripción de los Objetivos del Proyecto

Este proyecto que recibe el nombre: "Diseño y generación de trayectorias avanzadas para el control de robots móviles", ha sido desarrollado con la finalidad de obtener una navegación en la que los caminos, tanto prediseñados fuera de línea, como aquellos generados por el propio robot a medida que avanza, permitan que el desplazamiento sea lo más seguro posible.

Para lograrlo, serán necesarios dos hitos importantes, conseguir que el modo de conducción sea lo más cómodo posible, descomponiendo el circuito en tramos sencillos unidos por elementos que proporcionen suavidad a la trazada, y que sea capaz de reconocer la posición de los obstáculos presentes en su entorno para así poder evitarlos.

Por tanto los objetivos operativos que nos van a permitir cumplir con los objetivos generales del proyecto son:

- Respecto a la suavidad de la trazada:
  - Se pretende modelar los diferentes tramos del circuito por separado. Estos podrán ser representados mediante tramos rectos o tramos circulares.
  - A partir de ahí, se pretende generar y diseñar la mejor solución técnica que permita unir los tramos rectilíneos con los tramos circulares.
  - Y también, generar y diseñar la mejor solución técnica que permita unir los tramos circulares con los tramos rectilíneos.
  - A continuación, se pretende generar y diseñar la mejor solución técnica que permita unir un tramo con otro de menor radio de curvatura.
  - Y también, generar y diseñar la mejor solución técnica que permita unir un tramo con otro de mayor radio de curvatura.
  - A continuación se pretende simular la trayectoria que más tarde seguirá el robot, y analizar cómo se comporta respecto a la referencia diseñada.
  - También se quiere conseguir que el circuito que siga el robot sea fácilmente editable, de modo que con la misma función siga generándose la curva que une dos tramos a pesar de que estos hayan cambiado.
  - Por último, se buscará desarrollar trayectorias utilizando las soluciones generadas en la búsqueda de los objetivos operativos anteriores y cuya generación pueda ser aplicable en la vida real.

- Respecto a la evitación de obstáculos:
  - Se pretende encontrar una solución que permita confinar el recinto y convertirlo en un recinto cerrado. Recinto que establezca los límites de la zona de trabajo del robot.
  - También se pretende encontrar una solución que permita integrar en la anterior la posibilidad de que existan posibles obstáculos fijos que puedan encontrarse dentro del recinto y que interfieran en la labor del robot móvil.

  - En el mismo sentido, se pretende integrar en el diseño del espacio a los posibles obstáculos móviles. Considerando que puedan encontrarse dentro del recinto y que

puedan colisionar con el robot móvil, si no se soluciona correctamente la gestión de la trayectoria (objetivo este que se contempla más adelante).

- A continuación ya se pretende conseguir que el robot móvil sea capaz de avanzar hasta alcanzar el punto del plano fijado como meta.
- Además el anterior objetivo, se complementará con el de buscar que el robot móvil evite los posibles obstáculos fijos, presentes en el entorno.
- Y además, lograr que el robot móvil pueda esquivar los obstáculos móviles, que pudiesen cruzarse en el camino del robot.
- Diseñar el camino óptimo que debe seguir el robot para conseguir sus objetivos de la manera más eficaz posible.
- Simular la trayectoria que seguirá el robot, y analizar su comportamiento respecto a la trayectoria esperada.
- Y por último, permitir que el recinto sea fácilmente modificable, de modo que se pueda realizar el algoritmo para recintos con diferente número de obstáculos y en diferentes posiciones.

2. DESARROLLO TEÓRICO

2.1. Introducción a la Robótica

La robótica pertenece a la rama de la ingeniería, donde se conciben, diseñan, programan y fabrican las máquinas que tienen la capacidad de reproducir las tareas que le ordena el ser humano o aquellas que requieren de cierta inteligencia.

La idea de conseguir máquinas capaces de sustituir a los seres humanos para descargar a estos de trabajo no es nada nueva, incluso podríamos afirmar que viene de hace varios siglos. La búsqueda por obtener máquinas capaces de realizar tareas por ellas mismas no es tan reciente, siendo incluso también un tema recurrente en las novelas de ciencia ficción.

No obstante, lo cierto es que la robótica es un campo que está alcanzando su mayor relevancia en los últimos años. Y sus avances no son tan rápidos como se podría pensar, y los mayores logros se están produciendo gracias a reconocer que, reproducir las tareas que a simple vista parecen sencillas puede necesitar procesos largos, especialmente para la robótica autónoma.

El término "robot" fue acuñado por vez primera por **Karel Capek**, dramaturgo y novelista checo, en su obra de teatro "R.U.R" (*Rossum's Universal Robots*) de 1920. En checo "robot" viene de *"robota",* que significa *"trabajo",* entendido como *trabajo forzado o esclavitud.*

Un robot es un dispositivo mecánico capaz de realizar una variedad de trabajos productivos, siguiendo unas instrucciones previamente programadas.

La robótica siempre ha proporcionado al sector industrial un excelente avance entre productividad y flexibilidad, aumentando la calidad de los productos y la seguridad y fiabilidad del trabajo, pues los robots pueden sustituir a los operarios en situaciones peligrosas, y reproducir procesos fatigosos de trabajo con índices de fiabilidad muy altos.

Además, la robótica ha salido del sector industrial, irrumpiendo en la vida cuotidiana de las personas, haciendo su vida más cómoda y sencilla.

Los robots pueden estar equipados con articulaciones, ruedas, patas cadenas, etc. que permiten actuar sobre ellas, de forma que se puede controlar su movimiento. Además de poder poseer sensores que le permiten percibir ciertos aspectos del entorno, que afectan a sus fines funcionales.

2.2. Tipos de Robots

A la hora de tipificar, los diferentes formatos de robots existentes, podemos diferenciar entre los robots industriales y los robots de servicio. Y por otro lado, una segunda tipificación distingue entre los robots manipuladores y los robots móviles.

2.2.1. Robots Industriales

Los robots industriales son aquellos que han sido diseñados con el fin de ser utilizados en un entorno de fabricación industrial. Los robots industriales suelen ser articulaciones y brazos desarrollados para reproducir tareas específicas y repetitivas como soldaduras, manejo de material, pintura y otros.

El ámbito industrial ha sido clave para el desarrollo de la robótica, ya que la sucesión de tareas repetitivas ha permitido dar un mayor uso a este tipo de máquinas.

Siendo estos, quizás los más consolidados técnicamente. Y los grandes avances producidos generan un conocimiento base muy importante para el resto de grupos de robots.

Figura 2.1 Robot Industrial.

https://prnoticias.com/podcast/ondacro/tendencias-tecnologicas/20157879-robots-industriales#inline-auto1804

2.2.2. Robots de Servicio

Los robots de servicio son aquellos que se usan fuera de una instalación industrial, estos pueden ser usados para fines profesionales o para un uso más personal.

Entre estos se encuentran: **Robots domésticos**, tales como aspiradoras robóticas, limpiadores de piscinas, robots de vigilancia y telepresencia, y otros; **Robots médicos**, son utilizados en medicina y que pueden llegar a ser capaces de realizar cirugías; **Robots militares**; utilizados en aplicaciones militares, incluye robots de desactivación de bombas o aviones de reconocimiento; **Robots espaciales**, son robots utilizados en misiones espaciales como la Estación Espacial Internacional o los vehículos de Marte entre otros; **Robots Educacionales**, usados para enseñar robótica como Lego; además de otros.

Figura 2.2 Robot de Servicio

Wikipedia (2019) https://es.wikipedia.org/wiki/Robot_de_servicio

2.2.3. Robots Manipuladores

Los robots manipuladores son mayoritariamente robots industriales, pueden adoptar forma antropomórfica (robots de brazo) o cartesianas (robots de ejes). Estos robots se encuentran anclados en al menos uno de sus extremos.

A principios de la década de los sesenta se introducen en la industria, suscitando el interés de los ingenieros para hacer de estos, robots más rápidos, precisos y fáciles de programar. Permitiendo así avanzar en la automatización industrial.

Figura 2.3 Robot Manipulador

https://es.123rf.com/photo_61056152_modelo-de-pl%C3%A1stico-de-la-rob%C3%B3tica-industrial-brazo-del-robot-manipulador.html

2.2.4. Robots Móviles

Los robots móviles aparecen debido a la necesidad de transporte de material o de seres humanos, disponen de mecanismos para permitir su desplazamiento. Pueden ser utilizados en ambientes industriales, pero también pueden ser robots militares (para misiones de rescate o exploración), domésticos, espaciales,…

Los primeros vehículos guiados automáticamente (AGV) aparecen en los años setenta, como herramienta para poder dotar a la industria de un transporte eficaz de los materiales entre las distintas zonas de la cadena de producción.

Figura 2.4 Robot Móvil

https://www.robotnik.es/robots-moviles/summit-xl-hl/

2.3. Robótica Móvil

En 1949 W. Gray Walter construyó Machina Speculatrix, los dos primeros robots móviles (Elmer y Elise) equipados por un sensor de luz, que eran atraídos por las fuentes de luz, pudiendo evitar o mover los obstáculos que se encontraban a su paso.

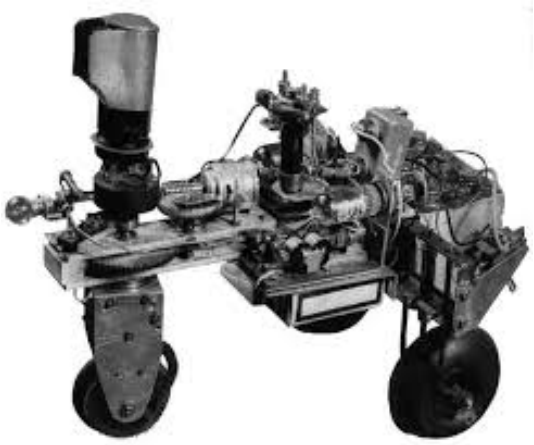

Figura 2.5. Machina Speculatrix

https://elprofetiraondas.wordpress.com/2012/02/22/machina-speculatrix/

En 1961 la universidad Johns Hopkins desarrolló "Bestia", un robot que incorporaba un sonar para facilitar su movimiento, y era capaz de encontrar y enchufarse a una toma de corriente cuando se le agotaban las baterías.

Figura 2.6. Bestia

http://cyberneticzoo.com/cyberneticanimals/1962-3-hopkins-beast-autonomous-robot-mod-ii-pre-sonar-jhu-apl-american/

En 1970 los primeros AGV entran en la industria con el seguidor de línea Stanford Cart.

Figura 2.7 Stanford Cart.

https://www.deviantart.com/leftcoastcreative/art/1970-The-Stanford-Cart-141247705

En sus inicios la tarea que se le ordenaba al AGV, se hacía estableciendo en una secuencia de acciones, y se suponía que el robot había alcanzado su objetivo cuando llegaba a su fin.

En 1976 la NASA envía dos naves no tripuladas a Marte.

En 1980 surgen los primeros robots domésticos, debido al gran interés que la robótica despertó en el público.

Figura 2.8 Robot Doméstico RB5X

http://cyberneticzoo.com/robots/1982-rb5x-the-intelligent-robot-joseph-bosworth-american/

En la década de los noventa J. Engelberger diseña los primeros robots de hospitales vendidos por HelpMate.

En 1991 se desarrolla el primer robot móvil autónomo destinado a labores de investigación.

En 1993 se diseñan los Robots Dante I y II para la exploración de volcanes activos.

Figura 2.9. Dante II

http://cyberneticzoo.com/walking-machines/1992-4-dante-dante-ii-john-e-bares-william-red-whittaker-american/

En 1994 los vehículos VaMP y VITA-2 conducen más de mil kilómetros en una autopista de 3 carriles de Paris, alcanzando velocidades de 130 Km/h, demostrando la conducción en carriles libres, la conducción en convoy y los cambios de carril de izquierda a derecha.

Figura 2.10. Vehículo VaMP

https://medium.com/@davidrostcheck/the-self-driving-car-from-1994-fb1ec617bd5a

En 2002 iRobot saca al mercado su robot doméstico Roomba.

Figura 2.11 Roomba

https://www.irobot.es/robots-domesticos/aspiracion

En 2010 surgen vehículos autónomos que mapean un gran entorno urbano, identifican y rastrean a los humanos y evitan objetos hostiles.

En 2017 desarrollan robots capaces de trabajar en condiciones extremas en instalaciones de gas y petróleo en alta mar.

Los robots móviles son menos precisos que los manipuladores, esto se debe a las prioridades de investigación, mientras que en los robot manipuladores se prima la precisión, la evolución de los robots móviles está orientada a las áreas de sensado y raciocinio.

El robot puede trabajar en un ambiente **interior**, cuando el espacio está delimitado por paredes y techos, principalmente, en estos ambientes, la mayor parte de la luz que reciben es artificial; o **exterior**, el área de trabajo no está definida y la iluminación es esencialmente natural.

El robot trabajara en un **ambiente controlado** si los obstáculos presentes son fijos y pueden ser reconocidos como figuras geométricas tales como prismas o cilindros; el **ambiente no será controlado** cuando los obstáculos pueden no ser distinguibles para el robot o existen otros robots móviles en el entorno.

Los componentes de un robot móvil son un controlador, un software de control, sensores y actuadores.

El controlador normalmente es un microprocesador integrado, pero también puede ser un PC.

El software de control móvil puede ser lenguaje de ensamblaje o de alto nivel como C, C++, Pascal, Fortran o software especial en tiempo real.

Los sensores dependerán de los requisitos del robot y del entorno, estos pueden hacer de contador, detectar posición, detectar colisión,…

Es posible clasificar los robots móviles según su modo de navegación:

El robot por **control remoto manual o teleoperado** está controlado por un conductor a través de un dispositivo de control o joystick. El dispositivo de control puede estar unido al robot o ser inalámbrico. Este tipo de robots se suelen utilizar para tareas peligrosas, manteniendo al operador fuera de peligro. Ejemplos de estos robots son: ANATROLLER ARI-100 de Robotics Design, Talon de Foster-Miller, PackBot de iRobot o Koster Tek's MK-705 de Roosterbot.

Figura 2.12 Robot Teleoperado ANATROLLER ARI-100.

https://www.youtube.com/watch?v=RrAHsA2bLIM

El **robot teledirigido** tiene capacidad para detectar obstáculos y evitarlos, pero navega como un robot teleoperado manualmente

Los **AGV** tienen un sistema de navegación basado en el seguimiento de una línea visual pintada o incrustada en suelo o techo; o un cable eléctrico. Estos robots tienen una orden simple de mantener la línea o cable en el sensor central. El principal problema es que ante cualquier imprevisto en el trayecto que afecte a la navegación, el vehículo se ve incapacitado de realizar la tarea, para la que había sido programado, con éxito, los AGV esperan a que se retire aquello que bloquea su camino, para poder seguir con su función. Ejemplos de este tipo de robot son Transbotics, FMC, Egemin o HK Systems.

Figura 2.13 AGV Egemin

http://warehousenews.co.uk/2013/12/less-cost-and-more-efficiency-with-egemin-agvs/

Los **robots guiados de forma autónoma** conocen en cierta medida donde están y cómo lograr ciertos objetivos o puntos de paso. La localización puede conseguirla por diversos métodos, utilizando sensores. Entre estos robots se encuentran el PatrolBot, capaz de interactuar con los sistemas de detección y control del edificio, para responder alarmas, manipular los ascensores y notificar cuando ocurre un accidente; el SpeciMinder o los TUG, que hacen entregas a hospitales.

Figura 2.14 Robots Guiados de Forma Autónoma TUG

https://www.youtube.com/watch?v=MLZMAW9lqXE

Los robots pueden combinar múltiples modos de navegación, esto se conoce como **autonomía deslizante** ofreciendo un modo manual, a pesar de poder ser guiados autónomamente. Por ejemplo el robot hospitalario HelpMate, o los robots con sistema operativo Motivity, tales como ADAM, PatrolBot, SpeciMinder o MapperBot.

Figura 2.15. Autonomía Deslizante PatrolBot

https://yenra.com/security-robot/

2.4. Sistemas de Control

La robótica móvil es un área que combina los conocimientos sobre cinemática, dinámica, control, procesado de señales, ciencia computacional y otros.

Entre los robots móviles se conocen diferentes configuraciones:

Configuración Diferencial: Disponen de 2 ruedas trabajando a tracción, además de poder poseer una o dos ruedas locas de apoyo, para facilitar el giro. Su diseño es fácil y simple, es barato, pero es difícil de controlar.

$$v = \frac{v_d + v_i}{2} \tag{4.1}$$

$$\omega = \frac{v_d - v_i}{2b} \tag{4.2}$$

De esta forma se obtiene las velocidades lineal y angular del robot, a partir de la velicidad lineal de las ruedas izquierda y derecha (v_i, v_d) y la distancia entre ruedas 2b.

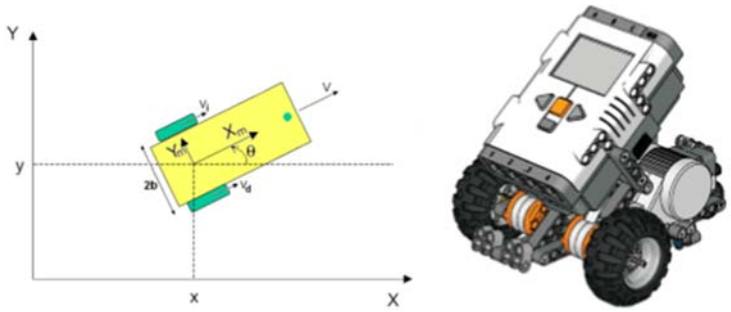

Figura 2.16 Configuración Diferencial

Configuración de Oruga: Poseen cadenas como mecanismo de transmisión de movimiento, se comporta como la configuración diferencial, con dos ruedas equivalentes en el centro de las cadenas, son más fáciles de controlar ya que disponen de un mejor sistema de tracción y sistema anti-derrape.

Las velocidades lineal y angular se consiguen considerando dos ruedas equivalentes en el centro de las cadenas, figura, y con las ecuaciones de la configuración diferencial.

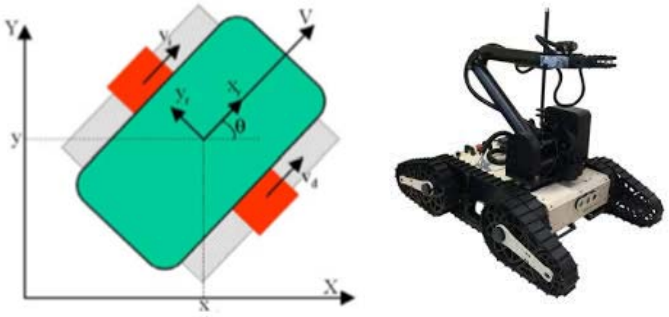

Figura 2.17 Configuración de Oruga

https://www.robotshop.com/es/es/plataforma-movil-con-orugas-dr-robot-jaguar-v6-con-brazo.html

Configuración Triciclo: Compuestos por dos ruedas de tracción y una rueda orientable, generalmente delantera, pudiendo actuar únicamente sobre la orientación de la rueda y la velocidad de avance del robot. Presenta menor deslizamiento que las anteriores configuraciones.

$$v = \cos(\alpha)\, V_t \tag{4.3}$$

$$\omega = \frac{-\sin(\alpha)}{l} V_t \tag{4.4}$$

Las velocidades lineal y angular del robot, en este caso dependen de la velocidad de avance y la orientacion de la rueda orientable.

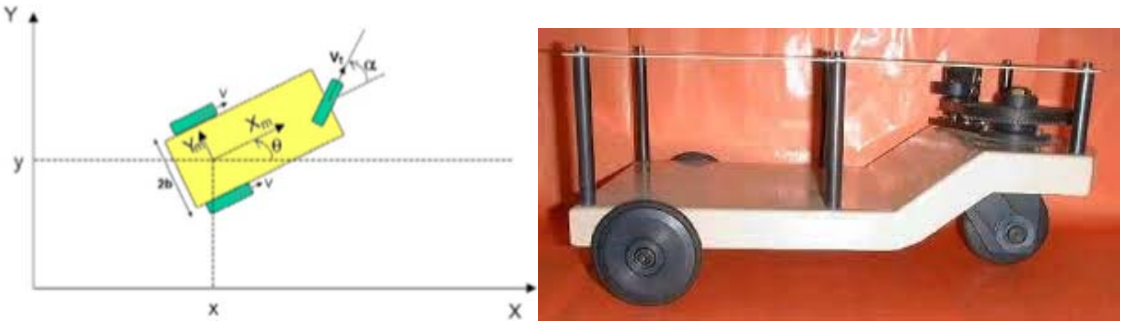

Figura 2.18 Configuración Triciclo.

https://www.feriadelasciencias.unam.mx/anteriores/feria23/feria276_01_robo_fastcar.pdf

Configuración Ackerman: Famosa por ser la configuración típica de los automóviles de 4 ruedas. Las ecuaciones cinemáticas para esta configuración coinciden con la configuración triciclo siendo la rueda orientable la equivalente a las dos ruedas orientables de la configuración Ackerman, coincidiendo la velocidad y la orientación de la rueda equivalente.

Las velocidades lineal y angular de robot se consiguen considerando una rueda equivalente a las dos ruedas delanteras, y utilizando las ecuaciones de la configuración triciclo.

Figura 2.19 Configuración Ackerman

https://www.robotnik.es/robots-moviles/rb-car/

Como se ha dicho, la robótica engloba otros campos, entre ellos, la cinemática y la dinámica.

La cinemática se dedica al estudio de la posición y orientación de los distintos mecanismos en el espacio, mientras que la dinámica se basa en el estudio de fuerzas necesarias para que se produzca movimiento.

La robótica usa estas ramas de la física para controlar en qué posición del espacio se encuentra el robot para poder generar los movimientos que le han sido programados.

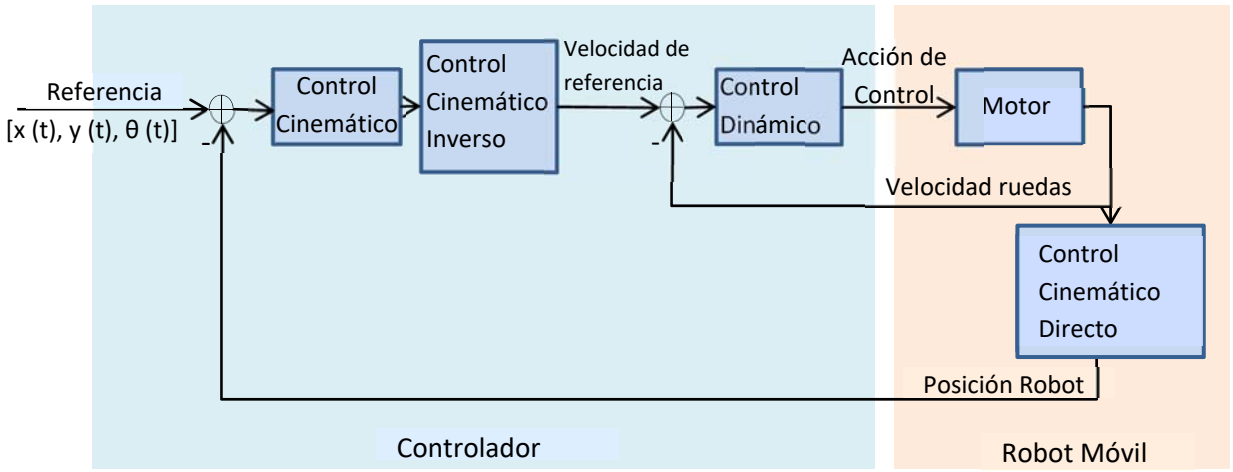

Figura 2.20. Bucle de Control.

El control cinemático de robots móviles radica en obtener las acciones necesarias para desplazar al robot desde su posición inicial a una posición final deseada, a partir de las velocidades y la orientación.

El **control cinemático directo** permite obtener la posición y orientación del robot en función de las velocidades y posiciones de las ruedas de los robots móviles. En los robots manipuladores permite calcular la posición de un punto del mecanismo en función de la configuración de este.

Para la configuración diferencial, este control se consigue mediante las siguientes ecuaciones:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2b} & \frac{1}{2b} \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix} \tag{4.5}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\theta)\frac{v_i + v_d}{2} \\ \sin(\theta)\frac{v_i + v_d}{2} \\ \frac{v_d - v_i}{2} \end{bmatrix} \tag{4.6}$$

El **control cinemático inverso** utiliza la posición y orientación del robot para obtener la velocidad y posición de las ruedas de los robots móviles. Para los robots manipuladores, se calcula la configuración necesaria para alcanzar un punto del espacio.

Se pueden obtener las ecuaciones del control cinemático inverso, a partir de las del control cinemático directo:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix} v_i \\ v_d \end{bmatrix} = \begin{bmatrix} 1 & -b \\ 1 & b \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{x}\cos\theta & \dot{y}\sin\theta & -b\dot{\theta} \\ \dot{x}\cos\theta & \dot{y}\sin\theta & b\dot{\theta} \end{bmatrix} \tag{4.8}$$

Integrando la ecuación de las velocidades, se consigue la ecuación que calcula la posición del robot en función del tiempo, que para valores de Δt y Δθ muy pequeños se puede asumir la siguiente aproximación:

$$\begin{bmatrix} x(t+\Delta t) \\ y(t+\Delta t) \\ \theta(t+\Delta t) \end{bmatrix} = \begin{bmatrix} x(t) + \frac{2v(t)}{\omega(t)} \sin\left(\omega(t)\frac{\Delta t}{2}\right) \cos\left(\theta(t) + \omega(t)\frac{\Delta t}{2}\right) \\ y(t) + \frac{2v(t)}{\omega(t)} \sin\left(\omega(t)\frac{\Delta t}{2}\right) \sin\left(\theta(t) + \omega(t)\frac{\Delta t}{2}\right) \\ \theta(t) + \omega(t)\Delta t \end{bmatrix} \approx \begin{bmatrix} x(t) + v(t)\,\Delta t\cos\big(\theta(t)\big) \\ y(t) + v(t)\Delta t\sin\big(\theta(t)\big) \\ \theta(t) + \omega(t)\Delta t \end{bmatrix} \tag{4.9}$$

Pudiendo discretizarla, de la siguiente forma:

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + v(k)T_s\cos\big(\theta(k)\big) \\ y(k) + v(k)T_s\sin\big(\theta(k)\big) \\ \theta(k) + \omega(k)T_s \end{bmatrix} \tag{4.10}$$

Siendo v (k) y ω (k) las velocidades lineales y angulares del robot en el instante k:

$$v(k) = \frac{v_i(k) + v_d(k)}{2} \tag{4.11}$$

$$\omega(k) = \frac{v_d(k) - v_i(k)}{2b} \tag{4.12}$$

El **control de trayectoria** establece un control sobre la curva temporal de cada una de las coordenadas del robot. El principal problema de este control es que las coordenadas están asociadas a una unidad de tiempo, por lo que si se da algún fallo o se encuentra un obstáculo, el robot dejara de seguir la trayectoria establecida e intentará alcanzar lo antes posible la posición correspondiente al tiempo transcurrido.

El control se obtiene a partir de la posición y velocidad del punto que está a una distancia g respecto al eje de tracción del robot, siendo la posición y la velocidad las siguientes:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x + g * \cos(\theta) \\ y + g * \sin(\theta) \end{bmatrix} \tag{4.13}$$

$$\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} = \begin{bmatrix} k_{rv}\dot{x_{ref}} \\ k_{rv}\dot{y_{ref}} \end{bmatrix} + \begin{bmatrix} k_{rp} & 0 \\ 0 & k_{rp} \end{bmatrix} \begin{bmatrix} x_{ref} - (x + g * \cos(\theta)) \\ y_{ref} - (y + g * \sin(\theta)) \end{bmatrix} \tag{4.14}$$

Con lo que la ecuación cinemática inversa del robot diferencial resulta:

$$\begin{bmatrix} v_i \\ v_d \end{bmatrix} = \frac{1}{g} \begin{bmatrix} b\sin(\theta) + g\cos(\theta) & g\sin(\theta) - b\cos(\theta) \\ g\cos(\theta) - b\sin(\theta) & b\cos(\theta) + g\sin(\theta) \end{bmatrix} \begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} \tag{4.15}$$

El **control de camino** constituye el control sobre la curva definida en el espacio Cartesiano sin tener en cuenta el factor tiempo.

El **control dinámico** establece el control de la velocidad y orientación de las ruedas considerando las referencias obtenidas por el control cinemático. Este control se realizará tanto para la posición como para la velocidad.

Las acciones control son las que se encargan de efectuar los cambios necesarios en el sistema para poder obtener una señal de salida lo más aproximada a la señal de referencia.

Existen cuatro acciones de control que se obtienen a partir de la combinación de las acciones básicas (proporcional, derivativa e integral):

La acción de **control proporcional** (P), es proporcional al error entre la señal de salida y la de referencia:

$$u(t) = k_p\, e(t) \tag{4.16}$$

k_p es la ganancia proporcional, siendo conocer este valor la única necesidad para diseñar esta clase de controlador.

Fácilmente se puede implementar el algoritmo de control, pues:

$$u(k) = k_p\, e(k) \tag{4.17}$$

Donde cada acción de control se calcula para un instante k.

El **control proporcional-diferencial** (PD) combina la acción proporcional y la derivativa.

La acción de control derivativa es proporcional a la derivada de la señal del error, permitiendo anticiparse al error, consiguiendo así una respuesta más rápida y estable.

$$u(t) = k_{pd}\left[e(t) + T_d\frac{de(t)}{dt}\right] \tag{4.18}$$

Si se quiere obtener el controlador discreto equivalente, la acción derivada se obtiene de la siguiente forma:

$$\frac{de(t)}{dt} \cong \frac{e(k) - e(k-1)}{T} \tag{4.19}$$

Siendo T el tiempo de muestreo, la acción de control de este control será:

$$u(k) = k_{pd}\left[e(k) + \frac{T_d}{T}e(k) - \frac{T_d}{T}e(k-1)\right] \tag{4.20}$$

La acción de control en este caso depende del error en el mismo instante y en el anterior.

El **control proporcional-integral** (PI) combina acción proporcional y acción integral.

Para la acción integral, la acción de control es proporcional a la integral de error, permitiendo eliminar el error en el régimen estable.

$$u(t) = k_{pi}\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt\right] \tag{4.21}$$

A la hora de obtener el discreto equivalente existen distintas opciones para aproximar la integral del error. Una de ellas es:

$$\int_0^t e(t)dt \cong T \sum_{i=0}^{k-1} e(i) \tag{4.22}$$

Quedando la siguiente expresión del controlador proporcional-integral:

$$u(k) = k_{pi} \left[e(k) + \frac{1}{T_i} T \sum_{i=0}^{k-1} e(i) \right] \tag{4.23}$$

$$u(k) = k_{pi} \left[e(k) + \frac{1}{T_i} T\, e(k-1) - e(k-1) + u(k-1) \right] \tag{4.24}$$

Como se puede ver la acción de control del PI depende tanto del error en el mismo instante y en el anterior como en la acción de control del instante anterior.

Y el **control proporcional-integral-diferencial** (PID) combina las acciones proporcional, derivativa e integral.

$$u(t) = k_{pid} \left[e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right] \tag{4.25}$$

Si se quiere obtener la discreta equivalente, se usarán las aproximaciones a la integral y la derivada vistas anteriormente en (4.19) y (4.22). Llegando a la siguiente expresión:

$$u(k) = k_{pi} \left[e(k) + \frac{T_d}{T} e(k) - \frac{T_d}{T} e(k-1) + \frac{1}{T_i} T \sum_{i=0}^{k-1} e(i) \right] \tag{4.26}$$

$$u(k) = k_{pid} \left[e(t) + T_d \frac{e(k)}{T} - e(k-1) + \frac{T}{T_i} e(k-1) - 2T_d \frac{e(k-1)}{T} + T_d \frac{e(k-2)}{T} + u(k-1) \right] \tag{4.27}$$

En este caso la acción de control depende de los valores del error en ese mismo instante y en los dos anteriores, además del valor de la acción en el instante anterior.

Con esto, si se realiza el control proporcional del robot con configuración diferencial, se alcanzaría la siguiente ecuación:

$$\begin{bmatrix} acción\ de\ control_i \\ acción\ de\ control_d \end{bmatrix} = \begin{bmatrix} k_{mp} \dfrac{\left(v_{i_{ref}} - v_{i_{robot}}\right)}{radio} \\ k_{mp} \dfrac{\left(v_{d_{ref}} - v_{d_{robot}}\right)}{radio} \end{bmatrix} \quad (4.28)$$

2.5. Curvas de Transición

Como se ha dicho antes, los dos grandes retos a los que se enfrenta la robótica móvil son el de generación de trayectorias, y el de guiar su movimiento según estas, evitando colisiones.

Para la generación de trayectorias es muy importante tener en cuenta la transición entre los distintos tramos de los que dispondrá el circuito que se desea que recorra el robot móvil.

En carreteras y vías de ferrocarril, los primeros trazados fueron realizados mediante rectas y circunferencias, lo cual, cuando los vehículos aumentaron su velocidad, producía una fuerte sacudida al entrar en la curva debido al aumento de la fuerza centrífuga. Para solventar este problema, los ingenieros usaron sus conocimientos en física y matemáticas, principalmente conceptos de geometría sobre el radio de curvatura y la fuerza centrífuga.

De este modo surgió la idea de las curvas de transición, permitiendo que el radio de curvatura se modificara suavemente, permitiendo un cambio en la fuerza centrífuga más suave eliminando así la fuerte sacudida al entrar en las curvas.

Estas transiciones serán lo más suaves posibles, sin ninguna sacudida, de manera que el trayecto sea lo más cómodo para aquello que esté transportando el robot, e incluso, para poder evitar posibles fallos en el robot debido al cambio repentino de dirección.

El principal objetivo de las curvas de transición es acomodar los radios de curvatura entre los diferentes tramos dando así continuidad a la curvatura del trazado.

Las curvas de transición fueron analizadas por primera vez por Max Von Loeber en 1860, siendo aplicadas a vías de ferrocarriles. En este campo, las curvas de transición son realmente importantes ya que sin ellas el efecto de la fuerza centrífuga es muy notable, desgastando así tanto las ruedas del ferrocarril como los raíles, hasta que se produce el descarrilamiento.

Otro campo donde estas curvas son muy importantes es en el diseño de carreteras, donde los conductores cuando se encuentran con una curva, procuran recorrerlas mediante una curva de radio variable.

En la actualidad se pueden encontrar diferentes curvas de transición, de acuerdo o de paso que permiten unir tramos rectos y circulares.

2.5.1. Curvas Compuestas

Las curvas compuestas se componen por una sucesión de arcos circulares, con mismo sentido de curvatura. De forma que si el arco central posee un radio R, los radios de los arcos contiguos no deben ser superiores a 1,5 veces el radio.

En diseño de carreteras estas curvas se pueden emplear en terrenos en los que se desea que la carretera quede lo más ajustada posible a la topografía natural, reduciendo así el movimiento de tierra. Sin embargo, estas curvas no son muy utilizadas.

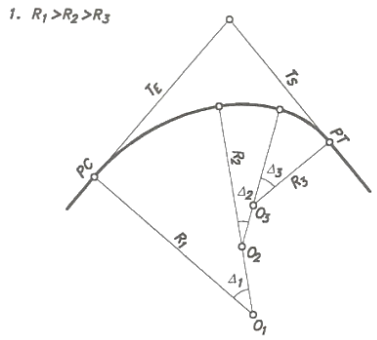

Figura 2.21. Curvas Compuestas

https://viasi.weebly.com/uploads/4/3/2/7/4327492/clase4._alineamiento_horizontal3.pdf

2.5.2. Parábola Cúbica

La parábola cúbica se caracteriza por el hecho de que las ordenadas aumentan proporcionalmente al cubo de la abscisa medida desde el origen. La ecuación de esta curva es:

$$y = \frac{x^3}{6RL} \tag{5.1}$$

Donde L es la longitud total de la curva y R, el radio del tramo circular.

2.5.3. Espiral Cúbica

La espiral cúbica es una parábola cúbica modificada que considera que la longitud relativa "l" es aproximadamente la abscisa de la curva. Esta aproximación empieza a generar errores apreciables para ángulos de deflexión superiores a 24º.

Estas curvas dan un cierto factor de seguridad, sin embargo, no son recomendables desde el punto de vista mecánico. Convergen con menor rapidez que las parábolas cúbicas y son menos precisas. Pero los ingenieros las prefieren porque se expresan en coordenadas cartesianas y son más fáciles de definir.

$$y = \frac{l^3}{6RL} \tag{5.2}$$

2.5.4. Curva de Transición de Schramm

La curva de transición de Schramm o espiral bicuadrática está compuesta por dos arcos de parábola de segundo grado, cuya ecuación aproximada es:

$$y = \frac{x^4}{6RL^2} \tag{5.3}$$

2.5.5. Curva Elástica

La curva elástica es una curva cuya curvatura aumenta proporcionalmente a su abscisa, toma como eje x, la tangente de la curva en el punto de curvatura 0. La ecuación de la curva es:

$$\rho = K_1 x \tag{5.4}$$

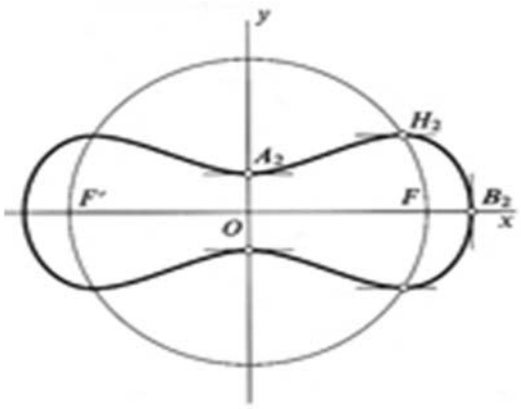

Figura 2.22 Curva Elástica

https://espiralcromatica.wordpress.com/tag/ovalos-de-cassini/

2.5.6. Lemniscata de Bernoulli

Jakob Bernoulli publicó en 1694 un artículo llamado "*Acta Eruditorum*" presentando esta curva con el nombre *lemniscus*, cinta colgante en latín.

Se trata de un caso especial de la curva elástica, cuyas propiedades como curva de transición fueron descubiertas en 1750 por Fagnano.

La lemniscata de Bernoulli o radioide a las cuerdas se conoce como el lugar geométrico de los puntos que cumplen que el producto de las distancias entre los puntos de la curva y dos puntos fijos A y B, es igual a la cuarta parte de la distancia entre A y B.

Puede ser obtenida como la transformada inversa de una hipérbola.

Se trata de una curva con simetría axial, cuyos ejes son los ejes cartesianos.

La ecuación cartesiana de la curva es:

$$(x^2 + y^2)^2 = a^2(x^2 - y^2) \tag{5.5}$$

Siendo 2a la distancia entre A y B.

En coordenadas polares puede ser definida la ecuación como:

$$p = a\sqrt{\sin \alpha} \tag{5.6}$$

Siendo p, la distancia polar; α, el ángulo polar y a el parámetro característico de la curva.

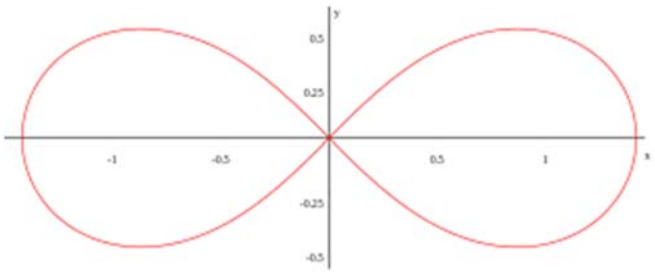

Figura 2.23. Lemniscata de Bernoulli

https://ca.wikipedia.org/wiki/Lemniscata_de_Bernoulli

2.5.7. Clotoide.

La clotoide o radioide de arcos, recibe su nombre del vocablo griego "*Klotho"* que significa *"hilandera"*. Se trata de una curva de la familia de las espirales (curvas planas), que se utiliza en la transición entre tramos curvos y rectos, caracterizada porque el radio de curvatura disminuye de manera inversamente proporcional a la distancia recorrida sobre ella.

Las clotoides, por tanto, proporcionan una transición suave entre tramos. Esta es la causa por la cual el diseño de nuevos trazados de carreteras se hacen por sucesiones de clotoides, pues facilitan la adaptación al terreno.

Sea una curva plana contenida en el plano OXY*:*

$$r(s) = \big(x(s), y(s)\big) \tag{5.7}$$

Siendo t(s) el vector unitario tangente a la curva en el punto r(s), que forma un ángulo φ(s) respecto al eje OX.

Entonces,

$$t(s) = r'(s) = \big(x'(s), y'(s)\big) = \big(cos\ \varphi(s), sin\ \varphi(s)\big) \tag{5.8}$$

Integrando se obtiene:

$$r(s) = \left(\int_{s_0}^{s} \cos\varphi(\mathrm{s})\mathrm{ds}\ , \int_{s_0}^{s} \sin\varphi(\mathrm{s})\ ds\right) \tag{5.9}$$

Siendo r(s) la representación geométrica de una curva plana tal que en cada punto el producto de r (radio de curvatura) por l (longitud de arco recorrido) sea siempre constante e igual a A^2.

$$\boldsymbol{r_1 l_1 = r_2 l_2 = r_3 l_3 = RL = A^2} \tag{5.10}$$

La ecuación de la clotoide viene dada por:

$$\mathrm{r(s)} = \left(\int_0^s \cos\left(\frac{s^2}{2A^2}\right) ds\ , \int_0^s \sin\left(\frac{s^2}{2A^2}\right) ds\right) \tag{5.11}$$

Partiendo de la propiedad que caracteriza las clotoides para un punto P cualquiera: ($r_p l_p$=RL=A^2,), de donde se obtiene que:

$$r_p = \frac{A^2}{l} = \frac{A^2}{s} \tag{5.12}$$

De que la curvatura es la inversa del radio de curvatura:

$$k = \frac{1}{r_p} \tag{5.13}$$

Y de que la curvatura es:

$$k(s) = \frac{\mathrm{d}\varphi}{ds} \tag{5.14}$$

Entonces se tiene que:

$$k = \frac{1}{r_p} = \frac{s}{A^2} = \frac{\mathrm{d}\varphi}{ds} \tag{5.15}$$

De aquí:

$$\int \mathrm{d}\varphi = \int \frac{s}{A^2} ds \tag{5.16}$$

Y por tanto:

$$\varphi(s) = \frac{s^2}{2A^2} \tag{5.17}$$

Con esto y sustituyendo para $s_0 = 0$, (x (s_0), y (s_0))= (0,0):

$$\boldsymbol{r(s) = \left(\int_0^s cos\left(\frac{s^2}{2A^2}\right) ds, \int_0^s sin\left(\frac{s^2}{2A^2}\right) ds \right)} \tag{5.18}$$

Si se hace el cambio de variable $s = \sqrt{\pi} A u$

$$x(s) = \sqrt{\pi}A \int_0^{s/(\sqrt{\pi}A)} cos\left(\frac{\pi u^2}{2}\right) du \tag{5.19}$$

$$y(s) = \sqrt{\pi}A \int_0^{s/(\sqrt{\pi}A)} sin\left(\frac{\pi u^2}{2}\right) du \tag{5.20}$$

Llegando a las expresiones $\int_0^s cos\left(\frac{\pi u^2}{2}\right) du$ y $\int_0^s sin\left(\frac{\pi u^2}{2}\right) du$ conocidas como **Integrales de Fresnel** del coseno y del seno, para estas integrales:

$$\int_0^{+\infty} cos\left(\frac{\pi u^2}{2}\right) du = \int_0^{+\infty} sin\left(\frac{\pi u^2}{2}\right) du = \frac{1}{2} \tag{5.21}$$

$$\int_0^{-\infty} cos\left(\frac{\pi u^2}{2}\right) du = \int_0^{-\infty} sin\left(\frac{\pi u^2}{2}\right) du = -\frac{1}{2} \tag{5.22}$$

Luego

$$r(+\infty) = \left(\frac{\sqrt{\pi}A}{2}, \frac{\sqrt{\pi}A}{2}\right) \tag{5.23}$$

$$r(-\infty) = \left(-\frac{\sqrt{\pi}A}{2}, -\frac{\sqrt{\pi}A}{2}\right) \tag{5.24}$$

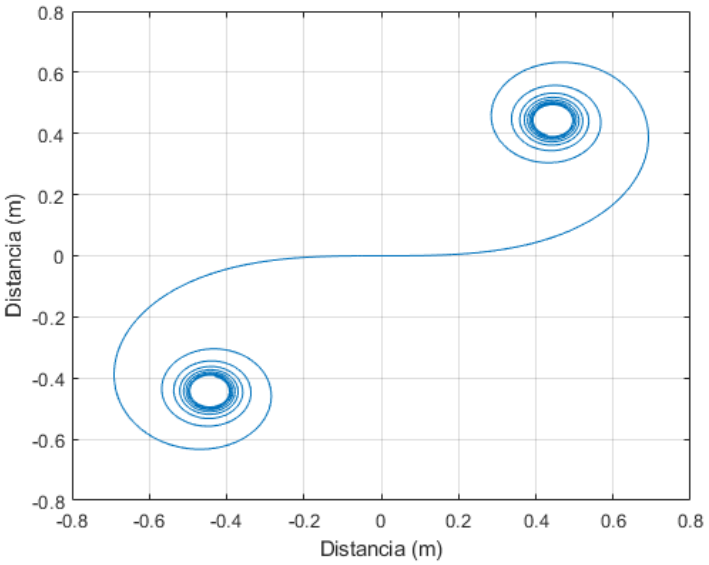

Figura 2.24. Clotoide

Con esto se llega a la conclusión de que en el origen la curva tiene radio infinito (recta) y a medida que se recorre la clotoide, el radio va disminuyendo enroscándose alrededor del punto $\left(\pm\frac{\sqrt{\pi}A}{2}, \pm\frac{\sqrt{\pi}A}{2}\right)$, dependiendo de si se recorre en sentido positivo o negativo.

Todas las clotoides son proporcionales entre ellas, si se quiere diseñar un arco de clotoide habrá que conocer el factor de escala.

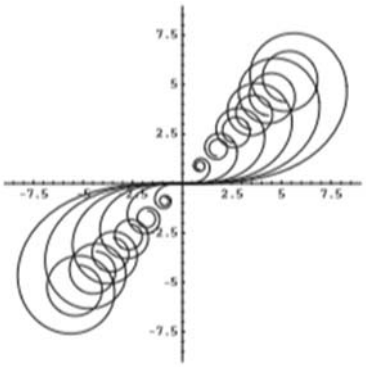

Figura 2.25. Proporcionalidad de Clotoides

https://www.researchgate.net/figure/Figura-4-Clotoide-con-radios-de-curvatura-y-circunferencias-osculatrices_fig4_273313030

Para conocer una clotoide en particular que una un tramo recto con una circunferencia es necesario su parámetro A. Siendo R el radio de la circunferencia, L la longitud total de arco de la clotoide y F= (xf, yf) el punto de enlace con el tramo circular A^2=RL.

Para calcular la clotoide se hará la aproximación mediante el desarrollo de Taylor:

$$x(s) = \int_0^s cos\left(\frac{s^2}{2A^2}\right) ds = \int_0^s cos\left(\frac{s^2}{2RL}\right) ds \approx \int_0^s \left[1 - \frac{1}{2!}\left(\frac{s^2}{2RL}\right)^2 + \frac{1}{4!}\left(\frac{s^2}{2RL}\right)^4 - \frac{1}{6!}\left(\frac{s^2}{2RL}\right)^6\right] ds \quad (5.25)$$

$$\boldsymbol{x(s) = s - \frac{s^5}{10(2RL)^2} - \frac{s^9}{216(2RL)^4} + \frac{s^{13}}{9360(2RL)^6}} \quad (5.26)$$

$$y(s) = \int_0^s sin\left(\frac{s^2}{2A^2}\right) ds = \int_0^s sin\left(\frac{s^2}{2RL}\right) ds \approx \int_0^s \left[\frac{s^2}{2RL} - \frac{1}{3!}\left(\frac{s^2}{2RL}\right)^3 + \frac{1}{5!}\left(\frac{s^2}{2RL}\right)^5 - \frac{1}{7!}\left(\frac{s^2}{2RL}\right)^7\right] ds \quad (5.27)$$

$$\boldsymbol{y(s) = \frac{s^3}{3(2RL)} - \frac{s^7}{42(2RL)^3} + \frac{s^{11}}{1320(2RL)^5} - \frac{s^{15}}{75600(2RL)^7}} \quad (5.28)$$

2.6. Navegación

El otro gran problema que presenta la robótica móvil es el de guiar su movimiento para evitar las colisiones con los posibles obstáculos que pueda encontrarse.

La navegación es la pieza fundamental en la robótica móvil debido a que ella incluye todos los demás aspectos. Dentro de la navegación se encuentra la percepción y la locomoción.

La percepción se entiende como la comunicación que tiene el robot con el entorno, esto se consigue gracias a la información que recibe de los sensores y su correcta interpretación para actuar en consecuencia.

La locomoción decide qué acciones debería adoptar el robot en cada momento, teniendo en cuenta la información que ha percibido de su estado y el estado del entorno, con el fin de moverse el robot y todos los mecanismos que participan para alcanzar la posición final.

Los robots móviles pueden andar, saltar, correr, deslizarse, sumergirse y volar, de forma que hay una gran variedad de opciones para la locomoción.

La navegación autónoma tiene el propósito de generar caminos flexibles entre la posición inicial y la final, lo cual le permita al robot evitar los obstáculos que se vaya encontrando. Normalmente el algoritmo que genera este camino discretiza el entorno y genera un segmento que conecte el punto inicial con el final. Esta conectividad se puede hacer de diversas maneras.

2.6.1. Hoja de Ruta

La hoja de ruta es un método que recae en reglas basadas en la geometría de los obstáculos. Digani y otros (2014) definieron la hoja de ruta como la unión de curvas unidimensionales cuyas propiedades son la accesibilidad, conectividad y salida. Además el robot está restringido a sólo poder moverse sobre las curvas.

Consiste en la creación del camino mediante la unión, mediante líneas rectas, de los vértices de los obstáculos poligonales entre sí y con los puntos iniciales y finales. La conexión solo se conseguirá si la unión no intersecta con el interior de un obstáculo.

El algoritmo crea un esqueleto con los puntos más cercanos a los obstáculos y paredes. De manera que, siguiendo las curvas del diagrama, el robot se mantiene lo más alejado posible de los obstáculos.

Son ampliamente usados los diagramas Voronoi definidos por Bräunl (2008) como:

Sea F el espacio libre del entorno y sea F’ el espacio ocupado. b ϵ F’ es un punto base de p ϵ F si b tiene la mínima distancia a p, comparado con otros puntos de F’. p ϵ F’ | p tiene al menos dos puntos base.

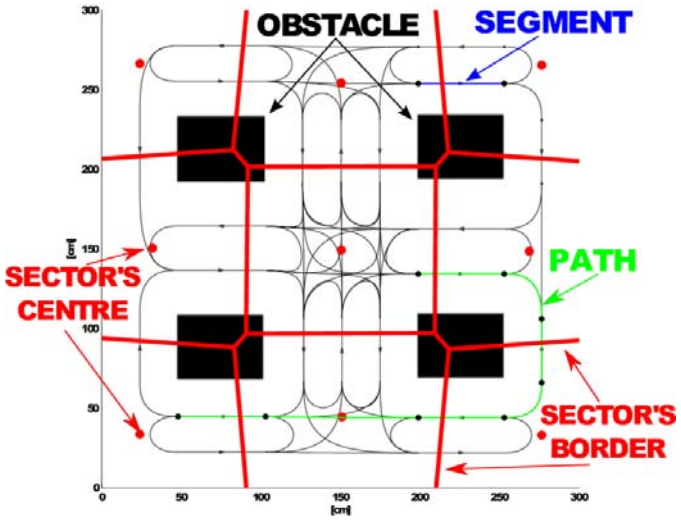

Figura 2.26. Hoja de Ruta

https://www.researchgate.net/figure/Roadmap-of-real-environment_fig24_312523934

Un camino sin colisiones entre dos configuraciones existe si, y solo si, existe un camino en el diagrama correspondiente.

2.6.2. Descomposición en Celdas

La descomposición en celdas se basa en la división del espacio en un número finito de regiones, donde el camino se genera en esas regiones completamente libres. El gráfico obtenido contiene la información de cada celda.

La resolución y la forma de la malla pueden ser elegidas. Dependiendo de ellos, el camino puede cambiar, para llegar al mejor camino la resolución debe ser la apropiada para el entorno y los obstáculos que contiene.

Tanto este método como la hoja de ruta se basan en la geometría, se centran en la conectividad ignorando la optimización y la complejidad computacional.

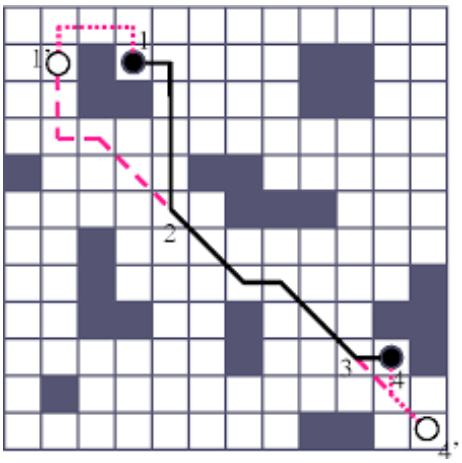

Figura 2.27 Descomposición en Celdas

https://www.semanticscholar.org/paper/Roadmap-methods-vs.-cell-decomposition-in-robot-Seda/d4b62b89acbf9eb50782f04c4b38c47cac515dbe

2.6.3. Algoritmo Insecto

El algoritmo insecto trabaja en ambientes desconocidos. Fue desarrollado por Lumelsky y Stepanov en 1987, existen múltiples versiones. Entre ellas destacan insecto1 en la que el robot se mueve alrededor del perímetro del obstáculo y entonces va al punto final desde el punto más cercano, e insecto 2 donde el robot se mueve alrededor del contorno del obstáculo hasta que es posible ir directamente al punto final.

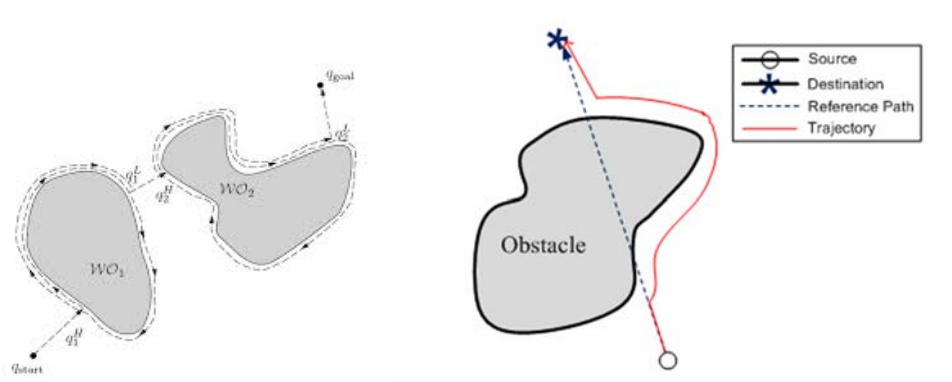

Figura 2.28 Algoritmo Insecto1 (izq.) Algoritmo Insecto2 (der.)

(http://www.aishack.in/tutorials/obstacle-avoidance-bug-algorithm/)
(https://www.researchgate.net/figure/Trajectory-of-dist-bug-algorithm_fig1_324562768)

2.6.4. Campos Potenciales Artificiales

Los campos potenciales es un método que se está ganando popularidad. Este método es una implementación fácil que da buenos resultados. Fue desarrollado por Khatib en 1986, propuesto como un mecanismo de control de bajo nivel para robots móviles.

La base de este método es que los robots son considerados partículas en movimiento en un campo potencial virtual. Los obstáculos se asignan a un campo potencial que repele al robot, mientras que al punto final se le asigna un campo que lo atrae.

La resultante de las fuerzas aplicadas sobre el robot le dará la dirección y la velocidad que permita al robot alcanzar la meta sin colisionar.

Aunque el método fue concebido para robots solitarios en ambientes controlados, se ha usado en casos con múltiples robots o en ambientes desconocidos.

El campo potencial total U_{tot} se define como:

$$U_{tot}(q) = U_{rep}(q) + U_{atr}(q) \tag{6.1}$$

Donde q es la posición del robot (x, y), U_{atr} es el campo de atracción del punto final y U_{rep} es el campo de repulsión de los obstáculos.

El potencial de atracción más usado es:

$$U_{atr}(q) = \frac{1}{2}\varepsilon\rho^{m}(q, q_f) \tag{6.2}$$

Siendo ξ el factor de escala, m es 1 para el campo de atracción con forma cónica o 2 si tiene forma parabólica, ρ(q, qf) es la mínima distancia entre la posición actual del robot y el punto final.

La fuerza de atracción viene dada por:

$$F_{atr} = -\nabla U_{atr}(q) = \varepsilon\rho(q, q_f) \tag{6.3}$$

El potencial de repulsión sólo tiene efecto si el robot se encuentra lo suficientemente cerca del obstáculo, comúnmente se usa:

$$U_{rep}(q) = \frac{1}{2}\eta\left(\frac{1}{\rho(q, q_{obs})} - \frac{1}{\rho_0}\right)^2 \tag{6.4}$$

Donde η es el factor de escala, q_{obs} es la posición del obstáculo y ρ_0 es la máxima distancia de influencia.

La fuerza de repulsión viene dada por:

$$F_{rep}(q) = -\nabla U_{rep}(q) = \eta\left(\frac{1}{\rho(q, q_{obs})} - \frac{1}{\rho_0}\right)\frac{\nabla\rho(q, q_{obs})}{\rho^2(q, q_{obs})} \tag{6.5}$$

El robot se moverá en la dirección de la resultante:

$$\dot{q} = -\nabla\left(U_{atr}(q) + U_{rep}(q)\right) = -\nabla U_{rep}(q) = F_{rep}(q) \tag{6.6}$$

Sin embargo, los principales problemas que plantea este método es que por la configuración de los campos, el robot puede quedarse atrapado entre los obstáculos.

Figura 2.29 Campos Potenciales Artificiales

https://www.fing.edu.uy/inco/grupos/mina/pGrado/easyrobots/doc/Campos_potenciales2_0.pdf

2.7. Lego NXT

Figura 2.30 Lego NXT

http://www.iitg.ac.in/cse/robotics/?page_id=561

El robot móvil con el que se ha trabajado en este proyecto es el ladrillo inteligente NXT de Lego. Este robot pertenece al grupo de **Robots Educacionales**, que a su vez se localiza entre los **Robots de Servicio**.

El ladrillo posee cuatro puertos de entrada con convertidor A/D de diez bits, cuatro botones para poder desplazarse por el menú, temporizadores internos, comunicación Bluetooth y USB 2.0 de 12 Mbit/s a máxima velocidad, tres puertos de salida que permite la lectura de encoders, una pantalla LCD y altavoces.

Dispone de dos procesadores, uno principal Atmel 32-bit ARM con 256 KB de memoria FLASH, 64 KB de RAM y de 48 MHz; y otro Atmel 8-bit AVR de 4 KB de memoria FLASH, 512 Byte de RAM y 8 MHz.

Los actuadores que utiliza son motores eléctricos de eje giratorio que permite una velocidad máxima de unos 200 RPM. Son robustos y precisos, e integran controladores de rotación incrementales. El control de los motores se consigue por modulación de ancho de pulso (PWM), es decir, si el motor está a máxima potencia, la señal que recibe el motor es activo durante todo el periodo. La potencia disminuirá a medida que la señal posea más tiempo inactivo en el periodo.

Los sensores que puede utilizar son sensores de distancia, pudiendo ver objetos desde 3 centímetros hasta a 1 metro de distancia; sensores de luz; sensores de contacto, que generan una señal digital donde 1 corresponde con la situación en la que hay contacto; y sensores de sonido.

2.8. Robot C

Figura 2.31. Robot C.

https://www.vexrobotics.com/robotc-vexedr-vexiq.html

Existen diversos programas capaces de comunicarse con el NXT como el NXT-G code, NXC, Robot C, pbLua, Java y otros.

El programa que se utilizará en este proyecto es el Robot C, un entorno basado en programación en C. Este programa se ejecuta por tareas, pudiendo realizar diversas tareas a la vez dando más prioridad a unas tareas que a otras.

La generación de la señal que llegue al actuador se indica con la función **motor** [nombre_motor]=acontrol, donde en nombre_motor se dirá el nombre que identifica el motor sobre el que se quiere actuar, y acontrol es el porcentaje en tanto por cien de la potencia máxima.

Cuando se trabaja con sensores, el programa leerá la variable que da el sensor gracias a la función variable= **SensorValue** (nombre_sensor), donde nombre_sensor corresponde con el nombre que se ha otorgado al sensor, y variable es el valor que proporciona el sensor.

Si se quiere leer la información del encoder, contenido en el motor, se utilizaría una función similar a la que lee los sensores valor=**nMotorEncoder** [nombre_motor].

Otras funciones interesantes, son las que permiten controlar el tiempo al que funciona el programa como **wait1Msec**(tiempo1) donde tiempo1 es el valor de tiempo en milisegundos que tiene que esperar el programa en la línea de código anterior, **ClearTimer**(tiempo1) vuelve a poner a cero el temporizador de la variable tiempo1.

Otra librería interesante para trabajar en Robot C es *writeFile.c*, esta librería se encarga de la generación de ficheros de texto e incluye funciones como: **createTextFile** (), que crea el fichero de texto; **writeFloatNumber** (), que escribe el valor en coma flotante de la variable deseada; writeNewLine (), hace un salto de línea en el fichero; y **closeWriteTextFile** (), que cierra el fichero de texto.

3. DESARROLLO APLICADO

En este capítulo, se van a aplicar los fundamentos desarrollados en el capítulo 2, para desarrollar soluciones que garanticen la seguridad del robot a medida que este avanza en busca de un punto final deseado. Para ello se han considerado como hitos clave para el cumplimiento del objetivo: que la trazada sea lo más suave posible, y que el robot sea capaz de llegar a su destino sin colisionar con los posibles obstáculos que se pueda encontrar en su camino.

El problema a resolver es el de la generación de trayectorias seguras en robots móviles de configuración diferencial, para trayectos en espacios cerrados, donde pueden existir obstáculos tanto fijos como móviles, a través del uso de las curvas de transición y los algoritmos de evitación de obstáculos.

El trabajo de manera específica, va a desarrollar las soluciones adecuadas a los objetivos operativos descritos en el epígrafe 1.2. Y para ello, aprovechará los fundamentos teóricos de todo el capítulo 2, pero en especial los relativos a curvas de transición y navegación.

El trabajo aportará una solución técnicamente consistente al problema principal de la movilidad de robots móviles en configuración diferencial en circuitos cerrados planos. Algo que adolece de suficientes trabajos empíricos que solucionen los múltiples problemas operativos que se pueden llegar a plantear.

3.1. Soluciones Elegidas

Como ya se ha dejado claro en el capítulo anterior, el trabajo se centra en una parte muy importante de los robots móviles de configuración diferencial, la locomoción, la parte de la navegación que decide qué acciones tomar para la consecución de los objetivos, que en este caso es alcanzar un punto final en el plano.

Y es que uno de los principales retos que tiene que resolver la robótica móvil es generar trayectorias y guiar su movimiento según estas, permitiendo al robot desplazarse por el ambiente de trabajo de manera segura, evitando colisiones.

Para ello se desarrolla la elaboración del presente trabajo siguiendo dos vías claramente diferenciadas. Por un lado se ha buscado generar trayectorias tal que el robot móvil siga una trazada suave, mientras que por otro, se ha buscado el logro de hacer llegar el robot desde un punto inicial a otro final deseado, evitando todos los obstáculos que puedan interponerse en la trayectoria del robot.

Para la primera parte se consideraron las curvas planas, como modo de unión entre tramos rectos y/o curvos, sin considerar peraltes ni demás variables que condicionen la trazada fuera

del plano 2D, pues no se trata de un diseño en 3D. Además, el principal entorno de aplicación va a ser un entorno interior, dentro del laboratorio, donde no van a haber cambios de nivel de ningún tipo. Entre las distintas curvas planas se ha escogido la **clotoide**, pues es el método que más se ha desarrollado últimamente por el amplio abanico de aplicaciones que puede generar, dado a la posibilidad de unir eficazmente tramos de cualquier curvatura y en la posición que sea, por su propiedad de proporcionalidad.

En la segunda, existen diversos algoritmos que permiten generar trayectorias capaces de evitar obstáculos. De entre todas ellas, **crear artificialmente campos potenciales** tales que los obstáculos repelan al robot, evitando así cualquier colisión; y que el punto final atraiga al robot, de manera que el robot se acerque al punto final hasta alcanzarlo; es la idea que más popularidad está obteniendo, además de tener una fácil implementación y de, por lo general dar buenos resultados.

3.2. Diseño del Sistema de Control

El bucle que controla al robot móvil es un bucle cerrado tal que en primer lugar se realiza un control cinemático inverso para obtener la velocidad relativa de las ruedas del robot, para así realizar un control dinámico, que genere la acción de control del robot. A partir de la información que le proporcionen los encoders entonces y el control cinemático directo, se obtendrá la nueva posición del robot que será la que se utilizará para realizar el control cinemático inverso de la siguiente iteración.

Este bucle fue testeado primero en Simulink para poder obtener una simulación de los futuros resultados, permitiendo obtener los resultados más rápidamente y centrarse en el desarrollo del bucle de control exclusivamente. Más tarde, las ecuaciones de este control será discretizado para poder ser implementado en Robot C.

Mediante los bloques de Simulink se implementaron las ecuaciones del control cinemático inverso (4.15), control dinámico y control cinemático directo (4.14), de forma que introduciendo la referencia deseada, calcula la simulación de cómo la seguirá el robot.

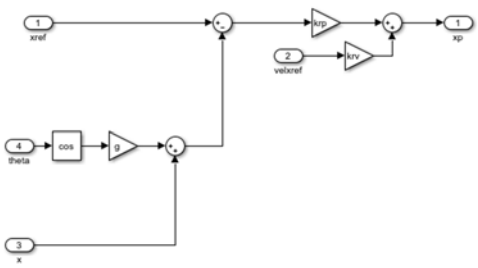

Figura 3.1. Control Cinemático Directo

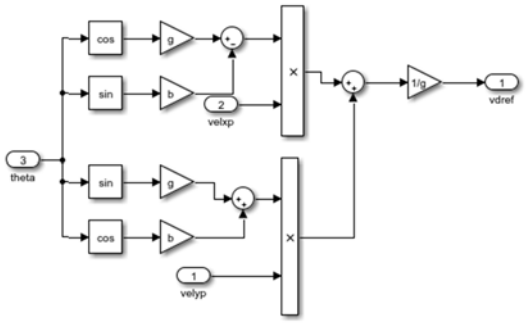

Figura 3.2 Control Cinemático Inverso.

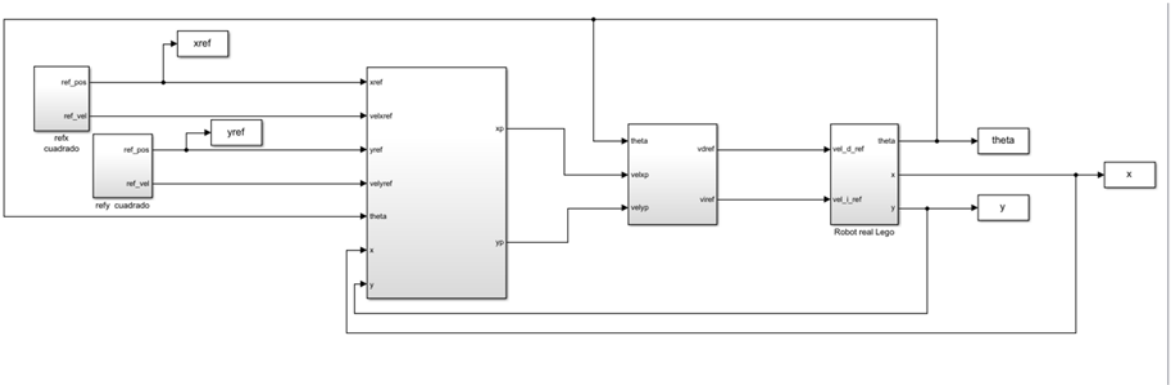

Figura 3.3 Bucle de control Completo

El empleo de la herramienta Simulink resultó bastante útil a la hora de determinar los valores de las constantes del control krp, krv y kmp.

Realizando diferentes simulaciones se pudo ver que el parámetro kmp era el más determinante, siendo este el primero que habrá que fijar, posteriormente se fijaría krp y por último krv.

Se inició probando con los valores kmp=1, krp=1 y krv=0.1.

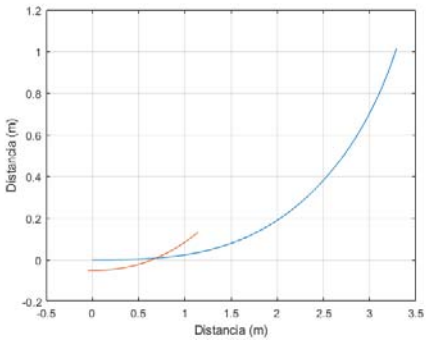

Figura 3.4 Simulación Kmp=1, Krp=1 Y Krv=0.1.

Viéndose que la simulación apenas seguía la referencia, se procedió a incrementar el valor de kmp, fijando el resto de parámetros.

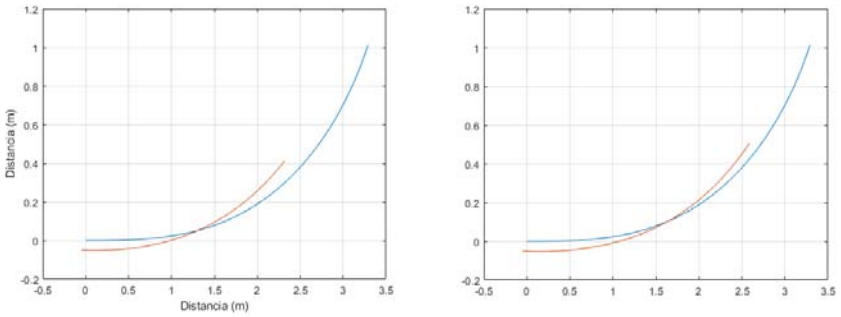

Figura 3.5 Determinación de Kmp.

Llegando a un valor de kmp en el cual apenas se veía diferencia al aumentar el valor de kmp, por ello se decidió a fijar el valor de kmp en 9.

Posteriormente, para intentar conseguir una mejor simulación se procedió a incrementar el valor de krp fijando el resto de parámetros.

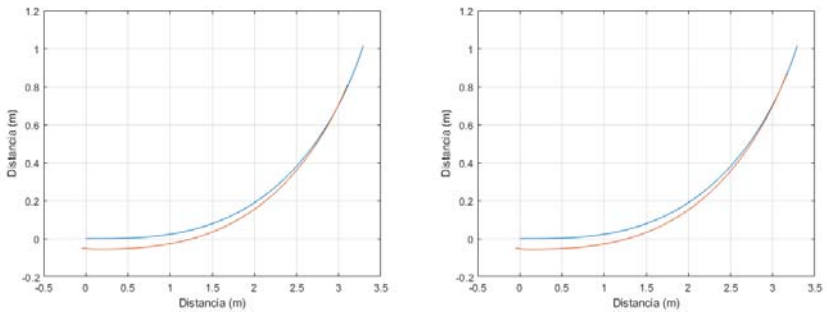

Figura 3.6 Determinación de Krp.

Al igual que ocurría con kmp, al aumentar krp a partir de cierto valor, la diferencia entre las simulaciones era inapreciable, por lo que se decidió fijar el valor de krp en 4.

Finalmente se observó que al aumentar el valor de krv la simulación empeoraba, por lo que se concluyó mantener el valor de krv en 0.1.

De este modo, los valores de los parámetros del control quedaron:

Kmp	Krp	Krv
9	4	0.1

Tabla 1 Parámetros de control.

Con esto, el algoritmo que realiza el control del robot móvil es el siguiente:

Nombre: Algoritmo de control.

Funcionalidad: Realizar el control cinemático y dinámico del robot móvil.

Parámetros de entrada: Señal de referencia (xref, yref).

Parámetros de salida: Simulación de la trayectoria.

Algoritmo:

1. Generar la velocidad de referencia, como derivada de la posición de referencia, obtenida de la trayectoria de referencia.
2. Obtener el valor de los encoders que dan la posición del robot.
3. Calcular la velocidad angular de las ruedas del robot.
4. Calcular la velocidad lineal de las ruedas del robot, como velocidad angular por radio.
5. Calcular la velocidad lineal del robot.
6. Calcular la velocidad angular del robot.
7. Calcular la posición y orientación del robot.
8. Realizar el control cinemático del robot.
9. Calcular el modelo cinemático inverso del robot.
10. Calcular la velocidad angular de referencia de las ruedas.
11. Calcular el error entre la velocidad angular de las ruedas del robot y la velocidad angular de referencia de las ruedas.
12. Calcular la acción de control del control dinámico proporcional.
13. Saturar la acción de control.
14. Aplicar la acción de control a los motores del robot.
15. Actualizar los valores de los encoders para la siguiente iteración
16. Si no se ha acabado de generar la trayectoria, entonces volver al paso 1.

El siguiente paso sería implementar el algoritmo en el código en C, que será el que más adelante se introducirá en el robot para generar la trayectoria. De forma que el algoritmo de control se reproducirá durante toda la trayectoria.

3.3. Diseño y Generación de Clotoides

Para la realización de esta parte del trabajo se ha utilizado el robot NXT de Lego debido a su facilidad de programación, pues al tratarse de un robot de servicio educacional, está diseñado para facilitar el aprendizaje en la robótica móvil. Además de implementarse con el software Robot C, que trabaja en lenguaje C.

Aparte de Robot C, también se ha utilizado Matlab como software de programación debido a su fácil disponibilidad, gracias a la licencia de alumno universitario, permitiendo trabajar en el proyecto sin necesidad de acudir al laboratorio y por permitir una visualización más rápida de los resultados.

La configuración con la que se ha trabajado el robot NXT es Configuración diferencial por su diseño fácil y simple, sin incurrir en gastos innecesarios, trabajando con sólo dos motores. Siendo esta la configuración más efectiva.

Para la consecución de esta parte del trabajo primero se implementó el código que dotaría al programa del control cinemático y dinámico del robot, diseñado anteriormente.

3.3.1. Clotoide Recta-Círculo

Para la generación de las clotoides que unen rectas con círculos es necesario conocer el radio de curvatura final deseado (R), que coincide con el radio del círculo y el ángulo que forman las tangentes deseadas de los puntos inicial y final de la clotoide (V),

Con esto será posible determinar el parámetro L, pudiéndose obtener de la siguiente forma:

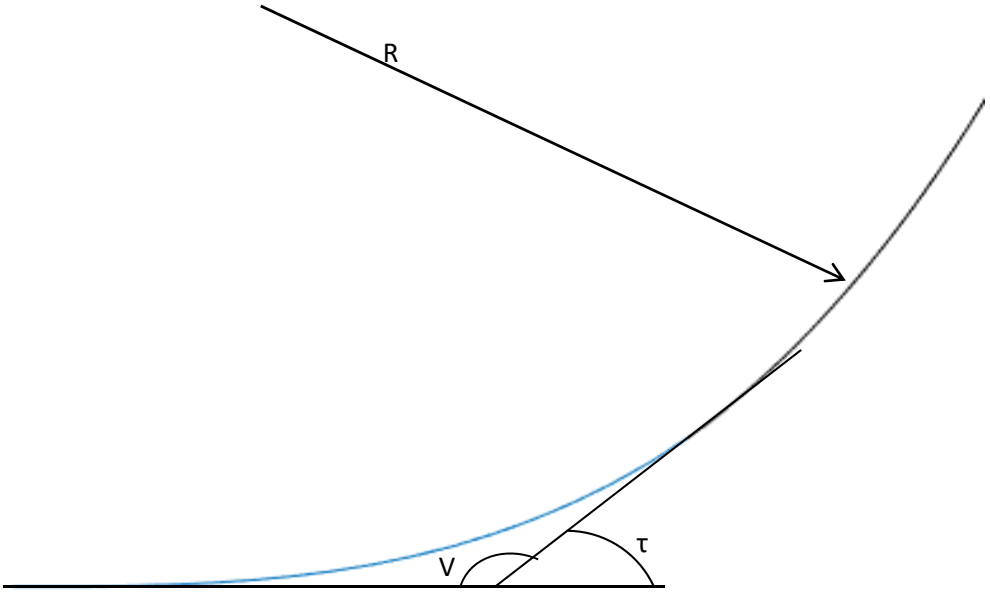

Figura 3.7 Parámetros de la clotoide

Siendo θ el ángulo que forma la tangente de la clotoide en un punto con la horizontal (Figura 3.7) se cumple que:

$$\theta = \frac{l}{2r} \tag{3.3.1}$$

De modo que la tangente en el punto final coincidirá con τ quedando de la siguiente forma:

$$\tau = \frac{L}{2R} \tag{3.3.2}$$

Sabiendo V, ángulo que forman las tangentes de los puntos inicial y final, se conocerá τ como:

$$\tau = \pi - V \tag{3.3.3}$$

De este modo, con τ en radianes, obtenemos la L:

$$L = \tau * 2R \qquad (3.3.4)$$

Una vez conocidos los valores L y R podrán ser aplicadas las aproximaciones a la clotoide que se vieron en las ecuaciones que calculan xref (l) (5.26) e yref (l) (5.28)

Para generar la referencia de la clotoide se usó el siguiente código que implementa las ecuaciones (5.26) y (5.28):

```
if l<= (L) {

    xref = ((l - ((l^5)/ (10*(2*R*L) ^2)) –
   - ((l^9)/ (216*((2*R*L) ^4))) + ((l^13)/ (9360*((2*R*L) ^6))) –
   - ((l^15)/ (75600*((2*R*L) ^7)))))

    yref = (((l^3)/ (3*(2*R*L))) - ((l^7)/ (42*((2*R*L) ^3))) +
   + ((l^11)/ (1320*((2*R*L) ^5))) –
   - ((l^15)/ (75600*((2*R*L) ^7))));

    l=l+0.01;

}
```

Ejemplo Numérico

Se pretende unir un tramo recto con una curva de radio constante R=0,25 m; midiendo en el mapa se observa que el ángulo que forman las tangentes de ambos tramos en los puntos de unión con la clotoide presenta un valor de V=160º=8/9 π rad. La clotoide capaz de unir estos tramos será la que cumpla:

$$\tau = \pi - V = \pi - \frac{8}{9}\pi = \frac{1}{9}\pi\ rad$$

$$L = \tau * 2 * R = \frac{1}{9}\pi * 2 * 100 = 0{,}1745\ m$$

Con estos resultados es posible obtener los valores de las coordenadas X-Y de la trayectoria

$$x(l) = l - \frac{l^5}{10 * (2 * 0{,}25 * 0{,}1745)^2} - \frac{l^9}{216 * (2 * 0{,}25 * 0{,}1745)^4} + \frac{l^{13}}{9360 * (2 * 0{,}25 * 0{,}1745)^6} - \frac{l^{15}}{75600 * (2 * 0{,}25 * 0{,}1745)^7}$$

$$y(l) = \frac{l^3}{3 * (2 * 0{,}25 * 0{,}1745)} - \frac{l^7}{42 * (2 * 0{,}25 * 0{,}1745)^3} + \frac{l^{11}}{1320 * (2 * 0{,}25 * 0{,}1745)^5} - \frac{l^{15}}{75600 * (2 * 0{,}25 * 0{,}1745)^7}$$

Las soluciones obtenidas son las que se grafican a continuación:

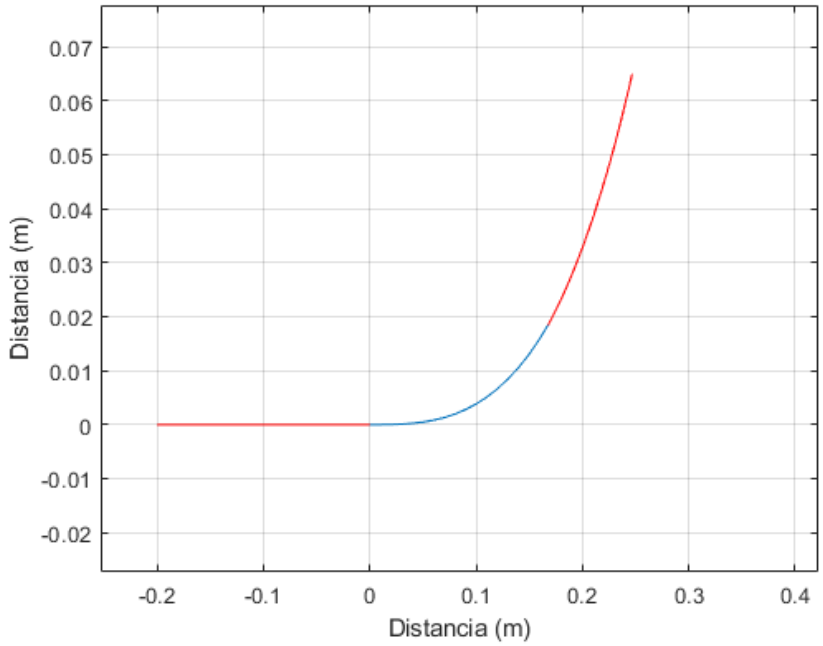

Figura 3.8. Primera Referencia Clotoide

Una vez obtenida la señal de la referencia, se aplica el bucle de control desarrollado en el apartado 3.2. Consiguiéndose así la señal de la acción de control que ordenará a los motores del robot cómo deben actuar.

Finalmente fue realizada la simulación de la trayectoria quedando de la siguiente forma:

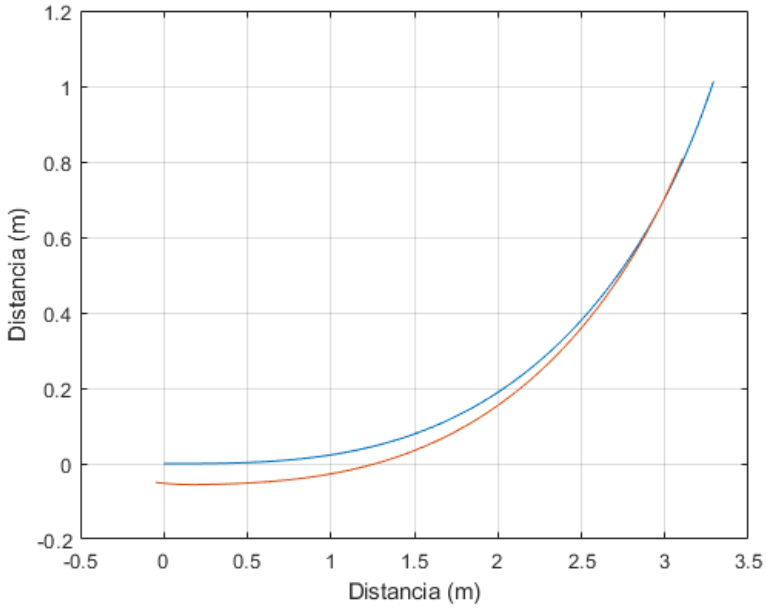

Figura 3.9 Generación Trayectoria Clotoide.

En resumen el algoritmo que genera la clotoide que une un tramo recto con uno circular se desarrollaría de la siguiente manera:

Nombre: Algoritmo Recta-Clotoide -Círculo

Funcionalidad: Diseñar la clotoide que una un tramo recto con uno circular.

Parámetros de entrada: Radio de curvatura del tramo circular (R) y ángulo que forman la tangente del círculo en el punto de unión y la recta (V).

Parámetros de salida: Simulación de la trayectoria.

Algoritmo:

1. Calcular el valor de L aplicando la ecuación (3.3.4), sabiendo que $\tau = 180 - V$
2. Como el primer tramo que queremos unir es la recta, l tendrá el valor 0 (l = 0)
3. Si l es menor que L, entonces se calcularán los valores de xref e yref a partir de las ecuaciones (5.26) y (5.28) como se vio en el código.
4. Realizar el algoritmo de control visto en el apartado 3.2 para obtener la señal de control.
5. Volver al paso 4 si l es menor que L y, si se da el caso, se incrementará el valor de l en 0.01.
6. Realizar la simulación, entregando a los motores las señales de la acción de control.

3.3.2. Clotoide Círculo Recta

Cuando se necesita una clotoide para unir un tramo de menor radio a un tramo de mayor radio se recorrerá la curva de la clotoide desde el punto de tangencia con la curva de menor radio hasta el punto tangente de la curva de mayor radio.

Al pasar de un tramo de radio finito a un tramo de radio infinito, es necesario realizar cambios respecto al algoritmo anterior.

Se obtendrá el valor de "l" de la siguiente forma:

$$l = -\tau * 2 * R \tag{3.3.5}$$

Donde τ está expresada en radianes

En este caso el valor de L_0 será el que tenga valor 0 ($L_0 = 0$)

El valor de L se calcula como en el algoritmo anterior con la ecuación (3.3.4), es decir, L=-l.

De este modo, se obtendrán los valores de xref e yref aplicando las ecuaciones (5.26) y (5.28) para los valores entre l y L_0.

Donde $[x(l), y(l)]$ será el punto de la clotoide que será tangente al círculo y $[x(L_0), y(L_0)]$ es el punto de mayor radio de curvatura de la clotoide:

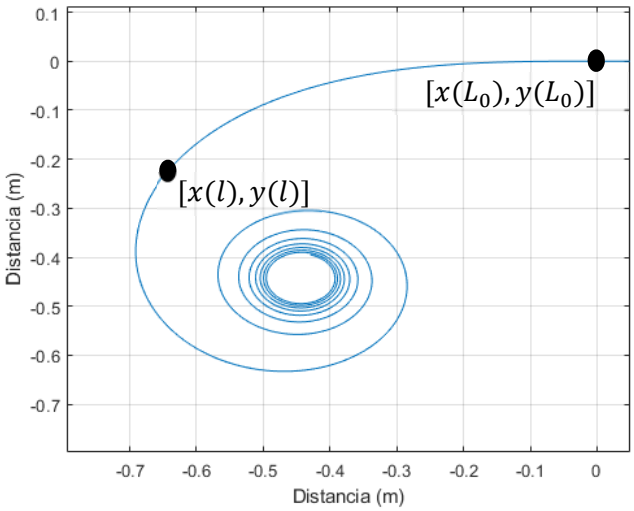

Figura 3.10 Clotoide que une un tramo con otro de mayor radio.

En resumen el algoritmo de la clotoide que une un círculo con una recta será:

Nombre: Algoritmo Círculo-Clotoide-Recta

Funcionalidad: Diseñar la clotoide que una un tramo circular con uno recto.

Parámetros de entrada: Radio de curvatura del tramo circular (R) y ángulo que forman la tangente del círculo en el punto de unión y la recta (V).

Parámetros de salida: Simulación de la trayectoria.

Algoritmo:

1. Calcular los valores de l (mediante la ecuación (3.3.5)) y L (mediante la ecuación (3.3.4)) de tal forma que L = -l.
2. Si l es menor que L, entonces se calcularán los valores de xref e yref mediante las ecuaciones (5.26) y (5.28).
3. Realizar el algoritmo de control visto en el apartado 3.2 para obtener la señal de control.
4. Si l es menor que L volver al paso 4, si se da el caso incrementar l en 0.01.
5. Realizar la simulación completa dándole a los motores la señal de la acción de control.

3.3.3. Caso Aplicado a una Situación de la Vida Real. Cambio de Sentido en una Rotonda.

Con estos conceptos, fue posible llevar las clotoides a un caso real, eligiendo la rotonda de "El Mirador" situada en la intersección entre la avenida de Cataluña y Tarongers, como lugar idóneo para este experimento. Fue elegida esta ubicación por ser bastante conocida, además de contar con tramos rectos y curvos, estos son la entrada, la salida y la propia rotonda.

La trayectoria escogida consta de tres tramos, primero un tramo recto, siendo este la entrada a la ciudad por la V21, un tramo circular correspondiente a la rotonda de 147 metros de diámetro, y un tramo recto correspondiente a la salida de la ciudad por la V21.

Con el fin de adecuar las medidas obtenidas en la realidad a las medidas con las que podemos trabajar con el robot en el laboratorio, se procedió a escalar la rotonda de manera que la rotonda pasara a medir 50 centímetros de radio.

Midiendo en el mapa se pudo obtener un ángulo V de 128.457º, tanto para la clotoide de entrada como para la de salida. Así, ya sabiendo los valores de R y V de ambas clotoides, se procedió a la generación de la referencia.

De este modo se crearon las referencias de los tramos rectos y el circular, para ello se usaron las funciones de la recta continua y la función del círculo.

A continuación se procedió a generar la clotoide de entrada mediante el algoritmo del apartado 3.3.1 y la de salida de la rotonda, utilizando el algoritmo del apartado 3.3.2.

Consiguiendo la trayectoria siguiente:

Figura 3.11. Clotoides en Rotonda

Lo que se puede ver en la figura 3.11 es que la trayectoria roja fue la referencia que se le pasó al robot, y la amarilla fue la que realmente siguió el robot.

Para el mismo caso se analizaron por separado, la evolución de las coordenadas en X del recorrido y por otro lado la evolución de las coordenadas en Y.

Posteriormente se procedió a estudiar el error de la trayectoria obtenida respecto a la esperada, de tal modo que se dedujo que apenas había error, pues el máximo error se da en las X con un error de solo 11 centímetros.

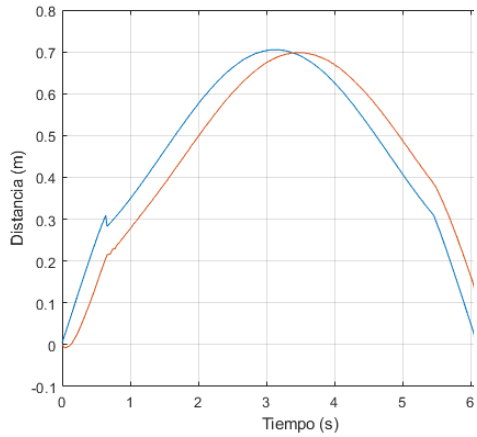

Figura 3.12 Señal del robot en X (rojo) vs. referencia en X (azul)

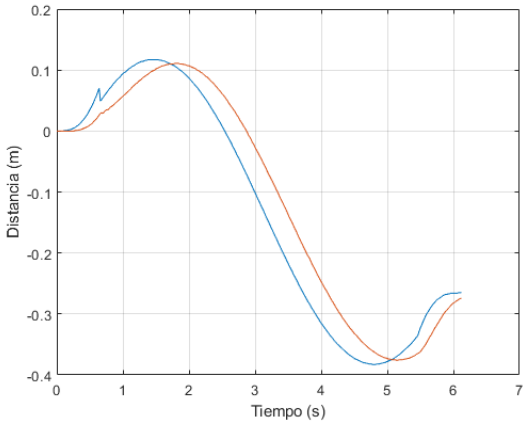

Figura 3.13 Señal del robot en Y (rojo) vs. referencia en Y (azul)

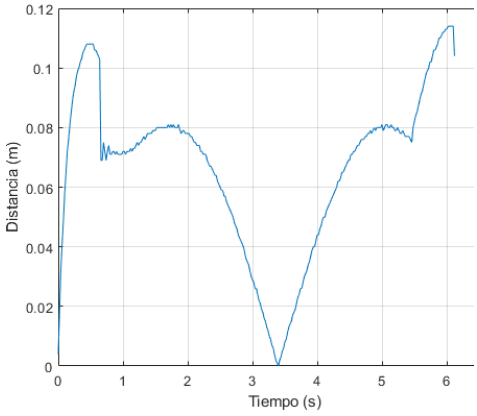

Figura 3.14 Error en X

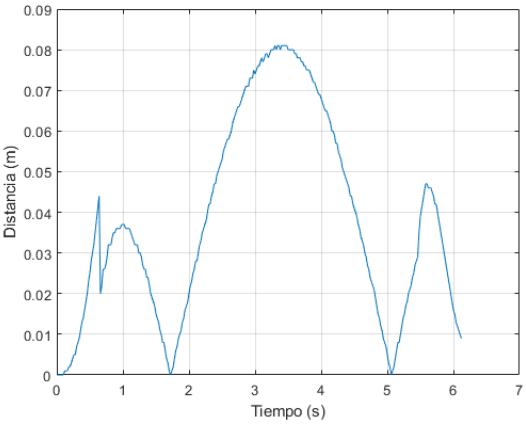

Figura 3.15 Error en Y

Para está configuración se recogió la señal que producía la acción de control, a modo de comprobar que está saturada entre 100 y -100.

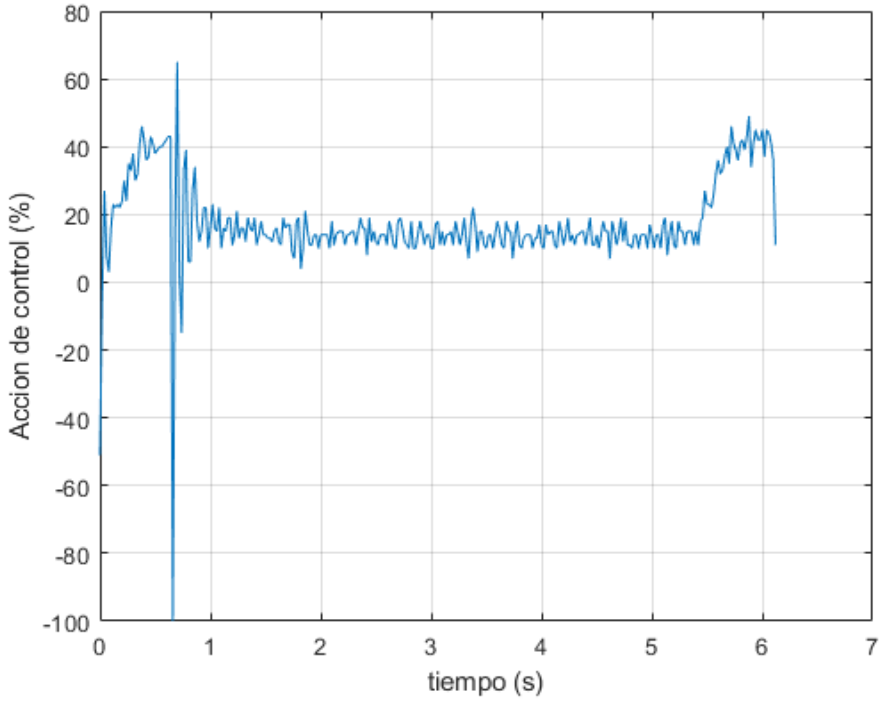

Figura 3.16 Acción de Control de la Rueda Derecha.

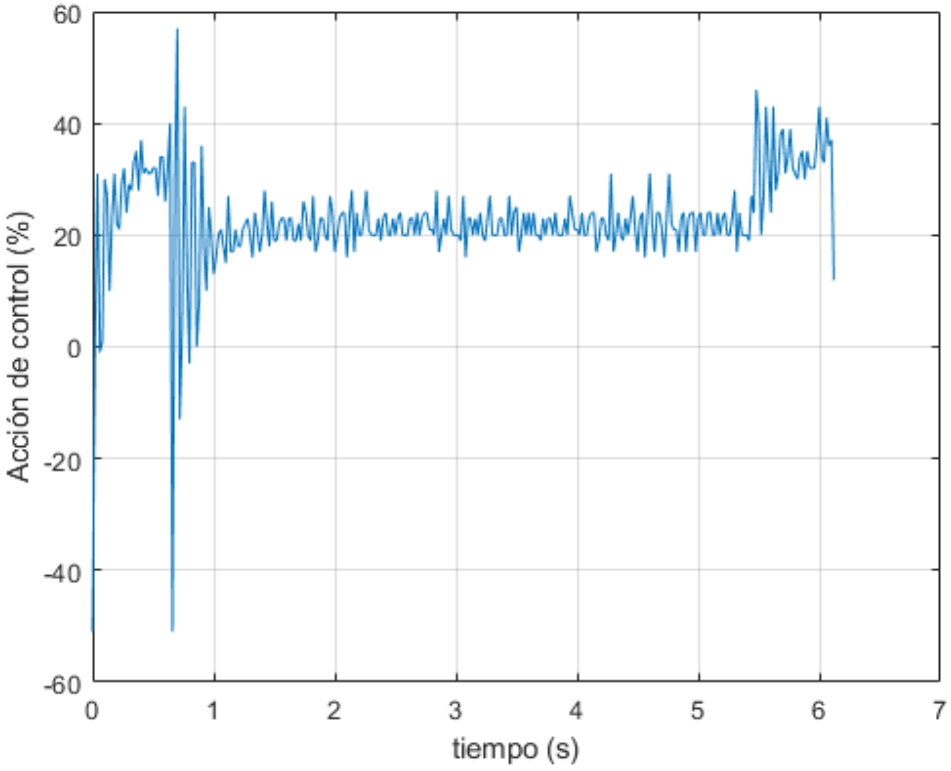

Figura 3.17. Acción de Control de la Rueda Izquierda

En resumen el algoritmo que se ha seguido para generar la trayectoria que hace un cambio de sentido en la rotonda de "El Mirador" es:

Nombre: Algoritmo Rotonda

Funcionalidad: Diseñar la trayectoria que trace un cambio de sentido en una rotonda.

Parámetros de entrada: Radio de curvatura del tramo circular (R) y ángulo que forman la tangente del círculo en el punto de unión y la recta.

Parámetros de salida: Simulación de la trayectoria.

Algoritmo:

1. Medir en el mapa para sacar los valores de R (0.5 metros) y V (128.457º).
2. Implementar las funciones (para xref e yref) de la recta, que hace de V21 entrada en la rotonda.
3. Calcular el valor de L utilizando la ecuación (3.3.4).
4. Generar la clotoide que une el tramo recto con el círculo de radio R (0.5 metros) mediante las ecuaciones (5.26) y (5.28), para las cuales el valor de s va desde 0 hasta L.
5. Calcular las funciones (xref e yref) del círculo, que hace de rotonda.
6. Obtener el valor de "l" mediante la ecuación (3.3.5)
7. Generar la clotoide que une el círculo con la recta utilizando las ecuaciones (5.26) y (5.28), para las cuales s va desde "l" a 0.
8. Calcular las funciones (xref e yref) de la recta que hace de V21 salida de rotonda.
9. Realizar el algoritmo de control visto en el apartado 3.2, con el fin de generar la señal de control.
10. Simular la trayectoria con el robot, introduciendo en los motores la señal de acción de control obtenida.

3.3.4. Generalización de las Clotoides

Finalmente, se procedió a crear la función que permita editar fácilmente la clotoide que une dos tramos, teniendo en cuenta los radios de curvatura de ambos tramos y sus posiciones.

Para generar cualquier clotoide, es necesario saber cuáles son los radios del tramo anterior (R_0) y del posterior R, además del ángulo V, pues dependiendo de cuál de los dos sea mayor, los puntos de definición de la clotoide serán unos u otros. Además dependiendo de los radios la longitud de la curva será mayor o menor.

Una vez conocido el valor de L según la ecuación (3.3.4), es posible conocer el punto de tangencia entre la clotoide y el primer tramo gracias a la propiedad fundamental de las clotoides, esta propiedad es la que dice que el radio de curvatura de un punto de la clotoide por la longitud de la clotoide recorrida es constante. De este modo, sabiendo R y L se obtiene la constante (A^2), y haciendo el cociente:

$$L_0 = \frac{A^2}{R_0} \tag{3.3.6}$$

Se consiguió el valor de la longitud de clotoide recorrida hasta el punto tangente con el primer tramo (L_0).

De modo que el tramo de clotoide que nos interesa será el que generen las ecuaciones (5.26) y (5.28) dando a s los valores entre L_0 y L

```
taurad=tau*(PI/180);

L=taurad*2*R;

A2=R*L;

Lo=A2/Ro;

l=Lo;
```

En el caso de que el primer tramo tenga un radio de curvatura menor que el segundo, el valor de L_0 será el que se obtiene de aplicar:

$$L_0 = -\tau * 2 * R_0 \tag{3.3.7}$$

Conociendo L_0 y R_0 se obtiene el valor de La constante, mediante la propiedad fundamental de las clotoides, a la inversa que en el caso anterior se obtiene L:

$$L = \frac{A^2}{R} \tag{3.3.8}$$

Sabiendo esto, podrá ser generada la clotoide aplicando las ecuaciones (5.26) y (5.28) tomando s valores entre L_0 y L igual que sucedía en el caso anterior.

El algoritmo capaz de generar cualquier clotoide es el siguiente:

Nombre: Algoritmo Clotoide

Funcionalidad: Diseñar la clotoide en función de los tramos anterior y posterior.

Parámetros de entrada: Radio de curvatura del tramo anterior (R_0), radio de curvatura del tramo posterior (R) y ángulo que forman la tangente del círculo en el punto de unión y la recta.

Parámetros de salida: Señal de referencia.

Algoritmo:

1. Si R_o es mayor que R, entonces se calcula el valor de L mediante la ecuación (3.3.4) y se obtiene L_0 a partir de la propiedad característica de las clotoides.
2. Si R es mayor que R_0, entonces se calcula el valor de L_0 mediante la ecuación (3.3.7) y se obtiene L a partir de la propiedad característica de las clotoides.
3. Generar la clotoide mediante las ecuaciones (5.26) y (5.28) tales que los valores de s van de L_0 a L.

3.4. Desarrollo de Trayectorias mediante Campos Potenciales

Esta parte del trabajo se ha desarrollado en colaboración con otro Trabajo Fin de Grado que estaba siendo elaborado en paralelo. Este segundo trabajo consistía en la persecución de robots móviles, mediante una cámara cenital que lee la posición del robot en cabeza, generando así la referencia que debe seguir el robot en cola. Ese trabajo ha utilizado la generación de trayectorias por medio de campos potenciales del presente TFG para definir la trayectoria del robot en cabeza.

El resultado de esta colaboración ha generado resultados de trabajo para ambos TFGs, cada uno en sus respectivos propósitos.

Han sido utilizados tanto Robot C para la programación del robot NXT, como Matlab para una más rápida visualización de la generación de la referencia.

Los robots utilizados en esta parte han sido tanto el NXT, que es el mismo que se utilizó en el punto 3.3. de este TFG, como el EV3, que es el robot que se utilizaba en el otro Trabajo Fin de Grado.

Ambos robots han sido diseñados físicamente bajo una Configuración diferencial porque, como ha sido comentado anteriormente, es la configuración más efectiva.

3.4.1. Definición de los Elementos del Entorno.

Como se va a trabajar en un entorno interior delimitado por paredes, se creó un cerramiento que delimitara el espacio de trabajo del robot, para este cerramiento se eligió que presentara una forma rectangular para dar simpleza al entorno y porque es la forma más común dentro de los posibles entornos. Este cerramiento se diseñó de manera que pueda ser fácilmente editable, cambiando las coordenadas de sus vértices.

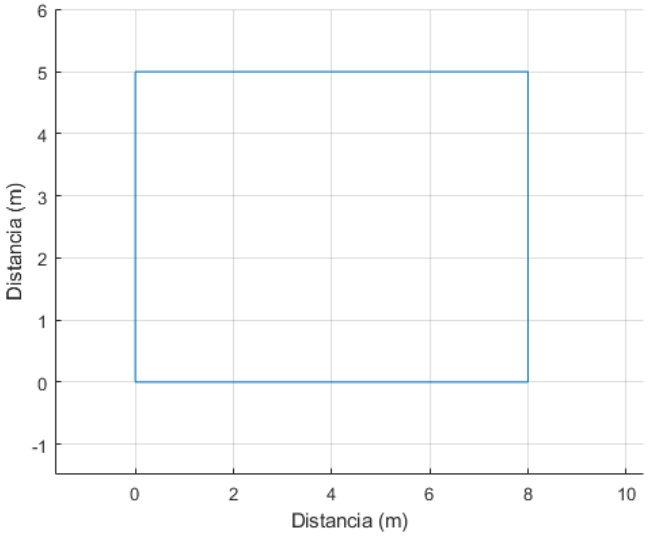

Figura 3.18 Definición del Entorno.

Dentro del recinto se situaron una serie de obstáculos repartidos. La forma de estos obstáculos puede ser tan irregular como se desee, pues este siempre podrá ser asimilado a un volumen virtual regular, por ello se ha trabajado con obstáculos prismáticos capaces de contener al obstáculo real. Los obstáculos serán fácilmente modificados pudiendo cambiar fácilmente la posición de su centro y sus dimensiones.

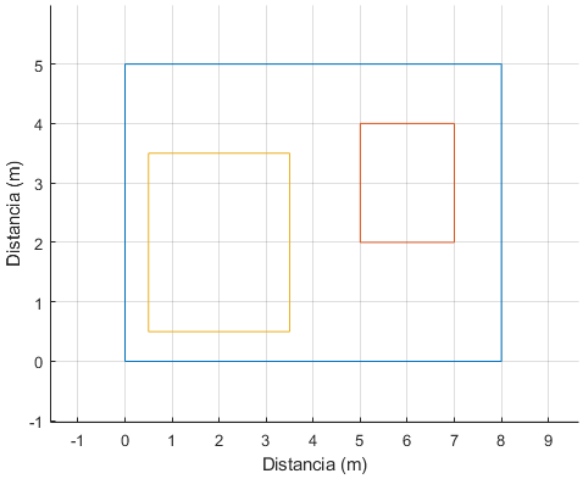

Figura 3.19 Definición Obstáculos Fijos.

Así mismo, dentro del espacio también se encontrará el punto inicial y el punto final del robot, permitiendo cambiar la posición de éstos a cualquier punto del plano dentro del recinto.

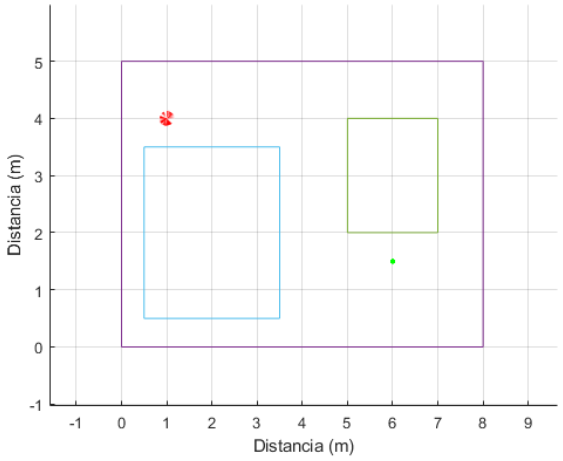

Figura 3.20 Definición de Punto Inicial y Meta.

En el caso de que se quiera trabajar con obstáculos móviles, será necesario, en primer lugar, definir la trayectoria que seguirá el obstáculo, en este proyecto únicamente se ha trabajado con obstáculos móviles con trayectoria lineal.

Se ha elegido que estos obstáculos, en cambio, tengan forma cilíndrica, con el fin de diferenciarlos de los obstáculos fijos, y también para demostrar que es posible crear campos potenciales con otras geometrías y no sólo está el prisma.

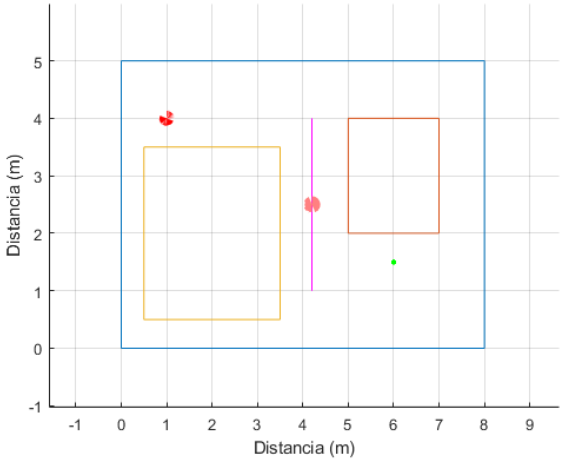

Figura 3.21 Definición Obstáculos Móviles.

3.4.2. Construcción de los Campos Potenciales

En todo momento se ha trabajado en el plano OXY.

La construcción de los campos potenciales es muy importante en este trabajo, pues la resultante de todos los campos potenciales que influyen en el robot es la que establece la dirección y velocidad a la que se moverá el robot.

En primer lugar, es necesario conseguir que el robot sea capaz de llegar hasta la meta. Para ello se implementó el algoritmo que calcula el campo potencial de atracción de la meta utilizando la ecuación (6.2). Con el producto del campo por la distancia entre el robot y la meta, se consigue la dirección hacia la cual avanzará el robot en la siguiente iteración, de manera que se encontrará en una posición más cercana al punto deseado.

El algoritmo que permite que el robot alcance la meta es:

Nombre: Algoritmo Campo Potencial de Atracción

Funcionalidad: Diseñar la trayectoria que permita al robot avanzar hasta llegar a la meta.

Parámetros de entrada: Posición inicial del robot posición de la meta.

Parámetros de salida: Señal de referencia (trayectoria esperada del robot).

Algoritmo:

1. Calcular la distancia mínima entre robot y meta.
2. Calcular el campo potencial de atracción usando la ecuación (6.2) a partir de la distancia medida.
3. Hacer el producto del campo por la distancia. Obteniéndose así la magnitud avanzada por el robot en “x” y en “y”.
4. Añadirle a la posición actual del robot el avance producido por el campo potencial, para obtener la nueva posición del robot.
5. Si no se ha alcanzado la meta, entonces volver al paso 1.

Siendo el campo potencial de atracción el que permite conseguir el camino óptimo, pues siempre buscará la posible posición que le acerque al punto final.

Una vez que el robot ha sido capaz de alcanzar la meta, se decidió trabajar en entornos donde todos los obstáculos presentes eran fijos y con forma rectangular, paralelos a las paredes del entorno. Estos estarían ubicados en un recinto cerrado lo suficientemente grande para que quepan los obstáculos y para que el robot pueda moverse fácilmente.

Teniendo en cuenta que el espacio de trabajo es un entorno cerrado y delimitado por paredes, se asumió que las paredes del entorno actuarán como obstáculos fijos impidiendo que el robot impacte contra ellas.

El siguiente paso fue, por tanto, evitar la colisión con los obstáculos fijos, para ello se implementó el algoritmo que calcula los campos potenciales de repulsión de los obstáculos, pero también el de las paredes del entorno.

Estos campos se calculan mediante la ecuación (6.4) para los cuales es necesario conocer la distancia mínima entre el robot y el obstáculo y la máxima distancia del obstáculo a la que afecta el campo potencial.

Al hacer el producto entre estos campos y la distancia entre el robot y el centro de los obstáculos, se obtiene la dirección que seguirá el robot en la próxima iteración, modificándose así la trayectoria del robot, de tal forma que si el robot se encuentra lo suficientemente cerca de ellos, buscará una posición más alejada del obstáculo.

Al sumar los efectos del campo de atracción y los de los campos de repulsión, es decir, sumar las direcciones que proporcionan los campos, el robot buscará una posición que lo acerque al punto final, siempre que no actué sobre él algún campo de repulsión, entonces se acercará, pero bordeando el obstáculo.

El algoritmo que permite evitar los obstáculos fijos es el siguiente:

Nombre: Algoritmo Entorno con Obstáculos Fijos.

Funcionalidad: Diseñar la trayectoria que permite al robot evitar los obstáculos fijos y llegar a la meta.

Parámetros de entrada: Posición inicial del robot, posición de la meta, posición del centro de los obstáculos fijos, dimensiones de los obstáculos fijos y máxima distancia de influencia del campo potencial de repulsión medida desde la pared del obstáculo.

Parámetros de salida: Señal de referencia (trayectoria esperada del robot).

Algoritmo:

1. Calcular la distancia mínima entre robot y meta.
2. Calcular la distancia mínima entre robot y centros de los obstáculos.
3. Calcular el campo potencial de atracción usando la ecuación (6.2) a partir de la distancia medida.
4. Hacer el producto del campo por la distancia. Obteniéndose así la magnitud avanzada por el robot en “x” y en “y”.
5. Calcular los campos potenciales de cada uno de los obstáculos mediante la ecuación (6.4).
6. Hacer el producto de campo potencial por distancia entre robot y centro del obstáculo, obteniendo así el avance que provoca sobre el robot cada obstáculo.
7. Sumar a la posición actual del robot el avance producido por cada obstáculo y el avance producido por el campo de atracción de la meta, visto en el algoritmo que permite que el robot alcance la meta, para obtener la nueva posición del robot.
8. Si no se ha alcanzado la meta volver al paso 1.

Con este algoritmo, se puede observar en las siguientes simulaciones diferentes trayectorias, debido al diseño de diversos entornos con diferente número y distribución de los obstáculos, en los cuales se consigue el objetivo de alcanzar el punto final sin colisionar con ningún obstáculo:

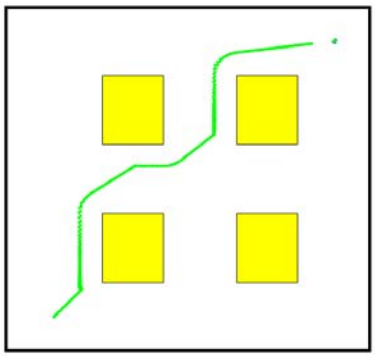

Figura 3.22 Campos Potenciales 4 Obstáculos.

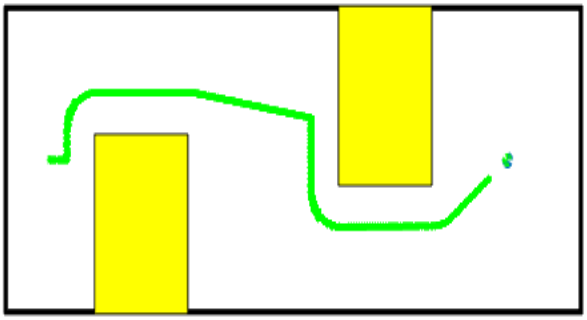

Figura 3.23 Campos Potenciales Trayectoria en S.

Viendo que el robot podía sortear los obstáculos fijos se procedió a complicar un poco el entorno introduciendo obstáculos móviles.

De esta manera se introdujeron unos obstáculos circulares, con movimiento lineal realizando recorridos de ida y vuelta entre dos puntos definidos.

Con el fin de poder eludir estos obstáculos también, se les dotó de campos potenciales de repulsión, con el propósito de modificar la trayectoria del robot si estos se cruzan en su camino.

El algoritmo que calcula la repulsión debida a los obstáculos móviles es diferente al de los obstáculos fijos al calcular el avance utilizando la siguiente ecuación:

$$dx = \rho(q, q_{mov}) * e^{\frac{-|\rho(q,q_{mov})|}{K}} \quad (3.4.1)$$

Siendo dx la dirección hacia la que empujará el campo de repulsión del obstáculo móvil, $\rho(q, q_{mov})$ la mínima distancia entre el robot y el obstáculo móvil y K una constante a la que se le ha dado el valor 0.03.

El algoritmo que calcula el avance producido por los obstáculos móviles es:

Nombre: Algoritmo Entorno con Obstáculos Fijos y Móviles.

Funcionalidad: Diseñar la trayectoria que permite que el robot evite los obstáculos fijos y móviles y que el robot llegue a la meta.

Parámetros de entrada: Posición inicial del robot, posición de la meta, posición de los centros de los obstáculos fijos, dimensiones de los obstáculos fijos, máxima distancia de influencia del campo potencial de repulsión medida desde la pared del obstáculo, puntos extremos del recorrido del obstáculo móvil y posición inicial del obstáculo móvil

Parámetros de salida: Señal de referencia (trayectoria esperada del robot móvil).

Algoritmo:

1. Calcular la distancia mínima entre robot y meta.
2. Calcular la distancia mínima entre robot y centros de los obstáculos.
3. Calcular la distancia mínima entre robot y obstáculo móvil.
4. Calcular el campo potencial de atracción usando la ecuación (6.2) a partir de la distancia medida.
5. Hacer el producto del campo por la distancia. Obteniéndose así la magnitud avanzada por el robot en “x” y en “y”.
6. Calcular los campos potenciales de cada uno de los obstáculos mediante la ecuación (6.4).
7. Hacer el producto de campo potencial por distancia entre robot y centro del obstáculo, obteniendo así el avance que provoca sobre el robot cada obstáculo.
8. Calcular el avance del robot producido por el obstáculo móvil mediante la ecuación (3.4.1).
9. Calcular la nueva posición del obstáculo, teniendo en cuenta que su recorrido es de ida y vuelta entre dos puntos definidos.
10. Calcular la nueva posición del robot sumando a la posición actual el avance producido por el obstáculo móvil, los avances producidos por los obstáculos fijos (si los hay) , el avance que pueda producirse por el entorno y el avance debido a la atracción del punto final.
11. Si no se ha alcanzado la meta, volver al punto 1.

Realizando la simulación del anterior algoritmo se obtuvo:

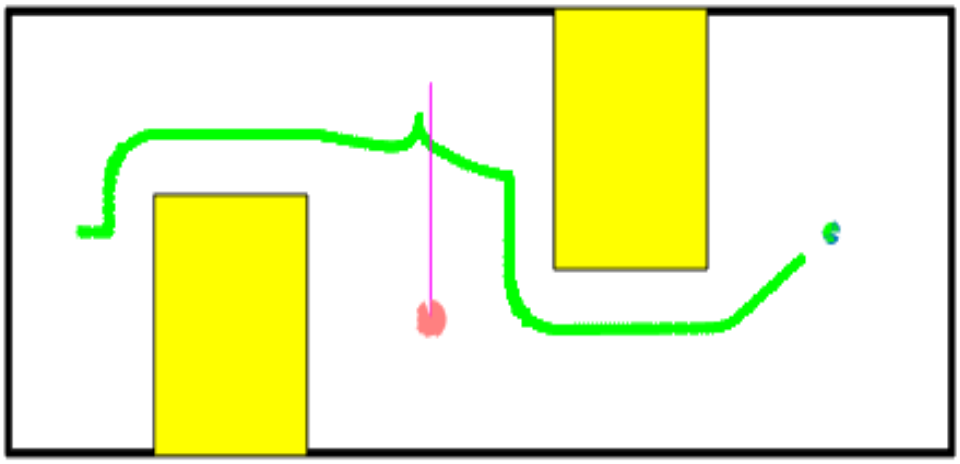

Figura 3.24 Obstáculos Móviles.

Como se puede ver en la figura 3.24, el robot es capaz de modificar su trayectoria en el momento que se encuentra con el obstáculo móvil.

3.4.3. Simulación.

El siguiente paso fue traducir todo el algoritmo desarrollado en Matlab para la obtención del algoritmo de los campos potenciales artificiales, a lenguaje C, de modo que pueda ser implementado en Robot C, con la finalidad de poder simular el recorrido con el robot móvil NXT.

La implementación de este algoritmo en Robot C se realizó mediante varias funciones, como la función que calcula el campo potencial de la meta, a partir de la distancia entre el robot y la meta, y aplicando la ecuación (6.2); también fue utilizada la función que calcula el campo de repulsión a partir de la posición central del obstáculo, la distancia mínima entre el robot y el obstáculo y la distancia máxima, respecto al obstáculo, a la que se quiere que afecte el campo potencial, esta función será la que se aplique tanto a obstáculos cómo a las paredes del entorno, utilizando la ecuación (6.4); también se pensó una función que calculara la distancia mínima entre obstáculo y robot, esta función necesitará los puntos de los vértices del obstáculo y la posición del robot.

El algoritmo seguido para generar las simulaciones con obstáculos fijos y móviles es el siguiente:

Nombre: Algoritmo Campos Potenciales Artificiales Completo.

Funcionalidad: Diseñar y generar la trayectoria que permite que el robot evite los obstáculos fijos y móviles y que el robot llegue a la meta.

Parámetros de entrada: Posición inicial del robot, posición de la meta, posición de los centros de los obstáculos fijos, dimensiones de los obstáculos fijos, máxima distancia de influencia del campo potencial de repulsión medida desde la pared del obstáculo, puntos extremos del recorrido del obstáculo móvil y posición inicial del obstáculo móvil

Parámetros de salida: Simulación de la trayectoria

Algoritmo:

1. Calcular la distancia mínima entre el robot y los diferentes elementos: obstáculos fijos, obstáculos móviles, paredes del entorno y punto final.
2. Calcular los campos de repulsión de obstáculos fijos y del entorno, conocida la mínima distancia entre el robot y ellos y la máxima distancia de influencia respecto las paredes.
3. Calcular el campo de atracción de la meta.
4. Calcular el avance producido por cada obstáculo fijo, por las paredes del entorno, por cada obstáculo móvil y por la meta.
5. Sumar todos los avances calculados a la posición actual del robot para obtener la nueva posición.
6. Calcular las nuevas posiciones de los obstáculos móviles.
7. Generar la velocidad de referencia, como derivada de la posición de referencia, obtenida de la trayectoria de referencia.
8. Obtener el valor de los encoders que dan la posición del robot.
9. Calcular la velocidad angular de las ruedas del robot.
10. Calcular la velocidad lineal de las ruedas del robot, como velocidad angular por radio.
11. Calcular la velocidad lineal del robot.
12. Calcular la velocidad angular del robot.
13. Calcular la posición y orientación del robot.
14. Realizar el control cinemático del robot.
15. Calcular el modelo cinemático inverso del robot.
16. Calcular la velocidad angular de referencia de las ruedas.
17. Calcular el error entre la velocidad angular de las ruedas del robot y la velocidad angular de referencia de las ruedas.
18. Calcular la acción de control del control dinámico proporcional.
19. Saturar la acción de control.
20. Aplicar la acción de control a los motores del robot.
21. Actualizar los valores de los encoders para la siguiente iteración
22. Si el robot no ha alcanzado la meta, entonces volver al paso 1.

3.4.4. Generalización del Algoritmo de Campos Potenciales Artificiales.

Con la finalidad de facilitar la edición del algoritmo se expresaron las diferentes variables en función de las posiciones del robot, los obstáculos y de la meta, de forma que se podrán generar cualquier entorno dando las **posiciones inicial y final del robot**, además de las **posiciones y dimensiones de los distintos obstáculos**. Esto fue realmente útil para el trabajo que estaba siendo realizado en paralelo, ya que mediante la cámara cenital se podría leer la posición tanto de los obstáculos, como de las posiciones iniciales y finales del robot.

De este modo, en el algoritmo Campos Potenciales Artificiales Completo, las entradas serían el valor de las coordenadas del punto inicial en X "xant" y en Y "yant", el valor de las coordenadas del punto final en X "xf" y en Y "yf", la matriz de posiciones de los centros de los obstáculos fijos "obs", donde la primera columna son las coordenadas en X, la segunda columna las coordenadas en Y, y el número de filas dirán cuántos obstáculos fijos hay. También se introducirán dos vectores más, uno con el ancho "w" y el otro con el largo "h" de cada obstáculo, pudiendo sacar el número de obstáculos según la longitud de estos. Además se introducirán una matriz con las posiciones iniciales de los obstáculos móviles "posobs", un vector con las orientaciones de la trayectoria de cada obstáculo "or", siendo 0 si es paralelo al eje de las X y 1 si es paralelo al eje de las Y y una matriz con los puntos extremos de los recorridos "rngmov".

```
[xnew, ynew] =CP (xant, yant, obs, posobs, rngmov, or, xf, yf, w, h)
```

Gracias al algoritmo desarrollado en Matlab podemos graficar cómo afectan los campos potenciales al robot:

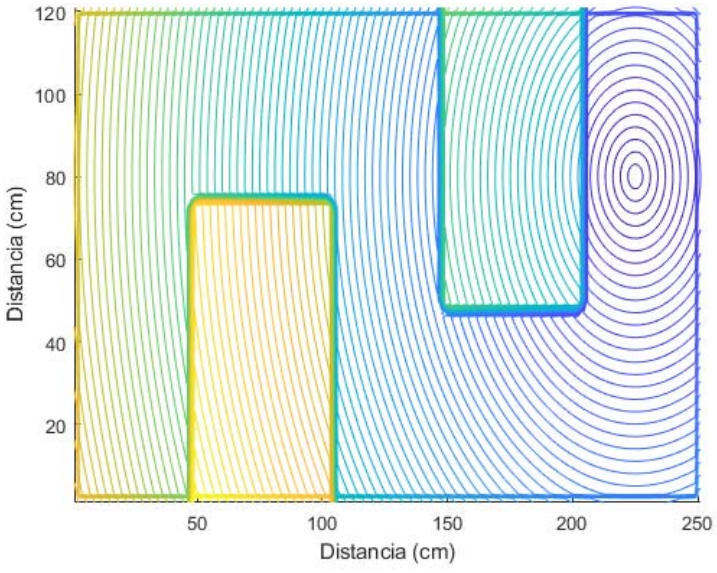

Figura 3.25. Representación del Campo Potencial.

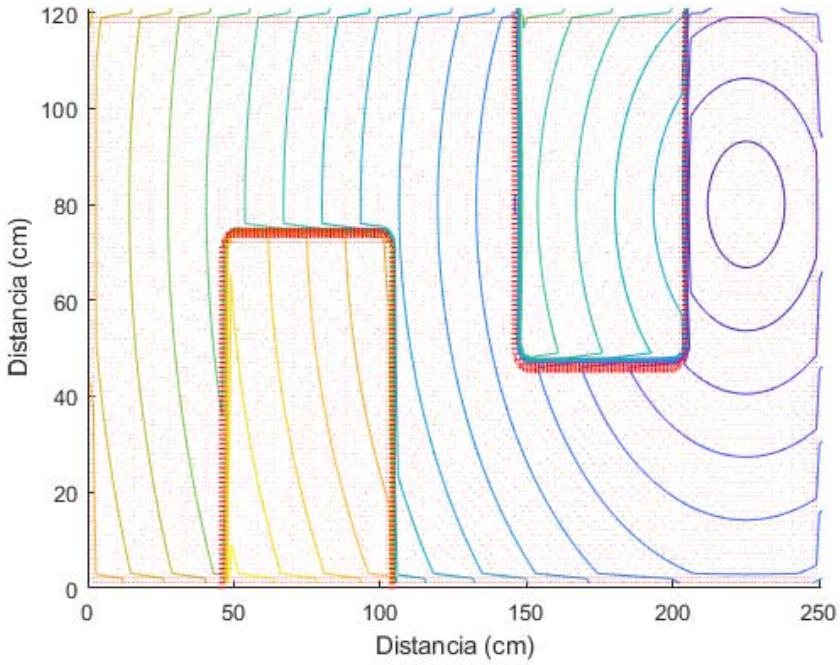

Figura 3.26 Representación de las Fuerzas Potenciales.

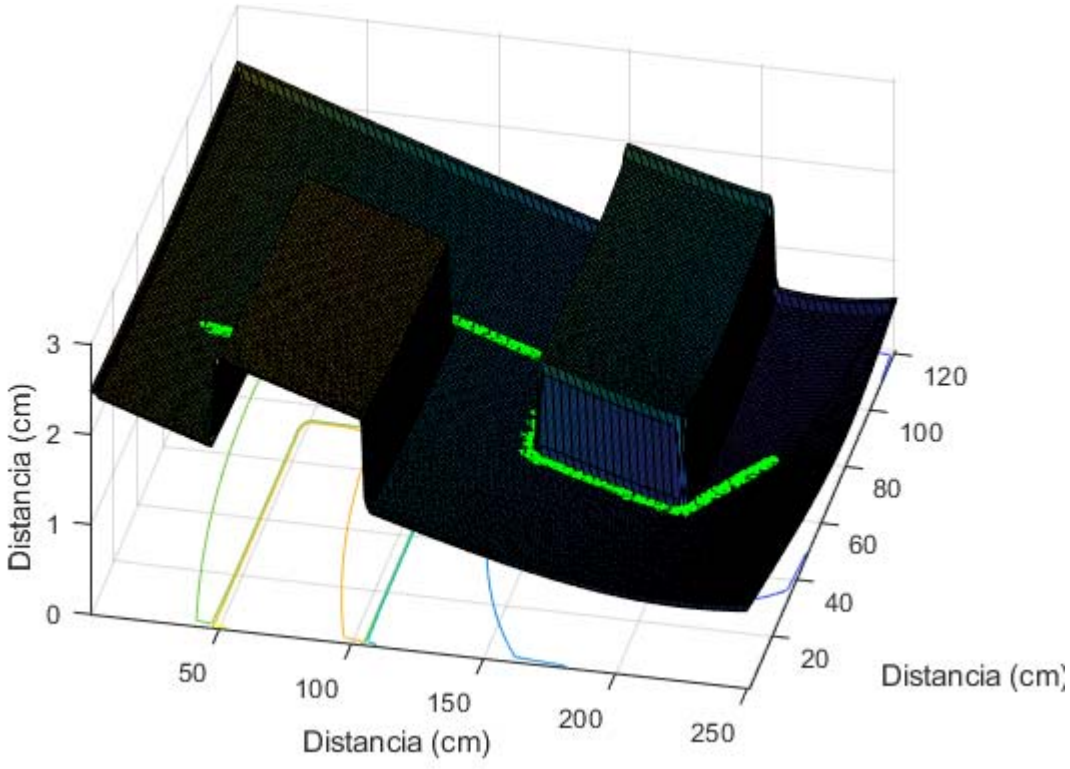

Figura 3.27 Representación en 3D.

Una vez conseguido que el robot realice el trayecto esperado, se analizaron las acciones de control, viendo que esta no se saturaba:

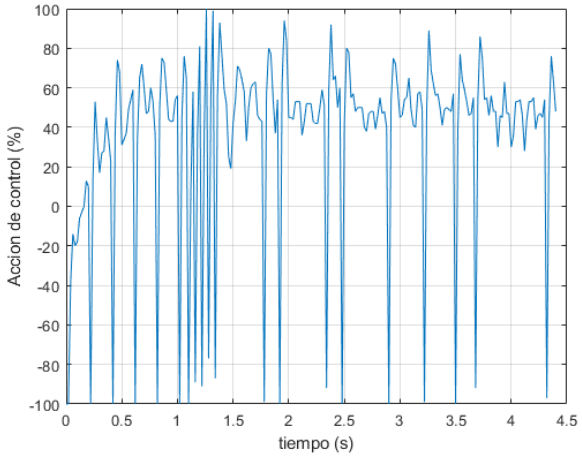

Figura 3.28 Acción de Control de la Rueda Derecha.

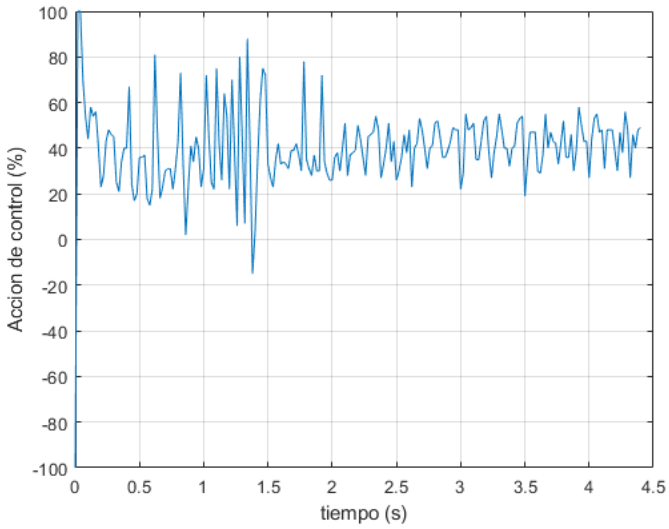

Figura 3.29 Acción de Control de la Rueda Izquierda.

3.4.5. Problemas de los Campos Potenciales Artificiales.

Sin embargo este método no es infalible, ya que se puede dar el caso, para diferentes configuraciones, en el que se llegue a un punto donde la fuerza de los campos de atracción y repulsión se anule, de esta forma el robot se quedará atascado, no pudiendo alcanzar nunca la posición final.

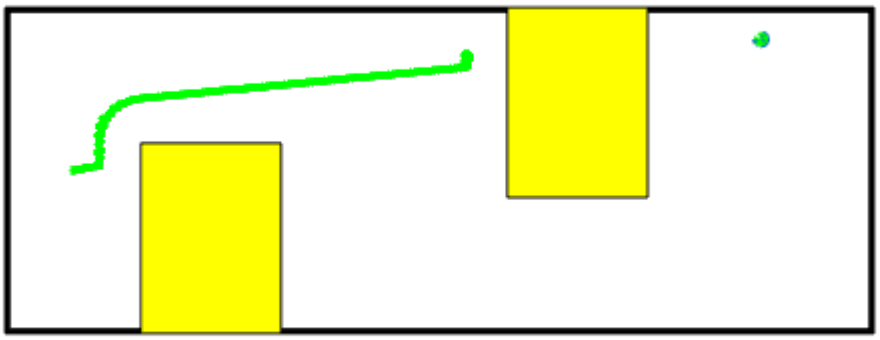

Figura 3.30 Robot Estancado.

Este problema puede ser solucionado, obligando que el robot se mueva en una dirección cualquiera, cuando lleve varias iteraciones en el mismo punto.

De este modo, a medida que el algoritmo se ejecuta, el programa lee la posición actual y la compara con la anterior, de modo que si apenas se ha movido incrementará un contador. Cuando este contador sea lo suficientemente grande, significará que el robot lleva demasiado tiempo en la misma posición y entonces será necesario obligarle a que se mueva en una dirección para así desatascarlo.

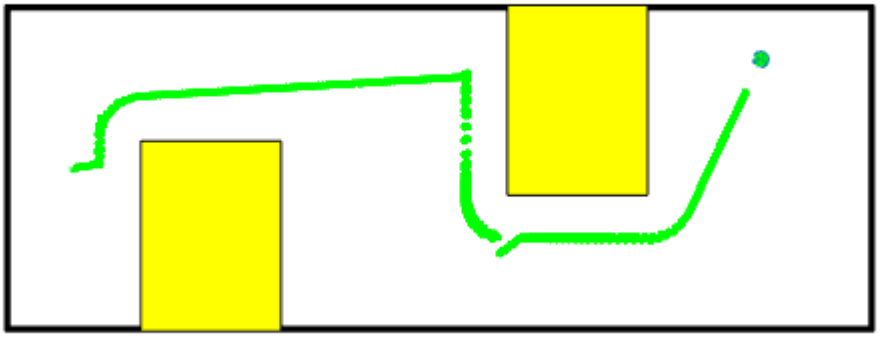

Figura 3.31 Solución al Estancamiento.

4. CONCLUSIONES

A partir de los resultados del proyecto, se pueden extraer algunas conclusiones.

Para empezar, hay que aclarar que los objetivos principales del proyecto han sido alcanzados. Los dos problemas principales que se han planteado en este trabajo han sido resueltos, la generación de trayectorias suaves, gracias a la unión entre tramos mediante clotoides, y la evitación de obstáculos, gracias al algoritmo de campos potenciales artificiales.

La generación de trayectorias suaves se ha logrado con la modelización de los diferentes tramos del circuito, siendo éstos rectas o círculos. Saber cómo generar las clotoides recta-círculo, como se hizo en el apartado 3.3.1, saber cómo generar las clotoides círculo-recta, visto en el apartado 3.3.2, y a partir de ahí saber cómo generar clotoides capaces de unir un tramo con otro de menor radio de curvatura y generar también clotoides capaces de unir un tramo con otro de mayor radio de curvatura. Finalmente, los resultados han podido ser contrastados experimentalmente gracias a la simulación de la trayectoria del robot, tanto vía Simulink, como a través del robot NXT. Con esto se ha podido analizar su incidencia en una posible aplicación en la vida real, apartado 3.3.3, y se ha logrado la creación de clotoides de manera más fácil y rápida en el apartado 3.3.4.

La evitación de obstáculos se logró gracias a la definición del entorno, lo que incluye la modelación de las paredes del recinto, de los obstáculos fijos, de los obstáculos móviles y de los puntos inicial y final, como ha quedado recogido en el apartado 3.4.1. Un hecho muy importante fue conseguir que el robot alcanzara la meta, lo cual se consiguió gracias al campo potencial de atracción situado en la meta. Igual que fue importante evitar los obstáculos fijos y los obstáculos móviles, mediante campos potenciales de repulsión. Gracias a esto el robot consiguió llegar a la meta sin colisionar, y siguiendo el camino óptimo, que es el que permite realizar el trabajo en el menor tiempo posible. Todo esto se trabajó en el apartado 3.4.2. Los resultados pudieron ser comprobados experimentalmente gracias a la simulación de la trayectoria del robot, tanto en MATLAB, como a través del robot NXT, obteniéndose los resultados que se muestran en el apartado 3.4.3. Finalmente, para poder realizar el algoritmo de campos potenciales artificiales en cualquier entorno, se generó una función que calcula el algoritmo conociendo los puntos de salida y llegada y las posiciones y dimensiones de los obstáculos, dentro del apartado 3.4.4.

El trabajo desarrollado ha tenido **una parte de revisión teórica** centrada en los fundamentos que después iban a ser críticos en la solución de los problemas planteados (generación de trayectorias suaves y evitación de obstáculos). Y ha tenido **una parte aplicada**, donde se ha hecho el desarrollo de las ecuaciones generales, pasándolas a las soluciones específicas que se buscaban. Se han construido a partir de ahí, lo modelos que dieran soporte a un posterior experimento. Y dicho **experimento** se ha llevado a cabo **en dos fases**:

1. La comprobación del resultado del modelo elaborado para la generación de trayectorias suaves, a través de la simulación de la trayectoria vía la herramienta de MATLAB, Simulink. Y posteriormente con la contrastación de resultados utilizando el robot NXT de Lego.
2. Y la comprobación del resultado del modelo cuando existe un entorno formado por obstáculos fijos y móviles, y el robot tiene que hacer un recorrido que alcance de manera

eficiente el objetivo mediante la observación del comportamiento del robot NXT dentro de ese mismo espacio.

Cabe decir que a medida que se ha estado realizando este trabajo han surgido diferentes problemas, que finalmente se han podido resolver. Como es el caso de los estancamientos en la generación de trayectorias usando el algoritmo de campos potenciales artificiales.

Pese a que las soluciones adoptadas han sido eficaces como solución a los problemas, estas se pueden considerar algo simples y podrían ser mejoradas con una mayor investigación y desarrollo, consiguiendo métodos más sofisticados que no presenten los problemas que estos generan.

Se abren por tanto, nuevas líneas de análisis e investigación como son la solución de problemas de los estancamientos cuando se contrarresten los campos potenciales de atracción y repulsión. Nada dice que la mejor solución, no sea la adoptada en nuestro caso, pero se entiende conveniente una exploración específica de este subproblema del proceso.

Finalmente, cabe destacar como logro relevante de este proyecto, que ofrece una solución contrastada experimentalmente al principal problema de la navegación en los robots móviles, que es generar trayectorias eficaces y eficientes, y guiar su movimiento según estas, permitiendo al robot desplazarse por el ambiente de trabajo de manera segura, evitando colisiones.

5. REFERENCIAS

Badesa, F.J. y otros (2015). *Métodos de control basados en campos potenciales y fuerza para robótica de rehabilitación.* Actas de la XXXVI Jornadas de Automática. Bilbao. 2-4 de septiembre. ISBN: 978-84-15914-12-9.

Blanch, L. y otros (2013). "Una aproximación a la curva de transición Clotoide vista desde Mathematica". *Modelling in Science Education and Learning. Volumen 6 (2). Nº 9.*

Cervera Andés, A. (2011) *Coordinación y control de robots móviles basado en agentes. Proyecto Final de Carrera*. Escuela Técnica Superior de Ingeniería Informática. Universidad Politécnica de Valencia. Julio.

García Martínez, E. (2018). *Multi-Robot Path Planning in Dynamic Environment.* Project/Dissertation. Sheffield Hallam University. June.

Martínez, S. y Sisto, R. (2009). *Control y Comportamiento de Robots Omnidireccionales. Proyecto de Grado. Instituto de Computación.* Facultad de Ingeniería de la Universidad de la República. Montevideo (Uruguay). Mayo.

https://www.hisour.com/es/mobile-robot-42899/ (17 de Julio de 2019)

https://robotica.wordpress.com/about/ (16 de Junio del 2019)

http://matematiquemos.blogspot.com/2014/09/la-lemniscata-de-bernoulli.html (14 de Junio del 2019)

http://help.autodesk.com/view/CIV3D/2015/ESP/?guid=GUID-DD7C0EA1-8465-45BA-9A39-FC05106FD822 (14 de Junio del 2019)

https://viasi.weebly.com/uploads/4/3/2/7/4327492/clase4._alineamiento_horizontal3.pdf (14 de Junio del 2019)

http://delegacion.topografia.upm.es/wp-content/uploads/2016/03/CAP%C3%8DTULO-6-PRIMERA-PARTE.pdf (20 de Junio del 2019)

http://delegacion.topografia.upm.es/wp-content/uploads/2016/03/CAP%C3%8DTULO-6-SEGUNDA-PARTE.pdf (20 de Junio del 2019)

https://cifrasyteclas.com/clotoide-la-curva-que-vela-por-tu-seguridad-en-carreteras-y-ferrocarriles/ (25 de Junio del 2019)

Documento 2:
PRESUPUESTO

En este documento se llevará a cabo el estudio económico presupuestario del proyecto, para el cual se calcularán los presupuestos parciales y, finalmente, el presupuesto final.

Para su análisis se distinguirán las siguientes actividades: Programación, Simulación y Redacción.

1. PROGRAMACIÓN

Consiste en la implementación de los algoritmos desarrollados tanto en Matlab como en Robot C para su posterior simulación en la vida real con el robot NXT.

Nº Actividad	Código	Descripción de las unidades de obra	Rendimiento	Precio (€)	Importe (€)
01	A001	h Implementación de los algoritmos de Campos Potenciales Artificiales y Generación de Clotoides.			
	MO01	h Ingeniero	1	40,50	40,50
		Costes directos			40,50
		Costes Directos Complementarios (1%)			0,40
		Coste Total			**40,90**

Tabla 2 Costes Programación.

El precio por hora ha sido obtenido de la Encuesta de Salarios y Actividad profesional de los COLEGIOS OFICIALES DE INGENIEROS INDUSTRIALES DE ALAVA, BIZKAIA, GIPUZKOA Y NAVARRA (la mejor disponible con la información buscada), que establece este importe para el caso de cuando el Ingeniero trabaja por cuenta propia, y factura por horas, con tareas propias de una Oficina Técnica[1] en 63,30 euros. De ahí se ha detraído el 21% de IVA, el 6% de beneficio industrial, y el 13% de gastos generales que se imputarán ya al final del cálculo del presupuesto (63,30-21%=50,00; 50,00-19%=40,5 euros).

[1] http://www.ingeniariak.eus/wp-content/uploads/2017/04/Encuesta-salarios-Ingenieros-Industriales-2016-2017.pdf

2. SIMULACIÓN

Consiste en la simulación del algoritmo desarrollado mediante el robot NXT además de la simulación en la herramienta Simulink.

Nº Actividad	Código	Descripción de las unidades de obra	Rendimiento	Precio (€)	Importe (€)
02	A002	h Simulación de los algoritmos de Campos Potenciales Artificiales y Generación de Clotoides mediante Simulink y el robot NXT.			
	MO01	h Ingeniero	1	40,50	40,50
		Costes Directos			40,50
		Costes Directos Complementarios (1%)			0,40
		Coste Total			**40,90**

Tabla 3 Costes Simulación.

3. REDACCIÓN

Consiste en la redacción de la memoria del Trabajo Fin de Grado

Nº Actividad	Código	Descripción de las unidades de obra	Rendimiento	Precio (€)	Importe (€)
03	A003	h Redacción del Trabajo Fin de Grado.			
	MO01	h Ingeniero.	1	40,50	40,50
		Costes Directos.			40,50
		Costes Directos Complementarios (1%)			0,40
		Coste Total.			**40,90**

Tabla 4. Costes Redacción.

4. PRESUPUESTO PARCIAL

Nº Actividad	Descripción de las unidades de obra	Medición	Precio (€)	Importe (€)
01	h Implementación de los algoritmos de Campos Potenciales Artificiales y Generación de Clotoides.	150	40,90	**6.135,00**
02	h Simulación de los algoritmos de Campos Potenciales Artificiales y Generación de Clotoides mediante Simulink y el robot NXT.	30	40,90	**1.227,00**
03	h Redacción del Trabajo Fin de Grado.	120	40,90	**4.908,00**
04	Ud. MATLAB	1	800,00	**800,00**
05	Ud. Robot C	1	263,24	**263,24**
06	Ud. Robot NXT	1	633,00	**633,00**
07	Ud. Ordenador Portátil Lenovo G500	1	595,00	**595,00**

Tabla 5 Presupuesto Parcial.

5. PRESUPUESTO BASE

Descripción	Importe
01 Programación	6.135,00
02 Simulación	1.227,00
03 Redacción	4.908,00
04 MATLAB	800,00
05 Robot C	263,24
06 Robot NXT	633,00
07 Ordenador Portátil	595,00
Presupuesto de Ejecución	**14.561,24€**
Gastos Generales 13%	1.892,96
Beneficio Industrial 6%	873,67
Presupuesto de Ejecución por Contrata	**17.327,87€**
I.V.A. 21%	3.638,85
Presupuesto base de licitación	**20.966,72€**

Tabla 6 Presupuesto Total.

Quedando un presupuesto de **VEINTE MIL NOVECIENTOS SESENTA Y SEIS EUROS CON SETENTA Y DOS CÉNTIMOS**