Universidad Zaragoza
1542

Trabajo Fin de Grado

Control de trayectorias de un robot móvil todo terreno con ruedas

Autor

Luis Benages Pardo

Director

Luis Montano Gella

Escuela de Ingeniería y Arquitectura
Zaragoza, 2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. LUIS BENAGES PARDO,

con nº de DNI 73105028-B en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO, (Título del Trabajo) CONTROL DE TRAYECTORIAS DE UN ROBOT MÓVIL TODO TERRENO CON RUEDAS,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, a 15 de Noviembre de 2017

Fdo: LUIS BENAGES PARDO

RESUMEN

CONTROL DE TRAYECTORIAS DE UN ROBOT MÓVIL TODO TERRENO CON RUEDAS

La navegación autónoma de robots móviles sin necesidad de intervención externa mediante la toma de decisiones con poca o nula supervisión humana, es una de las principales áreas de investigación en robótica hoy en día. Para llevarlo a cabo, el robot hace uso tanto de la obtención de su localización, como de la información captada por los sensores embarcados en él para poder seguir una trayectoria deseada, o desempeñar unos objetivos marcados.

En este trabajo de fin de grado se va a diseñar e implementar la generación de trayectorias para robots "tipo coche" utilizando diferentes técnicas aplicadas en el robot Robucar-TT del grupo de Robótica de la Universidad de Zaragoza. La primera de estas técnicas consiste en el cálculo y seguimiento de la trayectoria óptima, siendo ésta la más corta a una localización deseada, en un espacio libre de obstáculos mediante la aplicación de las relaciones geométricas de las trayectorias de Dubins. La segunda técnica implementada es la ejecución de un cambio de sentido en un espacio donde el robot no es capaz de llevarlo a cabo sin realizar maniobra alguna, como sería la situación de un túnel estrecho, de forma que lo realice de la manera más rápida posible quedándose alineado con la pared, mediante la ejecución de las maniobras necesarias y utilizando para ello los sensores de rango láser que dispone.

Para conseguir estos objetivos, en primer lugar se han realizado simulaciones haciendo uso de la plataforma *ROS*, para la integración del algoritmo en un nodo, del simulador *Gazebo* y del visualizador RViz para poder comprobar el correcto funcionamiento de las técnicas de navegación.

Una vez realizadas las simulaciones y comprobado que funcionen correctamente, se han realizado las correspondientes pruebas de campo en el robot real para el análisis de la precisión del seguimiento de las trayectorias, y la influencia de la dinámica en ella, utilizando como referencia la información de localización proporcionada por la odometría del robot.

Índice general

Capítulo 1. Introducción 1
1.1. Contexto y estado del arte 1
1.2. Objetivos 3
1.3. Organización de la memoria 5

Capítulo 2. Cálculo y seguimiento de las trayectorias de Dubins 6
2.1. Cálculo de la localización de los puntos de tangencia 7
2.1.1. Cálculo en trayectorias LSL y RSR 9
2.1.2. Cálculo en trayectorias *LSR* y *RSL* 12
2.2. Elección de la trayectoria 13
2.3. Seguimiento de la trayectoria 15
2.3.1. Modelo cinemático del robot 15
2.3.2. Trayectorias circulares 16
2.3.3. Trayectoria rectilínea 17

Capítulo 3. Cambio de sentido realizando maniobras 21
3.1. Cálculo de la orientación de la pared 23
3.1.1. Tratamiento de datos del láser con filtro de mediana 24
3.1.2. Elección del rango de puntos que definen la recta 25
3.1.3. Cálculo de la orientación de la pared 27
3.2. Estimación del movimiento final 28
3.2.1. Cálculo de la distancia más corta entre centro de rotación y la pared 28
3.2.2. Condición para iniciar la trayectoria final 29

Capítulo 4. Resultados 31
4.1. Simulación 31
4.1.1. Trayectorias de Dubins 31
4.1.2. Cambio de sentido realizando maniobras 34
4.2. Pruebas de campo 36
4.2.1. Trayectorias de Dubins 36
4.2.2. Cambio de sentido realizando maniobras 40

Capítulo 5. Conclusiones 46

Bibliografía 48

Anexo A. Conjunto de pruebas realizadas 49

Trayectorias de Dubins 49

Simulación 49

Pruebas de campo 52

Anexo B. Descripción software 54

Representación de gráficos de comunicación entre nodos 55

Índice de figuras

Figura 1.1: Robots autónomos terrestres de trabajo **1**
Figura 1.2: Vehículo autónomo de Google **2**
Figura 1.3: Robucar-TT de la Universidad de Zaragoza **2**
Figura 1.4: Ejemplos de trayectorias de Dubins CSC **4**
Figura 1.5: Maniobra dentro de un túnel **5**

Figura 2.1: Diferencia entre trayectorias de Dubins CSC y CCC **6**
Figura 2.2: Transformación de coordenadas globales a coordenadas relativas a la localización inicial. R representa rotación y T traslación. **7**
Figura 2.3: Composición de transformaciones. Transformación relativa ${}^{R1}T_{R2}$ **8**
Figura 2.4: Localización relativa **9**
Figura 2.5: Relaciones geométricas LSL **10**
Figura 2.6: Relaciones geométricas LSR **12**
Figura 2.7: Modelo cinemático del robot **16**
Figura 2.8: Ejemplo de trayectoria a controlar **17**
Figura 2.9: Control de trayectoria **18**
Figura 2.10: Diagrama de bloques del control en Bucle Cerrado **19**

Figura 3.1: Ejemplo de maniobras a realizar para cambio de sentido **22**
Figura 3.2: Sensores láser frontales del Robucar-TT **23**
Figura 3.3: Medidas reales de sensor láser con outliers **24**
Figura 3.4: Ejemplo filtro de mediana **25**
Figura 3.5: Elección de puntos para calcular la orientación de la pared **26**
Figura 3.6: Ángulo e índice de los puntos característicos del láser **27**
Figura 3.7: Trayectorias descritas durante la ejecución de la maniobra que concluye el cambio de sentido **29**

Figura 4.1: Escenario en simulador Gazebo para trayectorias de Dubins. **32**
Figura 4.2: Trayectorias posibles de Dubins generadas. **33**
Figura 4.3: Simulación de trayectoria de Dubins seguida. **33**
Figura 4.4: Escenario en simulador Gazebo para maniobra de cambio de sentido. **34**
Figura 4.5: Situación inicial antes de comenzar la maniobra para cambio de sentido. **35**

Figura 4.6: Situación final una vez concluido el cambio de sentido **36**
Figura 4.7: Trayectorias de Dubins seguidas, visualizadas en RViz **37**
Figura 4.8: Trayectorias de Dubins seguidas, visualizándolas con Matlab.... ..... **38**
Figura 4.9: Detalle de las trayectorias de Dubins seguidas, visualizadas con Matlab (ampliada)..... **38**
Figura 4.10: Gráficas de velocidad lineal, y ángulo de giro de las ruedas direccionales. **39**
Figura 4.11: Escenario real para el cambio de sentido..... **40**
Figura 4.12: Trayectoria seguida en cambio de sentido con una pared..... **41**
Figura 4.13: Velocidad y ángulo de las ruedas direccionales con una pared.. ..... **41**
Figura 4.14: Trayectoria seguida en cambio de sentido con dos paredes ... **42**
Figura 4.15: Velocidad y ángulo de las ruedas direccionales con dos paredes **43**
Figura 4.16: Robucar-TT dentro del túnel de Somport..... **44**
Figura 4.17: Trayectoria seguida en cambio de sentido en túnel de Somport **44**
Figura 4.18: Velocidad y ángulo de las ruedas direccionales en túnel de Somport **45**

Figura A.1: Trayectoria de Dubins RSL hasta localización (0, -7, 0) **49**
Figura A.2: Trayectoria de Dubins LSL hasta localización (-10, 3, 135) ... **50**
Figura A.3: Trayectoria de Dubins LSR hasta localización (-15, 7, 90) **50**
Figura A.4: Trayectoria de Dubins LSR hasta localización (10, 3, -90) **51**
Figura A.5: Trayectoria de Dubins RSR hasta localización (-10, -1, 90) **51**
Figura A.6: Trayectoria de Dubins LSR hasta localización (10, -6, 180) ... **52**
Figura A.7: Trayectoria de Dubins LSR hasta localización (-15, 7, 90) **53**
Figura A.8: Trayectoria de Dubins LSR hasta localización (0, -15, 0) **53**

Figura B.1: Gráfico de comunicación de nodos ROS con Dubins_node **55**
Figura B.2: Gráfico de comunicación de nodos ROS con Maniobra_tunel_node **56**

Capítulo 1

Introducción

1.1. Contexto y estado del arte

En la actualidad, el uso de robots autónomos terrestres se está popularizando en múltiples aplicaciones, ya sea en el ámbito de la investigación, por ejemplo en la realización de operaciones en terrenos hostiles o perjudiciales para la salud humana, o en tareas de trabajo como podrían ser la agricultura o la logística dentro de un almacén, con el fin de facilitar o sustituir la labor de las personas, o realizar tareas inaccesibles para las mismas. Los ejemplos de su uso en agricultura y logística quedan reflejados en la Figura 1.1 mediante dos aplicaciones reales de los mismos.

(a) Tractor autónomo

(a) Robot logístico

Figura 1.1: Robots autónomos terrestres de trabajo

Otra clara área de investigación que estamos viviendo hoy en día, es el desarrollo de automóviles autónomos que sustituyan a las personas en las tareas de conducción. Algunas empresas ya han desarrollado algunos prototipos, como el expuesto la Figura 1.2, pero todavía no se ha conseguido un modelo que asegure su correcta navegación autónoma sin necesidad de supervisión humana, por lo que todavía queda una gran trayectoria en esta línea de investigación.

Figura 1.2: Vehículo autónomo de Google

El robot Robucar-TT que se ha utilizado en el presente trabajo, mostrado en la Figura 1.3, es un robot fabricado por la empresa francesa Robosoft que ha sido modificado por el grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza mediante la incorporación de nuevos elementos extras, como más sensores de rango laser, o una cámara kinect. Se trata de un robot terrestre con tracción a las cuatro ruedas, que dispone de tres posibles modos de movimiento distintos. El primero es el llamado modo coche, donde la orientación de las ruedas traseras queda fijada paralela a los laterales del robot, y se establecen únicamente las ruedas delanteras como direccionales. En el segundo modo, de cuatro ruedas, todas ellas se mueven para permitir radios de giro menores mediante el cambio de orientación de las ruedas traseras en sentido contrario al de las ruedas delanteras. El último modo es el modo cangrejo, donde todas las ruedas giran en la misma dirección para permitir desplazamientos en diagonal. Debido a las características de los principales objetivos de este TFG, y siguiendo con la investigación de otros proyectos de fin de carrera, solo se contemplará el caso en el que el robot avanza en modo coche.

Figura 1.3: Robucar-TT de la Universidad de Zaragoza

Los vehículos tipo coche presentan unas restricciones cinemáticas que limitan su capacidad de maniobra; por ejemplo no pueden girar sobre sí mismos como los robots de tracción diferencial. Ello hace que la generación de trayectorias tiene que tener en cuenta estas restricciones a la hora de planificar trayectorias óptimas, como son las trayectorias de camino más corto entre dos configuraciones (posición más orientación). En este proyecto se implementa la técnica de trayectorias de Dubins que tiene en cuenta estas restricciones cinemáticas.

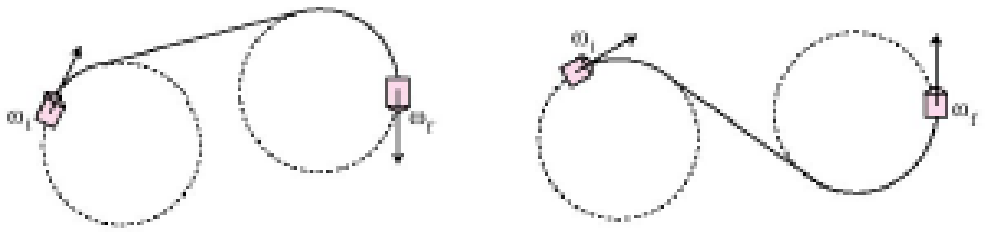
Estas trayectorias están compuestas por tramos rectilíneos y circulares concatenados, que combinados adecuadamente proporcionan las trayectorias óptimas entre configuraciones. El robot Robucar-TT con el que se va a trabajar no tiene implementadas este tipo de trayectorias, que es lo que se desarrolla en este Trabajo Fin de Grado. La programación se realizará en ROS (Robot Operating System), sistema operativo para robótica estandarizado actualmente, que permite la simulación y la implementación posterior en robots reales.

1.2. Objetivos

Para llevar a cabo los objetivos abordados en éste trabajo, se ha realizado la implementación de los mismos en un nodo de la plataforma de programación *ROS* [3], utilizando como lenguaje C++. También se ha utilizado el entorno de simulación *Gazebo* [4], que permite la simulación de robots y escenarios 3D para comprobar y corregir el funcionamiento de los algoritmos diseñados para el cumplimiento de dichos objetivos. Para poder verificar que se han seguido correctamente las trayectorias calculadas, u observar de forma visual la información obtenida por los sensores, se ha empleado el visualizador gráfico 3D *RViz* [5], el cual presenta estas aplicaciones tanto en el entorno de simulación, como en las pruebas de campo con el robot real. La explicación detallada del funcionamiento de éste software, se ha reflejado en el Anexo B de éste proyecto.

El primer objetivo de este proyecto es conseguir implementar el cálculo y generación de trayectorias de Dubins por las que un robot tipo coche consigue llegar a una posición y orientación (localización) finales deseadas de manera que siga el camino más corto, independientemente de cual sea su situación inicial, suponiendo que no hay obstáculos en el terreno. Asume también que el robot es únicamente capaz de movimiento hacia adelante. Este tipo de movimientos, sin considerar la marcha hacia atrás tiene sentido en el caso, muy habitual, de que el robot lleve embarcados sensores únicamente en la parte frontal, siendo por tanto seguros solamente los movimientos hacia adelante.

El robot deberá ser capaz de decidir de forma autónoma cuál de los cuatro tipos de trayectorias de Dubins *CSC* (Circular-Rectilíneo-Circular) [1] [2], como las representadas en la Figura 1.4, es la más corta para llegar a la localización objetivo, de modo que la única intervención humana que haya que realizar sea la de indicarle inicialmente cual es esa localización final. El cálculo y seguimiento de estas trayectorias consistirá en hallar las relaciones geométricas entre la posición final relativa a la inicial, que nos darán la posición exacta de los puntos de tangencia entre las dos trayectorias circulares, y la recta que une ambas.



(a) Trayectoria *RSR* (b) Trayectoria *RSL*

Figura 1.4: Ejemplos de trayectorias de Dubins CSC

El segundo objetivo es el de conseguir que el robot realice las mínimas maniobras posibles para llevar a cabo un cambio de sentido dentro de un túnel, o en un entorno donde el robot no puede girar sin tener que realizar ninguna maniobra, de forma similar a la mostrada en la Figura 1.5. En este caso, se presentan dos principales problemas, siendo el primero de ellos el conocer la orientación inicial de la pared respecto del robot con la que se podrá obtener el sentido del giro inicial para que la trayectoria sea más corta, y la orientación con que debe detenerse una vez realizado el cambio de sentido. El segundo problema será saber el momento en el que el robot ya puede finalizar el cambio de sentido sin necesidad de volver a realizar una nueva maniobra, en el momento en el que está moviéndose hacia atrás. Para poder resolver ambos problemas se hará uso de sus sensores de rango laser mediante los que el robot puede calcular la distancia del sensor/robot a la pared en cualquiera de sus puntos.

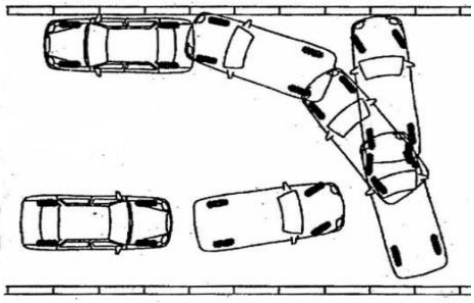


Figura 1.5: Maniobra dentro de un túnel

1.3. Organización de la memoria

El resto del trabajo se ha dividido en los siguientes capítulos:

- **Capítulo 2:** En este capítulo se explica la forma en la que se calculan las relaciones geométricas de las trayectorias de Dubins para su posible implementación en una aplicación real, siendo una de ellas el camino más corto entre la localización inicial de un robot "tipo coche" y la localización final que se pretende alcanzar. Al mismo tiempo se explica la manera en la que se realiza el seguimiento de la trayectoria calculada, mediante el envío de consignas de velocidad lineal y de ángulo de giro de las ruedas delanteras del robot.

- **Capítulo 3:** En el que se estudia la forma en la que un robot "tipo coche" pueda realizar un cambio de sentido de la manera más rápida posible en algún escenario en el que necesite realizar una o varias maniobras para conseguirlo, como podría serlo en la situación en la que el robot se encuentre dentro de un túnel en el que el radio giro mínimo que puede realizar el robot es mayor que el ancho del túnel. Para ello se hace uso de sensores de rango láser para saber la posición y orientación de la pared respecto del robot.

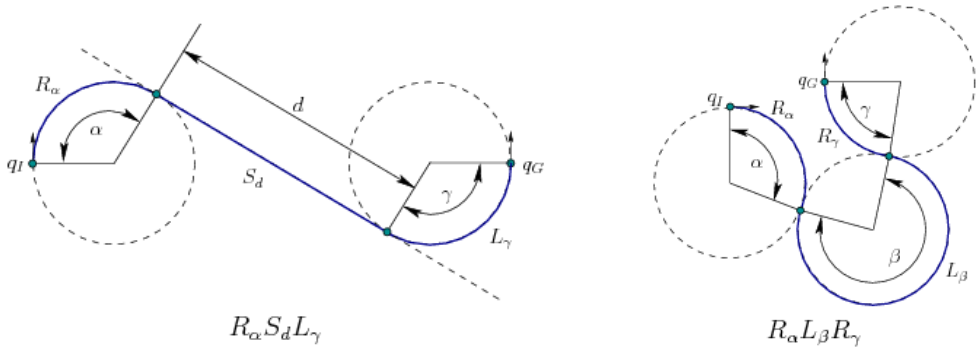
- **Capítulo 4:** Donde se analizan tanto los resultados obtenidos en simulación, como los obtenidos en los experimentos con el robot real, realizando las comparaciones necesarias entre ambos.

- **Capítulo 5:** En el cual se desarrollan las conclusiones del proyecto y las posibles ampliaciones del mismo.

Capítulo 2

Cálculo y seguimiento de las trayectorias de Dubins

Las trayectorias de Dubins [1], [2] consisten en la obtención del camino óptimo de un robot tipo coche u otro con capacidad de movimiento similar, siendo éste el más corto desde una localización inicial a una localización final deseada. Estas trayectorias están constituidas por tres segmentos que forman la trayectoria completa, pudiendo ser éstos de dos tipos: tramos con movimiento circular denominados como *C*, o tramos de movimiento rectilíneo asignados como *S*. Así entonces, se establece que la trayectoria más corta entre dos localizaciones queda definida por seis posibles trayectorias: cuatro *CSC* y dos *CCC* [1].



(a) Trayectoria *RSL* (b) Trayectoria *RLR*

Figura 2.1: Diferencia entre trayectorias de Dubins CSC y CCC

Sin embargo, las trayectorias *CCC* están diseñadas, para robots que solo puedan avanzar hacia adelante, en el caso de que alguna de las trayectorias *CSC* no pueda ser calculada debido a la proximidad entre la localización inicial y final, y puesto que en nuestro caso el robot también puede avanzar marcha atrás, las trayectorias *CCC* no supondrían una trayectoria óptima por lo que no se aplicarán en éste trabajo. De este modo, en caso de que no se pueda calcular alguna de las trayectorias *CSC*, el robot realizará una maniobra en línea recta hasta que las 4 posibles trayectorias puedan ser calculadas y así comprobar cuál es la mejor opción posible. No obstante, se trata de una solución no óptima, por lo que una posible ampliación a este

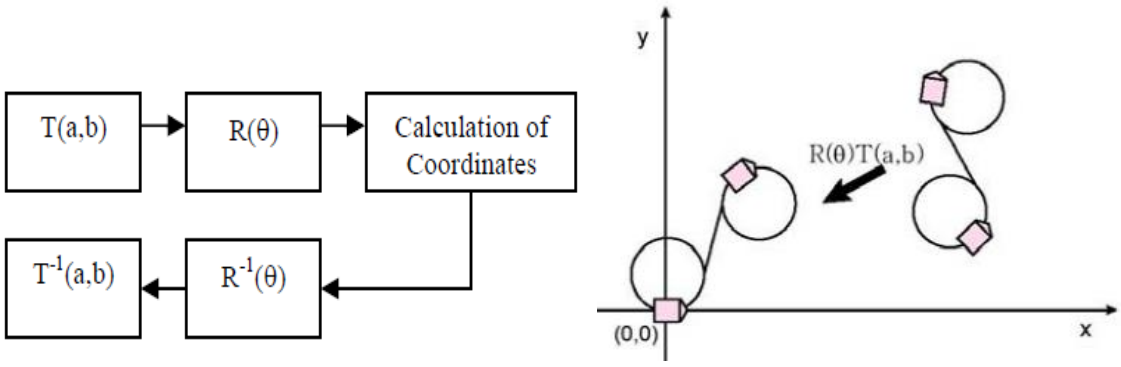
trabajo sería la aplicación de otras técnicas de generación de trayectorias para tratar estas posibles situaciones.

Las posibles trayectorias *CSC* a calcular estarán descritas por un primer tramo con movimiento circular *C*, con el menor radio de giro posible r_{MIN}, es decir máxima curvatura, seguido de un tramo recto *S*, y otro tramo circular *C* al final con el mismo radio r_{MIN}.

Los tramos circulares pueden ser de giro a izquierdas *L* o de giro a derechas *R*, de modo que las cuatro posibles trayectorias *CSC* serán: *LSL*, *LSR*, *RSR*, y *RSL*. El intervalo con movimiento rectilíneo es tangente a ambos intervalos circulares, lo que presenta que el principal problema del cálculo de este tipo de trayectorias sea el cálculo de los puntos de tangencia entre la recta y las dos circunferencias para así poder obtener el ángulo a girar en ambos tramos. Éstos resultan necesarios en caso de querer realizar una aplicación real de este tipo de trayectorias. Dichos ángulos se pueden encontrar representados en la Figura 2.1 (a) como α y γ.

2.1. Cálculo de la localización de los puntos de tangencia

Para el cálculo de las trayectorias, el primer paso que se tiene que llevar a cabo es la transformación de coordenadas globales en las que está referenciada la información de posición que lee el robot, a coordenadas relativas a la localización inicial para así poder realizar el seguimiento de la trayectoria conociendo en todo momento la localización del robot respecto de la posición inicial donde empezó dicha trayectoria.



(a) Diagrama de bloques (b) Ejemplo *LSR*

Figura 2.2: Transformación de coordenadas globales a coordenadas relativas a la localización inicial. R representa rotación y T traslación.

La Figura 2.2 representa estas transformaciones (T, R) para conseguir que una posición inicial arbitraria pase a ser el origen al establecerla como referencia, y la localización final quede en coordenadas relativas a esa nueva referencia. Como se muestra en el diagrama de bloques, una vez realizadas las operaciones necesarias para el cálculo de la trayectoria, si se desea se puede volver a obtener todos los puntos calculados en coordenadas globales mediante las transformaciones inversas (R^{-1} y T^{-1}).

El cálculo de estas localizaciones relativas se lleva a cabo mediante composición de transformaciones homogéneas ($^{R1}T_{R2}$), tal y como se representa en la Figura 2.3, de forma que se puede obtener la transformación relativa entre dos localizaciones R2 y R1.

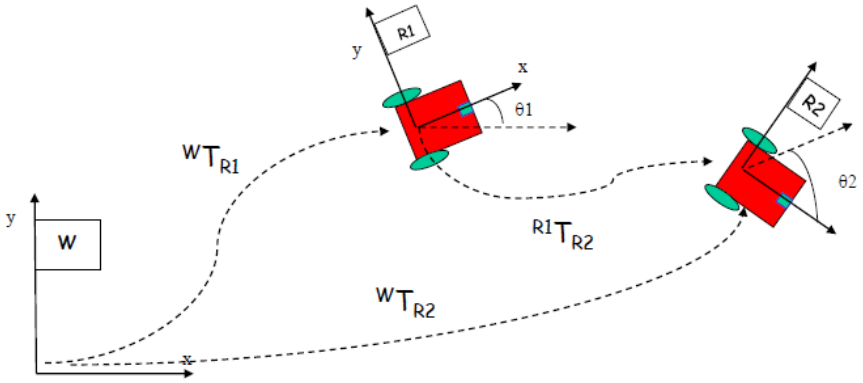


Figura 2.3: Composición de transformaciones. Transformación relativa $^{R1}T_{R2}$

$$^{W}T_{R2} = {}^{W}T_{R1} * {}^{R1}T_{R2} \Rightarrow {}^{R1}T_{R2} = ({}^{W}T_{R1})^{-1} * {}^{W}T_{R2} \quad (2.1)$$

$$^{R1}T_{R2} = \begin{pmatrix} n_x & o_x & p_x \\ n_y & o_y & p_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R & p \\ 0 & 1 \end{pmatrix} \quad (2.2)$$

Donde R representa la matriz de rotación y p el vector de posición. La representación de una localización relativa a otra se puede realizar mediante una transformación homogénea ($^{R1}T_{R2}$), pero también como un vector localización $^{R1}X_{R2}$ que nos indica la posición (p) y orientación (θ) de la localización *R2* relativa a la de *R1*, tal y como se muestra en la Figura 2.4. Las transformaciones homogéneas simplifican notablemente la composición de las mismas frente al uso de los vectores de localización, que representan directamente la posición y la orientación.

La forma de convertir una transformación a vector localización y viceversa, queda reflejada en las siguientes expresiones:

$$ {}^{R1}T_{R2} = \begin{pmatrix} n_x & o_x & p_x \\ n_y & o_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \longrightarrow {}^{R1}X_{R2} = (\, p_x,\ p_y,\ atan2(n_y, n_x)\,) \tag{2.3} $$

$$ {}^{R1}X_{R2} = (x, y, \theta) \longrightarrow {}^{R1}T_{R2} = \begin{pmatrix} \cos(\theta) & -sen(\theta) & x \\ sen(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4} $$

Sin embargo en este trabajo se han implementado las transformaciones explicadas mediante el uso de la clase *tf::Transform* de las librerías de *ROS* [7] puesto que permite realizar composición de transformaciones mediante funciones de la librería sin necesidad de operar externamente con matrices homogéneas. Esta clase también proporciona la posibilidad de extraer o introducir, en una transformación, la rotación entre localizaciones mediante los giros RPY (Roll-Pitch-Yaw en el caso 3D). Teniendo en cuenta que el escenario sobre el que va a navegar el robot es 2D, el ángulo *Yaw* se corresponderá directamente con θ.

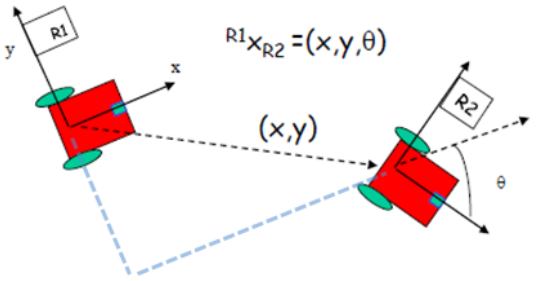


Figura 2.4: Localización relativa

De esta manera, una vez conocida la localización final deseada respecto de la inicial, ${}^{R_i}X_{R_f}$, y se haya realizado la transformación a coordenadas relativas de la localización del robot, ya es posible realizar el cálculo y seguimiento de las trayectorias de Dubins, mediante la obtención de las relaciones geométricas entre los distintos puntos de la trayectoria. El cálculo de estas relaciones es diferente en el caso de que se trate de una trayectoria *LSL* o *RSR*, que en el caso de que se corresponda con una trayectoria *LSR*, o RSL.

2.1.1. Cálculo en trayectorias LSL y RSR

Las relaciones geométricas de éste tipo de trayectorias aparecen reflejadas en la Figura 2.5 a partir de la cual se van a explicar los pasos necesarios para el cálculo de las coordenadas de los puntos de tangencia, y su orientación respecto la posición inicial del robot.

Para diferenciar entre la forma de calcular una trayectoria LSL o una RSR se va introducir un parámetro ***k*** que representará un cambio de signo en algunas de las ecuaciones citadas a continuación, ya que es la única diferencia entre ambos casos, del siguiente modo:

$$k = \begin{cases} 1 & si \quad LSL \\ -1 & si \quad RSR \end{cases} \tag{2.5}$$

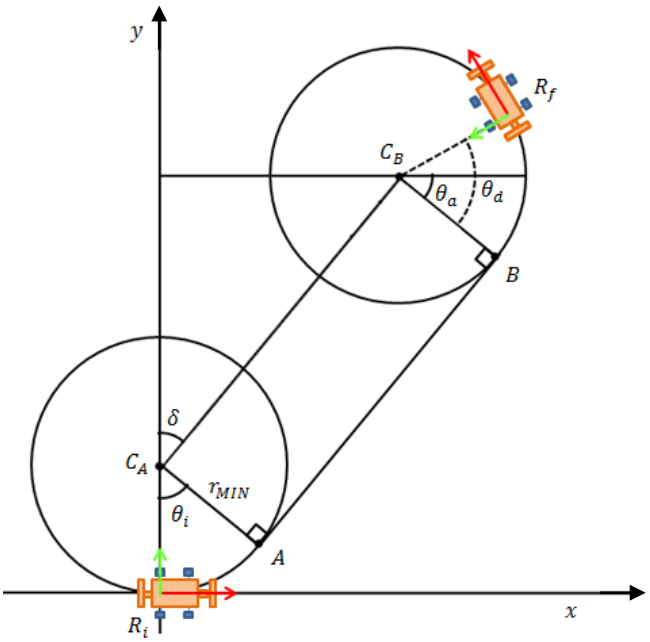


Figura 2.5: Relaciones geométricas LSL

El primer paso necesario, es conocer la posición relativa entre los centros de rotación de los giros inicial y final de la trayectoria, cuya notación se ha designado como C_A y C_B respectivamente, de modo que quede definida mediante el vector localización ${}^{C_A}X_{C_B}$. Los cálculos se van a realizar relativos a la referencia de la localización de origen, estando el robot siempre orientado en esta referencia con su eje longitudinal en el eje x de la referencia.

El cálculo de la posición de C_A es trivial puesto que solo tiene componente en eje y, debido a que las coordenadas son relativas a la posición inicial R_i de la trayectoria, siendo el valor de esta componente el radio de giro r_{MIN}. Puesto que C_A es un punto y no tiene orientación, se asigna a su vector localización un cero lo que significa que tendrá la misma orientación que la referencia base.

$${}^{R_i}X_{C_A} = (0, k * r_{MIN}, 0) \overset{[7]}{\Rightarrow} {}^{R_i}T_{C_A} \tag{2.6}$$

En el caso del cálculo de las coordenadas de C_B respecto de R_i, el artículo [1] hace uso de las relaciones geométricas entre la localización objetivo R_f y el centro del giro final C_B. Sin embargo, en el presente trabajo se ha optado por calcular las coordenadas de C_B mediante el uso de transformaciones, debido a su mayor facilidad de comprensión e implementación en todos los casos posibles en cada tipo de trayectoria. De este modo, conociendo el vector localización ${}^{R_i}X_{R_f}$ se obtienen las transformaciones:

$$ {}^{R_i}X_{R_f} \Rightarrow {}^{R_i}T_{R_f} \tag{2.7} $$

$$ {}^{R_f}X_{C_B} = (0, k * r_{MIN}, 0) \Rightarrow {}^{R_f}T_{C_B} \tag{2.8} $$

$$ {}^{R_i}T_{R_f} * {}^{R_f}T_{C_B} = {}^{R_i}T_{C_B} \tag{2.9} $$

Mediante la localización obtenida (2.9) y ${}^{R_i}T_{C_A}$ ya es posible calcular la posición relativa entre ambos centros de rotación mediante composición de transformaciones. Una vez se tienen las componentes del vector localización calculado, se les asigna una notación para su uso posterior:

$$ ({}^{R_i}T_{C_A})^{-1} * {}^{R_i}T_{C_B} = {}^{C_A}T_{C_B} \Rightarrow {}^{C_A}X_{C_B} = (x_{CC}, y_{CC}, \theta_{CC}) \tag{2.10} $$

El siguiente paso a realizar es el cálculo del ángulo δ a partir del cual es posible obtener el resto de ángulos necesarios para establecer las relaciones trigonométricas que permiten conseguir las localizaciones de los puntos de tangencia, *A y B*, entre la trayectoria rectilínea y las trayectorias circulares inicial y final. Es importante realizar dicho cálculo mediante la arcotangente2, y no la arcotangente, puesto que es necesario distinguir entre los distintos cuadrantes, en función de en cual se encuentre la localización objetivo.

$$ \delta = k * atan2(x_{CC}, y_{CC}) \tag{2.11} $$

También es importante destacar que en operaciones de suma o multiplicación de ángulos, el resultado debe estar normalizado entre $[-\pi, \pi]$.

$$ \theta_i = \begin{cases} \pi - \left(\delta + \frac{\pi}{2}\right) & si \ \ LSL \\ \delta + \frac{\pi}{2} & si \ \ RSR \end{cases} \quad \in [-\pi, \pi] \tag{2.12} $$

$$ \theta_a = \delta \tag{2.13} $$

Una vez obtenidos los ángulos θ_i y θ_a, ya es posible establecer las relaciones trigonométricas que nos darán las coordenadas de los puntos de tangencia *A y B,* y cuya orientación esta descrita en ambos casos por θ_i.

$$ {}^{R_i}X_A = (k * r_{MIN} * \text{sen}(\theta_i), k * r_{MIN} * (1 - \cos(\theta_i)), \theta_i) \tag{2.14} $$

$$^{R_i}X_B = (x_{CC} + k * r_{MIN} * \cos(\theta_a), y_{CC} - r_{MIN} * \text{sen}(\theta_a), \theta_i) \quad (2.15)$$

2.1.2. Cálculo en trayectorias *LSR* y *RSL*

En este caso, las relaciones geométricas están representadas en la Figura 2.6 donde se pueden apreciar las diferencias respecto al caso anterior. Debido a esas diferencias geométricas, el cálculo de los primeros pasos para obtener el ángulo δ será distinto.

Puesto que en el caso anterior ya se ha razonado los pasos a seguir para obtener la localización de los puntos de tangencia, en este tipo de trayectorias se va a proceder a exponer directamente todos los cálculos necesarios para obtenerlas

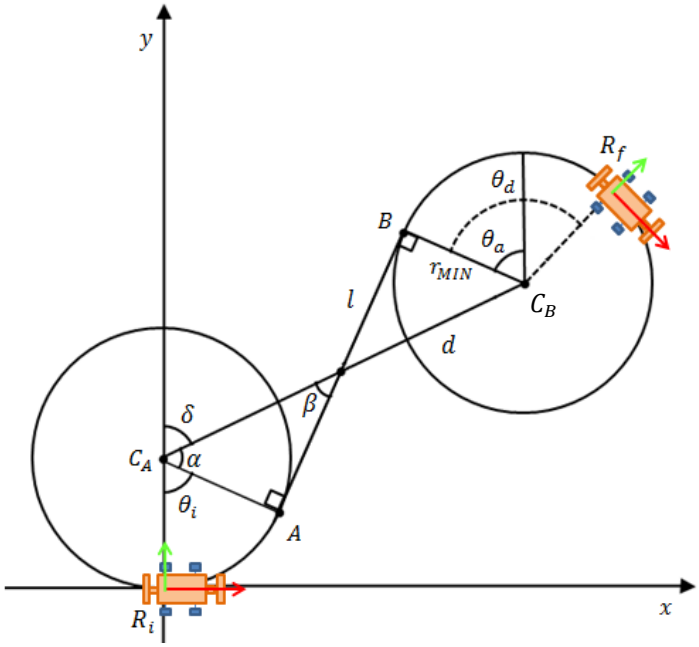


Figura 2.6: Relaciones geométricas LSR

Del mismo modo que en el caso anterior, se introduce un parámetro k para calcular una trayectoria u otra según sea el caso.

$$k = \begin{cases} 1 & si \quad LSR \\ -1 & si \quad RSL \end{cases} \quad (2.16)$$

$$^{R_i}X_{C_A} = (0, k * r_{MIN}, 0) \overset{[7]}{\Rightarrow} {}^{R_i}T_{C_A} \quad (2.17)$$

$$^{R_i}X_{R_f} \Rightarrow {}^{R_i}T_{R_f} \quad (2.18)$$

$$^{R_f}X_{C_B} = (0, -k * r_{MIN}, 0) \quad \Rightarrow \quad ^{R_f}T_{C_B} \tag{2.19}$$

$$^{R_i}T_{R_f} * {}^{R_f}T_{C_B} = {}^{R_i}T_{C_B} \tag{2.20}$$

$$(^{R_i}T_{C_A})^{-1} * {}^{R_i}T_{C_B} = {}^{C_A}T_{C_B} \quad \Rightarrow \quad {}^{C_A}X_{C_B} = (x_{CC}, y_{CC}, \theta_{CC}) \tag{2.21}$$

A la hora de calcular el ángulo θ_i será necesario unos pasos adicionales, en comparación con el caso anterior, puesto que en este tipo de trayectorias no es posible calcularlas únicamente con delta, ya que es necesario conocer el ángulo α.

$$\delta = k * atan2(x_{CC}, y_{CC}) \tag{2.22}$$

$$d = \frac{\sqrt{x_{CC}^2 + y_{CC}^2}}{2} \tag{2.23}$$

$$l = \sqrt{d^2 - r_{MIN}^2} \tag{2.24}$$

$$\beta = atan2(r_{MIN}, l) \tag{2.25}$$

$$\alpha = \left(\frac{\pi}{2} - \beta\right) \quad \in [-\pi, \pi] \tag{2.26}$$

$$\theta_i = \begin{cases} \pi - (\delta + \alpha) & si \quad LSR \\ \delta + \alpha & si \quad RSL \end{cases} \quad \in [-\pi, \pi] \tag{2.27}$$

$$\theta_a = -\theta_i \tag{2.28}$$

Conociendo el valor de los ángulos θ_i y θ_a, se obtiene la localización de los puntos de tangencia:

$$^{R_i}X_A = (k * r_{MIN} * \text{sen}(\theta_i), k * r_{MIN} * (1 - \cos(\theta_i)), \theta_i) \tag{2.29}$$

$$^{R_i}X_B = (x_{CC} + k * r_{MIN} * \text{sen}(\theta_a), y_{CC} + k * r_{MIN} * \cos(\theta_a), \theta_i) \tag{2.30}$$

2.2. Elección de la trayectoria

Uno de los pasos fundamentales a la hora de aplicar un algoritmo de cálculo y seguimiento de trayectorias, es la elección de la trayectoria óptima de entre todas las posibles para alcanzar la localización objetivo, ya que es un indicativo del grado de autosuficiencia del robot en la navegación.

En nuestro caso, trayectorias de Dubins, la trayectoria óptima será aquella que presente menor distancia a recorrer, que será a su vez la ruta más rápida entre las dos localizaciones.

Inicialmente el objetivo fue conseguir obtener la trayectoria más corta sin necesidad de realizar el cálculo de los puntos de tangencia de todas las posibles trayectorias, sino simplemente conociendo la distancia entre los 4 posibles centros de rotación, asignar los dos que tengan menor distancia entre sí. Tras realizar múltiples pruebas se concluyó que éste método no era totalmente fiable puesto que en ciertos casos especiales escogía una trayectoria que no era la óptima. Así entonces, la elección de la trayectoria de Dubins a realizar consistirá en el cálculo del perímetro de las trayectorias circulares recorridas, más la longitud del segmento que une los dos puntos de tangencia $\overline{\mathrm{AB}}$ para cada una de las 4 posibles trayectorias, de modo que se designará como óptima aquella que presente una distancia menor. En este caso se consigue la correcta elección de la trayectoria más corta, pero a costa de tener que calcular todas las posibles trayectorias.

Tomando la notación representada en las Figura 2.5 y 2.6, los ángulos recorridos en las trayectorias circulares inicial y final, con sus valores expresados entre $[0, 2\pi]$, serían θ_i y θ_d respectivamente, y la distancia del tramo rectilíneo sería $\overline{AB}$. Las coordenadas de los puntos de tangencia están expresadas en coordenadas relativas a la localización inicial R_i.

$$\partial l = r * \partial\theta \Rightarrow \begin{cases} l_{C_A} = r_{MIN} * \theta_i \\ l_{C_B} = r_{MIN} * \theta_d \end{cases} \tag{2.31}$$

$$\overline{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} \tag{2.32}$$

$$d_{TOTAL} = l_{C_A} + \overline{AB} + l_{C_B} \tag{2.33}$$

De este modo, la menor distancia total a recorrer en cada una de las trayectorias de Dubins, será la que indique cuál de ellas es el camino óptimo a seguir.

No obstante, existen algunos casos en los que no es posible el cálculo de una o más de las trayectorias debido a la proximidad entre la localización final indicada respecto de la inicial. En estos casos, tal y como se ha introducido al inicio de éste capítulo, se realizaría una maniobra corta en línea recta hacia atrás, actualizando la localización ${}^{R_i}X_{R_f}$ hasta que todas las posibles trayectorias sean calculadas y así poder elegir la más corta.

Esta solución introducida, no sería la solución óptima de realizar la maniobra para poder llegar a la localización final, sino que se trata de una simplificación para poder calcular las cuatro trayectorias posibles y así elegir las más corta. En lugar de esta simplificación, la solución óptima en

estos casos vendría dada por la aplicación de una de las trayectorias de *Reeds-Shepp Curves* [8], que consideran tanto los movimiento hacia a delante como hacia atrás, quedando como una posible ampliación para éste trabajo, puesto que su implementación conllevaría gran cantidad de trabajo que no es posible afrontar en un único proyecto.

2.3. Seguimiento de la trayectoria

Una vez se ha calculado la trayectoria teórica, el robot debe ser capaz de seguirla lo más fielmente posible para alcanzar la posición objetivo con el mínimo error admisible, y en el menor tiempo de ejecución. Para ello es necesario obtener la localización del robot en todo momento ${}^{R_i}X_R$, mediante la transformación a coordenadas relativas de la información leída por el robot a través de su odometría. Teniendo en cuenta que además de las restricciones cinemáticas mencionadas en robots tipo coche, existen restricciones dinámicas (máxima aceleración y deceleración) y errores en la localización proporcionada por los sensores del robot, es necesario realizar un control preciso de la trayectoria que se está siguiendo para minimizar los errores acumulativos.

Como ya se ha introducido anteriormente, las trayectorias CSC de Dubins se dividen en tres segmentos que describen la trayectoria completa. A continuación se va a proceder a describir la manera en que se ha implementado el seguimiento de las distintas secciones de la trayectoria, así como las características del modelo cinemático del robot que influyen en él.

2.3.1. Modelo cinemático del robot

Las variables del robot sobre las que se actúa, permitiendo el control de su movimiento, son la velocidad lineal del robot y el ángulo de giro de las ruedas direccionales. Puesto que trabajar sobre la variable del ángulo de las ruedas para realizar los radios de giro deseados presenta una relación tediosa para su asimilación, se trabaja con la velocidad lineal y angular del robot transformándolas a las variables de control, una vez se desee actuar sobre ellas, mediante las relaciones que presenta el modelo cinemático del robot [6].

$$\tan(\theta) = \frac{S}{R} \tag{2.34}$$

$$\omega = \frac{v}{R} \;\Rightarrow\; \theta = tan^{-1}(\frac{S}{v} * \omega) \tag{2.35}$$

En la Figura 2.7 se muestra el modelo cinemático del robot, donde θ es el ángulo que gira la rueda virtual del robot situada en el centro del eje de las ruedas delanteras, y que tiene el mismo centro de rotación que ambas siguiendo la geometría de Ackermann. La longitud entre los ejes de las ruedas se representa como S.

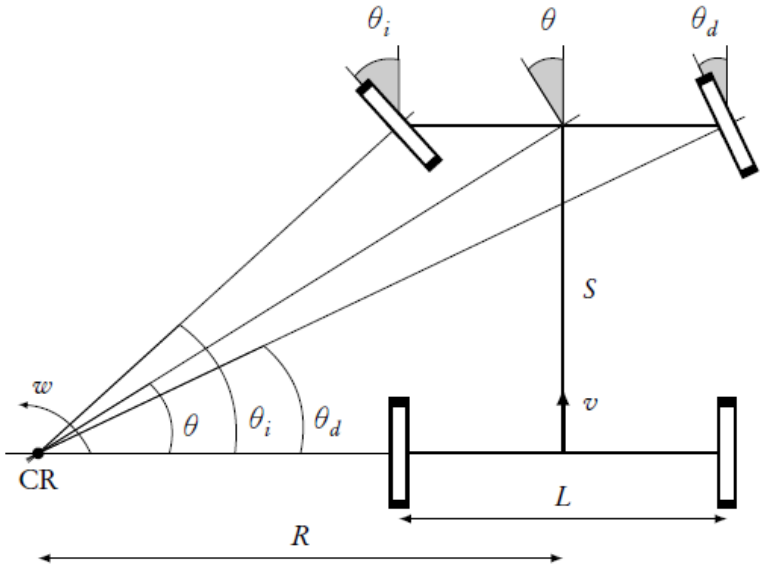


Figura 2.7: Modelo cinemático del robot

2.3.2. Trayectorias circulares

Su ejecución está basada en el cálculo del ángulo completo que deberá girar el robot, con valor entre $[0\,, 2\pi]$, para llegar desde una posición inicial P_i al último punto de la trayectoria circular P_f, con la misma orientación. Dependiendo de si se trata del giro inicial o final de la trayectoria completa, los puntos P_i y P_f se corresponderán con el origen de R_i y A, o con B y el origen de R_f, respectivamente. Del mismo modo el ángulo se corresponderá con α o γ, según la representación mostrada en la Figura 2.1 (a).

Para poder obtener los ángulos mencionados, se ha definido una función mediante la cual basta con conocer la orientación en la posición P_i, y en la posición P_f, de modo que al saber la dirección del giro que se pretende realizar, R o L, se puede obtener la orientación relativa entre ambas localizaciones, y transformarla a en un ángulo con valor entre $[0\,, 2\pi]$.

Una vez calculado el ángulo a girar, se impone una consigna de velocidad lineal y angular constantes determinados para que el robot realice una trayectoria de radio r_{MIN}. Para evitar aceleraciones bruscas, esa consigna se establece mediante una rampa lineal para el caso de la

aceleración inicial, y mediante un control proporcional al ángulo que le queda por recorrer en caso de la deceleración, ya que en este último caso la deceleración lineal no es una opción al tener que pararse el robot en un punto concreto.

Mientras tanto, se va actualizando por muestreo un vector localización ${}^{P_i}X_R$, que registra la posición y orientación del robot en cada momento en coordenadas relativas a la localización inicial del giro, mediante la información obtenida de la lectura de la odometría del robot. De este modo se puede comparar la orientación relativa del robot en cada momento, con valor entre $[0\,,2\pi]$, y el ángulo total de giro para saber si ha finalizado la trayectoria circular o no.

2.3.3. Trayectoria rectilínea

En un principio el primer razonamiento lógico sería introducir una consigna de velocidad angular nula, y mantener la velocidad lineal constante, siendo éste el caso en que el robot haya conseguido llegar al punto de tangencia A con una precisión exacta en posición y orientación, de modo que avanzaría hacia el punto de tangencia B alcanzándolo con error nulo.

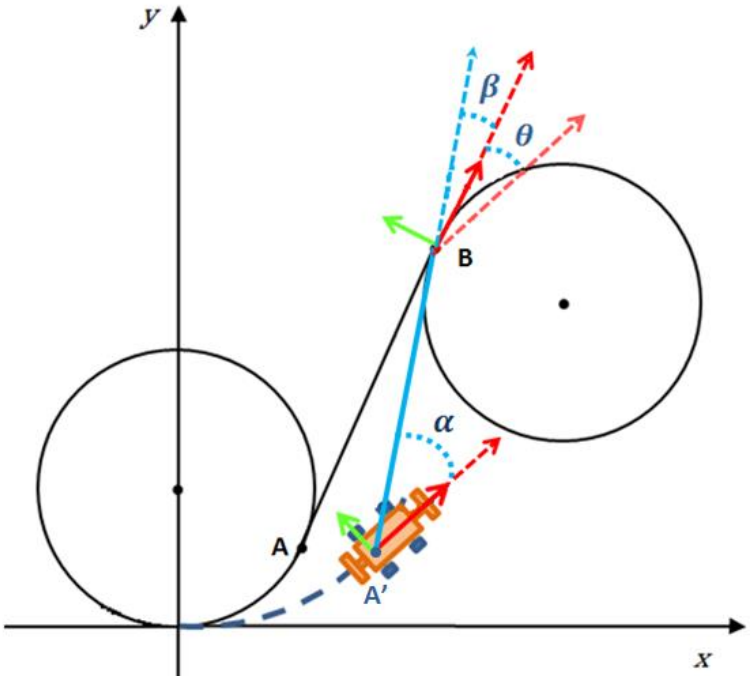


Figura 2.8: Ejemplo de trayectoria a controlar

Sin embargo, en una aplicación real no sería posible su implementación debido al error de orientación acumulado al finalizar la primera trayectoria circular, proveniente del muestreo a la hora de actualizar la localización del

robot. Además también habrá cierto error de posición derivado de múltiples factores geométricos y dinámicos, como son los parámetros geométricos (radio de ruedas, distancia entre ejes), la inercia del robot y rozamientos. Este error queda representado en el ejemplo de la Figura 2.8 (exagerándolo para una mejor visualización), donde el robot habría llegado a la localización A' en lugar de A al finalizar la primera trayectoria circular, siendo necesaria la corrección de la trayectoria para que el robot llegue a la localización B con la orientación calculada.

Por ello ha sido necesario incorporar un regulador proporcional en bucle cerrado con dos variables de control, actuando sobre la velocidad angular del robot y manteniendo la velocidad lineal constante, para ajustar la posición y la orientación simultáneamente al alcanzar una determinada localización, en este caso el punto de tangencia B. La elección de éste tipo de controlador se basa en que debido a la aplicación, en la que el camino seguido tiene que seguir fielmente al marcado, el error a corregir por el controlador en el momento en el que comienza a actuar es muy pequeño, por lo que con un regulador proporcional se consigue llegar a la consigna deseada con un error prácticamente nulo.

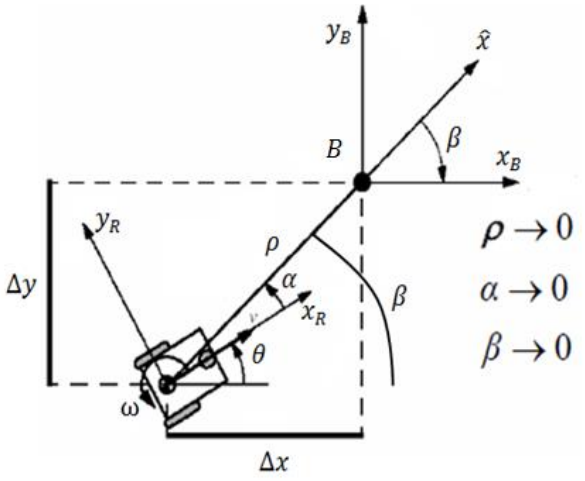


Figura 2.9: Control de trayectoria

Las variables a controlar son los dos ángulos α y β (Figura 2.8 y 2.9), siendo α el ángulo de alineamiento entre el eje x del robot y el eje ρ que une la localización objetivo, en este caso B, con la del robot, y β el ángulo de alineamiento final entre el eje x del robot y el eje x de la localización B. Para poder obtener éstos ángulos será necesario realizar una transformación de modo que se obtenga el vector localización ${}^{B}X_R$ a partir de la información de localización recibida del robot.

$$({}^{R_i}T_B)^{-1} * {}^{R_i}T_R = {}^{B}T_R \quad \Rightarrow \quad {}^{B}X_R = (\Delta x\ ,\ \Delta y\ ,\ \theta\ ) \tag{2.36}$$

En la Figura 2.8 y 2.9 se puede apreciar un ejemplo de las variables de control necesarias para llegar a la posición deseada con una orientación determinada, mediante un bucle de control para que su valor tienda a cero. La distancia ρ, que hay desde el robot hasta el punto B, no se introduce como variable de control puesto que solo influiría para actuar sobre la velocidad lineal del robot, y dado que en nuestro caso permanece constante durante toda trayectoria, el hecho de que α y β tienda a cero hará que ρ tienda a cero igualmente.

Así entonces, el cálculo de las variables de control necesarias para que el robot llegue a la localización B con el menor error posible, se realizará mediante las siguientes expresiones.

$$\beta = (atan2(\Delta y , \Delta x) + \pi) \quad \in [-\pi, \pi] \tag{2.37}$$

$$\alpha = \beta - \theta \tag{2.38}$$

A continuación se establece la ley de control del bucle cerrado quedando reflejada en la Figura 2.10, donde se puede apreciar la actuación sobre la variable de salida ω, en función de la localización del robot relativa a la del punto B que se quiere alcanzar.

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & k_\alpha & k_\beta \end{pmatrix} \begin{pmatrix} v_{cte} \\ \alpha \\ \beta \end{pmatrix} \tag{2.39}$$

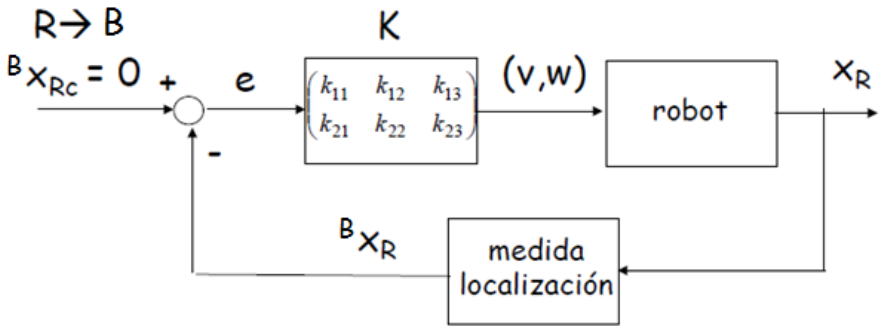


Figura 2.10: Diagrama de bloques del control en Bucle Cerrado

Para asegurar una respuesta rápida, pero sin carácter oscilante se han escogido los siguientes valores de los parámetros proporcionales K, cumpliendo con las condiciones de estabilidad.

$$Condiciones\ de\ estabilidad \begin{cases} k_\alpha > 0 \\ k_\beta > 0 \end{cases} \tag{2.40}$$

$$parámetros\ del\ regulador \begin{cases} k_\alpha = 0.55 \\ k_\beta = 1 \end{cases} \tag{2.41}$$

Capítulo 3

Cambio de sentido realizando maniobras

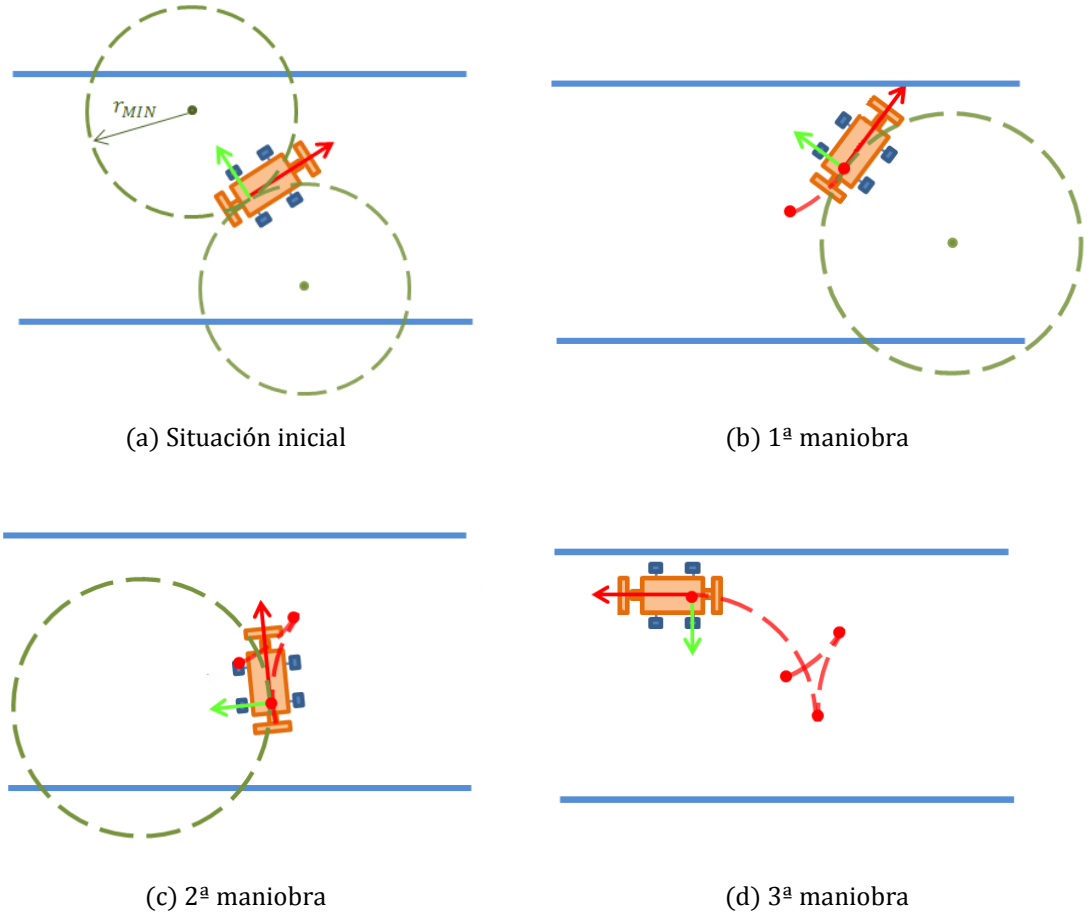
Existen múltiples escenarios en los que un robot tipo coche va a tener limitada su capacidad de movimiento en único sentido debido a las restricciones que éste tipo de robots presentan a la hora de realizar movimientos angulares, al no disponer de las mismas posibilidades de movimiento que otros tipos de robots, como pueden ser los diferenciales, donde existe la posibilidad de efectuar giros en torno al eje del robot. Estas restricciones están marcadas por la capacidad de giro de las ruedas direccionales del robot, con un ángulo máximo de giro, que definen el radio mínimo de trayectoria que este tipo de robots puede realizar.

En este proyecto se ha querido establecer un posible escenario como el representado en la Figura 3.1, en el que queden reflejadas las limitaciones citadas anteriormente, de modo que sea necesaria la realización de maniobras para la ejecución de la tarea deseada. Para ello, se ha llevado a cabo la implementación de un algoritmo que permita al robot realizar el mínimo número de maniobras para un cambio de sentido dentro de un túnel, en el que con el radio mínimo de giro del robot no sea capaz de realizarlo debido a la situación inicial de proximidad del robot respecto de las paredes (Figura 3.1(a)). Cabe a destacar que el algoritmo diseñado permite realizar esta maniobra independientemente de que se trate de un túnel, con ambas paredes a cada lado del robot, o se trate de una única pared en la que el robot tenga que cambiar de sentido tan próximo a ella como sea posible.

Para llevar a cabo la tarea, existen dos problemas fundamentales a abordar. El primero de ellos es el cálculo de la orientación de la pared respecto de la localización inicial del robot, con lo que se podrá conocer tanto la trayectoria más corta, en función de si el giro inicial es en dirección izquierda o derecha, como un ángulo de referencia con el que establecer la condición de parada para considerar completado el cambio de sentido. El segundo problema, consiste en conocer mientras se esté realizando la maniobra marcha atrás el momento en el que el robot ya tiene posibilidad de realizar la última trayectoria, para concluir el cambio de sentido, sin

necesidad de volver a realizar otra maniobra, tal como sería el caso de la Figura 3.1 (c) donde el robot ya puede llevar a cabo la última maniobra (3ª), mostrándose en verde con una línea discontinua la nueva maniobra a realizar y en rojo la trayectoria seguida en las maniobras realizadas. En el caso de que el robot en la Figura 3.1 (c) hubiera llegado a la pared trasera sin posibilidad de efectuar todavía la última maniobra, habrían sido necesarias dos maniobras adicionales para concluir el cambio de sentido.

La solución de estos problemas será llevada a cabo mediante la lectura de la información recogida por los sensores laser del robot en ambos casos.



(a) Situación inicial (b) 1ª maniobra

(c) 2ª maniobra (d) 3ª maniobra

Figura 3.1: Ejemplo de maniobras a realizar para cambio de sentido

El sensor láser embarcado en el robot es un escáner LIDAR que proporciona información de los obstáculos existentes en un área que corresponde a 180 grados de barrido frente al escáner láser. Uno de estos sensores se ubica en la parte frontal del robot y otro en la trasera por lo que se tendrá una información de los obstáculos situados frente al robot y detrás de él. En la Figura 3.2 se muestra una imagen de los sensores láser frontales del Robucar, de los cuales en ésta aplicación solo usaremos el

situado debajo ya que es el que está orientado de manera que los rayos estén situados en un plano horizontal. Del mismo modo hay otros dos sensores láser situados en la parte trasera del robot de los que usaremos también el orientado horizontalmente.

Figura 3.2: Sensores láser frontales del Robucar-TT

3.1. Cálculo de la orientación de la pared

Para el cálculo de la orientación de la pared relativa a la posición inicial del robot, el primer paso necesario es obtener la ecuación de la recta de la misma respecto de las coordenadas relativas del robot en su posición inicial. Para ello se lee la información proporcionada por el sensor de rango láser el cual nos devuelve un vector con las distancias medidas de todos los puntos del láser. De este modo, habrá que elegir el rango de puntos del láser que definen la pared, y sacar una recta de regresión de los puntos que estén dentro de ese rango.

Antes de realizar la regresión, será necesario un tratamiento previo de los datos mediante un filtro de mediana para eliminar los posibles datos espurios (*outliers*) que provocarían la obtención de una regresión errónea de los datos. En la Figura 3.3 se ha representado las medidas reales tomadas por el sensor laser frontal cuando el robot se encontraba de cara a una pared, donde se ha señalado con círculos rojos tres datos espurios que causarían que la recta de regresión no se adecuara a la medida de los puntos que se están obteniendo de la pared, en especial el *outlier* situado en la esquina superior de la imagen.

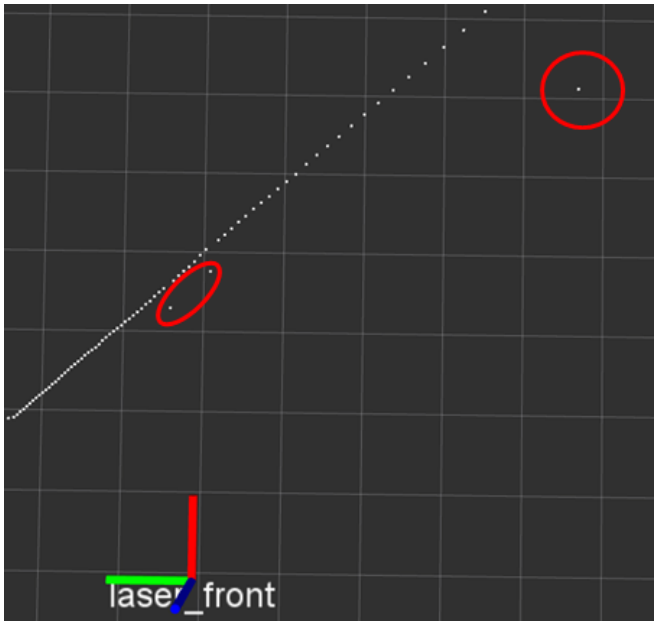


Figura 3.3: Medidas reales de sensor láser con *outliers*

Para la elección automática de los puntos extremos que definen el rango de puntos de la pared, se ha creado un algoritmo que te devuelve el índice del vector de ambos puntos, de modo que los puntos con los que se realiza la recta de regresión son aquellos que estén entre esos dos índices del vector total de puntos. Para obtener las coordenadas de cada punto del rango elegido, sabiendo que cada punto del láser está orientado con incremento del ángulo de un grado respecto del anterior (180 puntos en total), se realizará una transformación de coordenadas polares a cartesianas.

3.1.1. Tratamiento de datos del láser con filtro de mediana

Con el fin de eliminar los posibles datos espurios, también llamados *outliers,* se lleva a cabo la aplicación de un filtro de mediana de 4 vecinos a todos los puntos del láser.

Suponiendo una ventana de 5 datos, siendo el dato central el valor del punto del láser a tratar y el resto se trata sus 4 vecinos más próximos, se realiza la mediana de dicha ventana de datos, ordenándolos en primer lugar en orden de menor a mayor y quedándonos con el valor situado en la mitad de la ventana con los datos ya ordenados. En la Figura 3.4 queda representado un ejemplo del procedimiento explicado, para el filtrado de un único dato (valor de un punto del láser), siendo éste el dato central de la ventana, en el que se ha asignado como dato a tratar un dato *outlier* que no guarda relación alguna con el valor de sus datos vecinos. Como podemos ver

al aplicar el filtro de mediana, el dato pasaría a tomar un valor que si guarda correlación con el valor de sus vecinos, por lo que se ha conseguido eliminar el *outlier*.

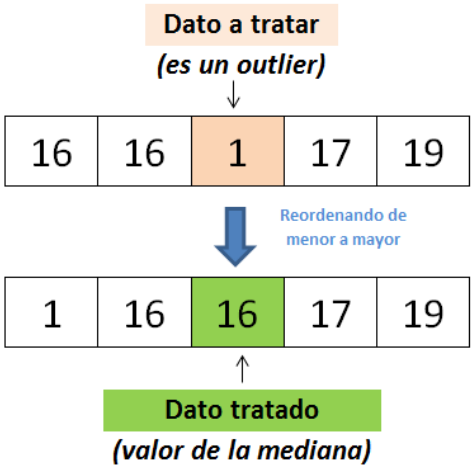


Figura 3.4: Ejemplo filtro de mediana

De este modo para realizar el filtrado de todos los puntos del láser será necesario ir desplazando dicha ventana a lo largo de todo el vector de datos.

3.1.2. Elección del rango de puntos que definen la recta

La elección de los puntos que definen la recta se realiza mediante dos pasos distintos:

- En primer lugar se aplica un filtrado de todos aquellos puntos que realicen una medida de mayor valor que una determinada distancia indicada d_{MAX}, tal como se ha representado en la Figura 3.5, guardando mientras tanto los índices, de la posición que ocupan en el vector de medidas, de todos aquellos puntos cuyo valor medido esté por debajo de dicha distancia. En un principio se introduce como valor por defecto la distancia máxima que puede medir el sensor, ya que los puntos laser que no encuentran ningún objeto en su recorrido toman éste valor. Sin embargo, es recomendable establecer un valor de d_{MAX} menor, para que los puntos que elija de la recta estén a una distancia menor o igual que dicho valor, y así disminuir la posibilidad de que coja puntos pertenecientes a una pared u objeto distintos que se encuentren más alejados. Esta distancia d_{MAX} será introducida como parámetro al lanzar el nodo que se está explicando.

- Una vez se conocen los índices de los puntos seleccionados como admisibles, es posible distinguir entre los distintos intervalos de puntos contiguos, fijándose en el valor de cada índice respecto al anterior. Tras la comparación de cada uno de éstos intervalos entre sí, en función del número de índices de puntos que contengan, se seleccionará aquel con mayor cantidad puesto que será el que tenga la probabilidad más elevada de corresponderse con la pared más cercana al robot. En la Figura 3.5 se ha representado un ejemplo en el que el algoritmo encontraría dos intervalos distintos A y B, escogiendo el A, al contener más cantidad de puntos.

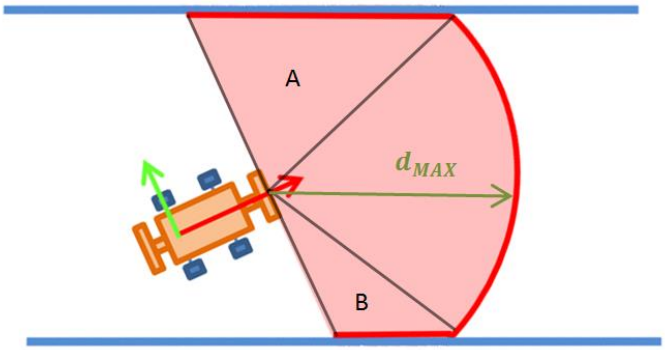


Figura 3.5: Elección de puntos para calcular la orientación de la pared

Una vez se poseen los índices de los puntos escogidos i_p , se procede a leer la medida de la distancia de los mismos en el vector de puntos que proporciona el sensor. De esta forma, mediante esa distancia d_p y el ángulo de cada punto θ_p , obtenido a partir del ángulo incremental θ_{inc} entre cada punto, ya es posible obtener las coordenadas de todos los puntos en coordenadas relativas a la referencia del sensor láser, teniendo en cuenta que tiene la misma orientación que la del robot. En la Figura 3.6 están representados los índices y ángulos de los principales puntos del láser, en el que se observa el origen de la orientación de los puntos, y el punto correspondiente al primer índice de todo el vector de medidas.

$$\theta_p = \left(-\frac{\pi}{2} + \left(i_p * \theta_{inc} \right) \right) \qquad \in \left[-\frac{\pi}{2} , \frac{\pi}{2} \right] \tag{3.1}$$

$$coordenadas\ punto\ p \begin{cases} x_p = d_p * \cos(\theta_p) \\ y_p = d_p * \text{sen}(\theta_p) \end{cases} \tag{3.2}$$

De este modo quedaran calculadas las coordenadas de todos los puntos de la pared que nos da el láser respecto de la referencia del láser frontal, por lo que basta con realizar una transformación que contenga únicamente

traslación sobre el eje x, con el valor de la distancia entre el eje trasero del robot y el láser frontal, para transformarlas a coordenadas relativas a la referencia del robot.

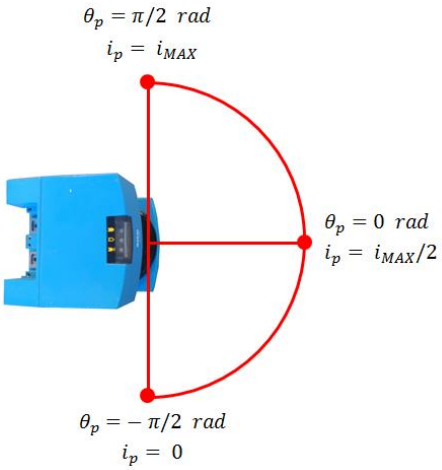


Figura 3.6: Ángulo e índice de los puntos característicos del láser

3.1.3. Cálculo de la orientación de la pared

Finalmente se obtiene la orientación de la pared mediante el cálculo previo de la regresión de la recta que forman los puntos del láser. Las siguientes expresiones muestran la forma de cálculo de la recta de regresión, donde N es el número de datos a los que se va a aproximar la recta, x_i e y_i corresponden a los valores de las coordenadas de cada punto del láser, m es la pendiente de la recta de regresión, y b el punto de corte con el eje de ordenadas de la recta.

$$\bar{y} = m * \bar{x} + b \tag{3.3}$$

$$m = \frac{N*S_{xy} - S_x*S_y}{N*S_{xx} - (S_x)^2} = \frac{N*\sum(x_i*y_i) - \sum(x_i)*\sum(y_i)}{N*\sum(x_i^2) - (\sum(x_i))^2} \tag{3.4}$$

$$b = \frac{S_y - m*S_x}{N} = \frac{\sum(y_i) - m*\sum(x_i)}{N} \tag{3.5}$$

Para el cálculo de la orientación de la pared, a partir de la pendiente de la recta de regresión, hay que tener en cuenta que lo que se pretende realizar es encontrar su orientación en el sentido contrario al de la orientación de dicha recta respecto del robot, para poder asignarle una referencia con la que establecer una condición de parada.

$$\theta_{recta} = \text{atan}(\, m \,) \qquad \in [\, -\tfrac{\pi}{2} \,,\, \tfrac{\pi}{2} \,] \tag{3.6}$$

$$\theta_{pared} = \begin{cases} \theta_{recta} + \pi & si \;\; \theta_{recta} < 0 \\ \theta_{recta} - \pi & si \;\; \theta_{recta} > 0 \end{cases} \qquad \in [-\pi, \pi] \tag{3.7}$$

Además de conseguir la condición de paro que se buscaba, también se consigue de éste modo conocer el sentido del giro inicial para realizar la trayectoria más corta hacia la orientación deseada. Se representa la dirección de giro con su velocidad angular ω.

$$\omega = \begin{cases} > 0 & si \;\; \theta_{pared} > 0 \\ < 0 & si \;\; \theta_{pared} < 0 \end{cases} \tag{3.8}$$

3.2. Estimación del movimiento final

Cuando el robot se encuentre realizando una maniobra marcha atrás, de forma que se esté alejando de la pared que tiene delante, acercándose a su vez a la orientación final deseada mediante un giro con el menor radio posible, será necesario conocer el momento en el que el robot pueda avanzar con velocidad lineal positiva, y el mismo radio de giro, concluyendo así el cambio de sentido sin necesidad de realizar ninguna maniobra adicional.

La forma de conocer en todo momento si el robot ya puede girar es deduciendo de nuevo la ecuación de la recta de regresión de forma periódica, para así calcular la distancia $r_{necesario}$ que hay entre el centro de rotación de ese posible giro final y la recta, de manera que se pueda comprobar si esa distancia es mayor que el radio mínimo de giro $r_{posible}$ con el que puede llegar a girar el punto del robot más cercano a la pared, tal y como se muestra en la Figura 3.7.

3.2.1. Cálculo de la distancia más corta entre centro de rotación y la pared

El cálculo de la ecuación de la recta que describe la pared, se realizará a través del mismo método utilizado en el apartado 3.1.3.

Una vez se tienen los parámetros de la recta de regresión de todos los puntos del láser que definen la pared, se procederá a calcular la distancia

$r_{necesario}$ entre el centro de rotación CR (x_{CR}, y_{CR}) del giro que concluiría el cambio de sentido, y el punto de la pared más cercano a ese centro. Ésta distancia se corresponde con el radio de giro que debería tener el punto del robot más cercano a la pared, para que la circunferencia que describe que fuera tangente a ella. En la figura 3.7 se puede visualizar perfectamente la distancia descrita.

$$r_{necesario} = d(CR,\ recta) = \frac{|\ m * x_{CR} - y_{CR} + b\ |}{\sqrt{m^2+1}} \tag{3.9}$$

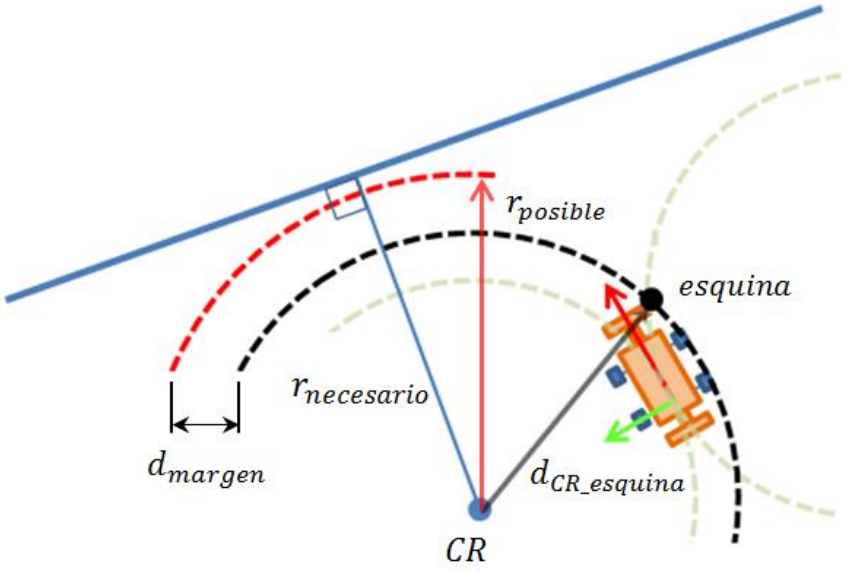


Figura 3.7: Trayectorias descritas durante la ejecución de la maniobra que concluye el cambio de sentido

3.2.2. Condición para iniciar la trayectoria final

La condición necesaria para saber si el robot ya puede realizar su última trayectoria, consistirá en el cálculo previo de la distancia que hay entre el centro de rotación del giro, y la esquina de la estructura del robot que más se acercará a la pared, añadiéndole también la distancia de seguridad que se pretende que tenga dicha esquina respecto de la pared. Así se consigue un radio $r_{posible}$ que describe la trayectoria circular que puede realizar la estructura completa del robot, sin sobrepasar una determinada distancia de seguridad con la pared.

Para el cálculo de la distancia entre el punto CR y la esquina, será necesario conocer las coordenadas de dicha esquina tomando como referencia la localización de CR a la que se le ha asignado la misma

orientación que el robot. Para ello se realizan las siguientes transformaciones, siendo d_{R_LF} la distancia entre el eje trasero del robot y laser frontal, y L lo que mide de ancho el robot. La distancia calculada corresponde al caso de que el giro final fuese con velocidad angular positiva, teniendo el mismo valor que si se hubiera calculado con el caso contrario.

$$^{R}X_{CR} = (\,0\,,\, r_{MIN}\,,0\,) \;\Rightarrow\; {}^{R}T_{CR} \tag{3.10}$$

$$^{R}X_{esquina} = \left(\, d_{R\_LF}\,, -\frac{\mathrm{L}}{2}\,,\, 0\,\right) \;\Rightarrow\; {}^{R}T_{esquina} \tag{3.11}$$

$$({}^{R}T_{CR})^{-1} * {}^{R}T_{esquina} = {}^{CR}T_{esquina} \;\Rightarrow\; {}^{CR}X_{esquina} = (\,x_e\,,\, y_e\,,\, \theta_e\,) \tag{3.12}$$

$$d_{CR\_esquina} = \sqrt{{x_e}^2 + {y_e}^2} \tag{3.13}$$

De este modo, y según se ha comentado anteriormente, la distancia calculada corresponderá con el radio que es capaz de realizar la esquina del robot que más se acerque a la pared. Así el radio con el compararemos con $r_{necesario}$, vendrá dado por la distancia calculada, añadiéndole un margen de seguridad con la pared.

$$r_{posible} = d_{CR\_esquina} + d_{margen} \tag{3.14}$$

Se comprueba si es posible comenzar la trayectoria final con velocidad lineal positiva, o seguir maniobrando marcha atrás, en función de si el robot es capaz de ejecutar con su radio mínimo el giro sin necesidad de una nueva maniobra. La velocidad angular vendrá dada por el sentido en que se esté realizando el giro.

$$condicion\ de\ fin\ de\ maniobra \begin{cases} v < 0 & si\ r_{necesario} < r_{posible} \\ v > 0 & si\ r_{necesario} > r_{posible} \end{cases} \tag{3.15}$$

Para facilitar el entendimiento de todos los radios y distancias calculados, se han representado en la Figura 3.7 mediante el ejemplo de un cambio de sentido en el que es necesaria la ejecución de una maniobra, siendo el momento exacto de la imagen, aquel en el que el robot ya tendría la posibilidad de concluir dicho cambio de sentido al cumplir con la condición de fin de maniobra citada anteriormente.

Capítulo 4
Resultados

A continuación se van a exponer los resultados obtenidos en algunos ejemplos de los experimentos realizados tanto en la parte de simulación, como en las pruebas de campo con el robot real. En el Anexo A están incluidos los resultados del resto de pruebas realizadas.

4.1. Simulación

Para realizar las simulaciones, se ha hecho uso del simulador *Gazebo* en comunicación con *ROS*, que ha permitido diseñar y corregir el programa realizado, sin necesidad de presentar los problemas de tiempo y comodidad que supondría depurar directamente el programa con el robot real. Para la visualización de la información captada por los sensores, así como las trayectorias calculadas y seguidas, se ha hecho uso de la herramienta de visualización 3D *RViz*, que capta la información de los *topics* publicados por cualquier nodo. La explicación del funcionamiento del software mencionado queda reflejada en el Anexo B, así como los gráficos de comunicación entre nodos.

Para realizar los experimentos se ha utilizado un modelo realista del Robucar-TT en el simulador, implementado en un proyecto anterior [6], con el que se ha podido comprobar el correcto funcionamiento de los nodos *ROS* realizados en éste trabajo.

4.1.1. Trayectorias de Dubins

Para el primer experimento realizado en este proyecto se ha dispuesto de un escenario plano sin obstáculos en el simulador *Gazebo*, tal y como se muestra en la Figura 4.1 que sería un entorno similar al que se utilizará en

las pruebas de campo y con el que se van a probar las trayectorias de Dubins generadas.

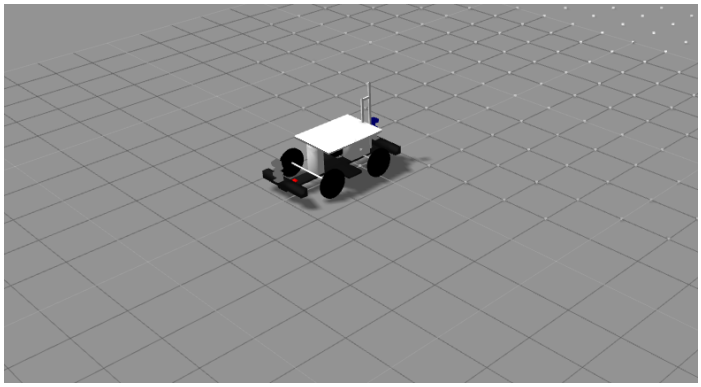


Figura 4.1: Escenario en simulador Gazebo para trayectorias de Dubins.

Para la ejecución del nodo *ROS* implementado, se introduce como parámetros el vector localización de la situación final deseada respecto a la situación inicial del robot, la velocidad lineal máxima que queremos que alcance el robot, y los parámetros característicos del robot siendo éstos la medida de longitud entre ejes y el ángulo máximo de giro de la rueda "virtual" que habría en el centro del eje de las ruedas delanteras del robot. De esta manera se permite el uso de éste nodo para cualquier robot tipo coche, simplemente introduciendo las dimensiones de la estructura del robot en el que se desee ejecutar, sin necesidad de volver a compilar el programa.

Una vez lanzado el nodo, el programa sigue los pasos descritos en el capítulo 2, calculando inicialmente las cuatro posibles trayectorias con las que el robot puede llegar a la localización objetivo, y quedándose con la trayectoria más corta. En la Figura 4.2 se ha mostrado un ejemplo en el que se observa con claridad cómo el algoritmo genera las cuatro posibles trayectorias y selecciona la más corta, visualizando todas ellas en *RViz* y representando, en la trayectoria escogida como la óptima, las referencias de los puntos característicos de la misma.

Tras la elección de la trayectoria, se realiza su seguimiento mediante la lectura de la odometría del robot, que en el caso de la simulación es la proporcionada por el simulador *Gazebo*. Durante la ejecución del nodo, se visualiza en el entorno *RViz* la trayectoria teórica que el robot debe seguir, así como la representación en tiempo real de la trayectoria que va siguiendo el robot, con lo que se ha conseguido depurar el programa con mayor facilidad al no tener que realizar una extracción externa de los datos recogidos para comprobar la precisión del seguimiento de dicha trayectoria.

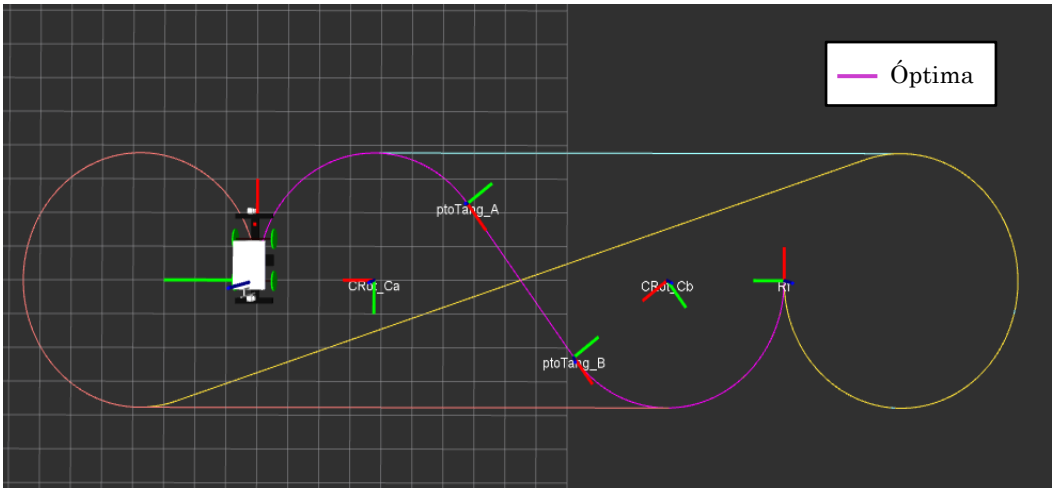


Figura 4.2: Trayectorias posibles de Dubins generadas.

Se presenta a continuación un ejemplo de simulación en el que se le indica al robot que llegue a la localización objetivo definida en la expresión (4.1) donde los dos primeros valores correspondientes a las coordenadas se expresan en metros, y el tercero correspondiente a la orientación en grados, todos ellos respecto de los ejes de la posición inicial del robot. En la Figura 4.3 quedan representadas la trayectoria teórica y la realizada, de modo que se puede apreciar visualmente el correcto seguimiento de la trayectoria marcada.

$$ {}^{R_i}X_{R_f} = (-15\ ,\ 7\ ,\ 90\ ) \tag{4.1}$$

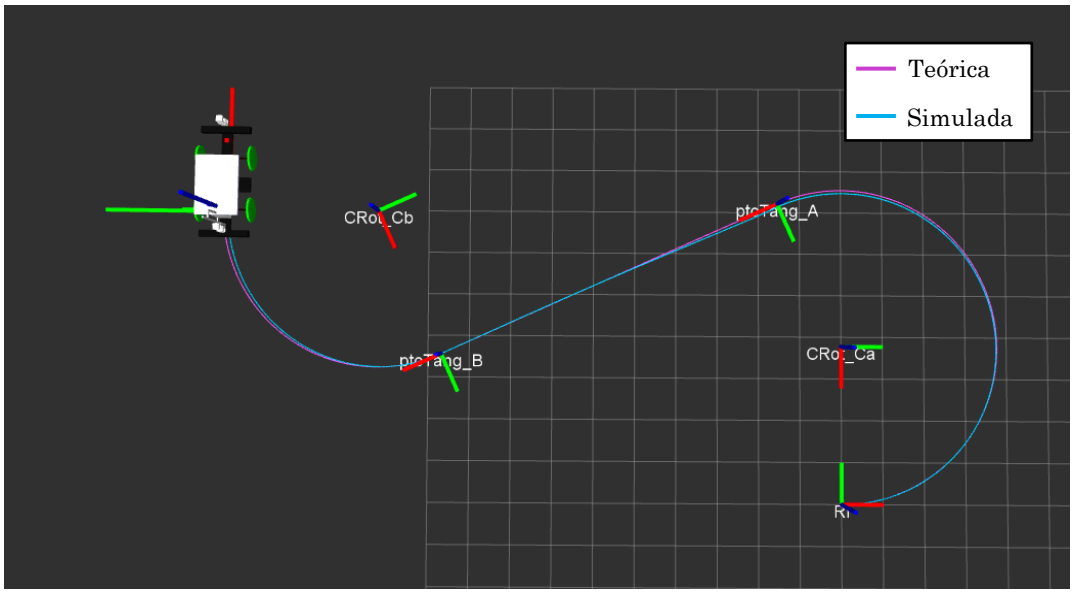


Figura 4.3: Simulación de trayectoria de Dubins seguida.

Aunque en el caso visto no se aprecia demasiado la actuación del controlador en el tramo recto debido a que se realiza el seguimiento con precisión, se observa como el pequeño error existente al comenzar el tramo respecto de la trayectoria teórica es corregido. En el Anexo A se pueden ver algunas pruebas de campo donde se puede ver mejor la actuación de dicho controlador.

4.1.2. Cambio de sentido realizando maniobras

En el segundo experimento realizado se dispone de un escenario de *Gazebo*, mostrado en la Figura 4.4, en el que el robot se encuentra dentro de un túnel, en el cual se ha introducido una columna para simular las posibles irregularidades que tendría un entorno real.

En este caso, los parámetros necesarios para la ejecución del nodo serán la velocidad lineal máxima, los parámetros característicos del robot, la distancia mínima que queremos mantener con la pared, y la distancia máxima que queremos que tengan los puntos del láser para obtener la recta de regresión de los puntos que miden la pared.

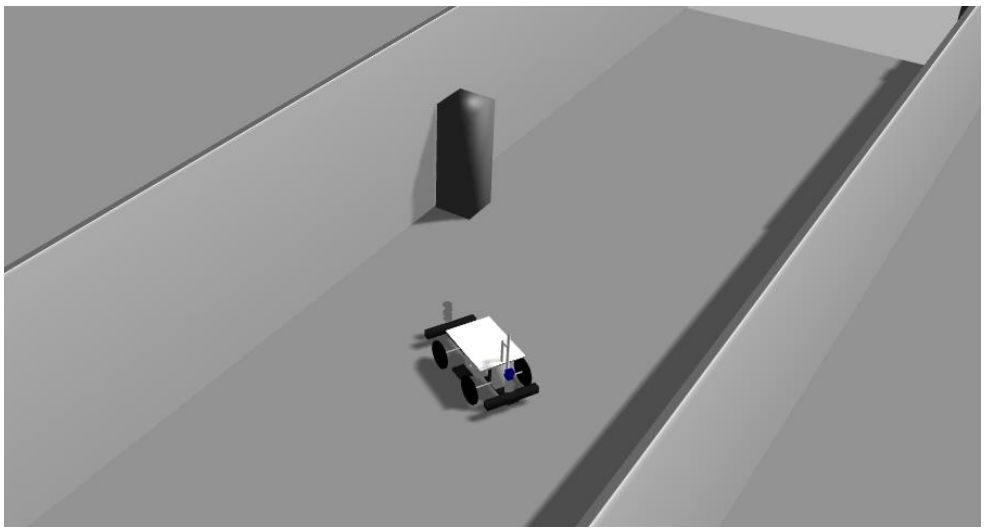


Figura 4.4: Escenario en simulador Gazebo para maniobra de cambio de sentido.

Cuando ejecutamos el nodo, el programa obtiene la recta de regresión que permitirá calcular la localización de la pared respecto de la situación inicial del robot, visualizada en color naranja en RViz tal y como se puede ver en la Figura 4.5. Como se puede apreciar también se visualizan todos los puntos que nos devuelven los láseres frontal y trasero del robot mediante una nube de puntos, además de la trayectoria seguida por el robot una vez

comenzada la maniobra en color azul. También se han representado las referencias de dos puntos de la recta para visualizar la orientación con la que debe acabar alineado el robot.

En la Figura 4.5, se representa la localización inicial del robot respecto de la pared, donde se puede apreciar la situación de la recta de regresión respecto de la ubicación real de la pared dada por los puntos del láser. La columna introducida produce una pequeña desviación de la recta de regresión, pero al no ser demasiado significativa el robot consigue realizar correctamente el cambio de sentido, tal y como se muestra en la Figura 4.6, produciéndose únicamente un pequeño error en el alineamiento con la pared al finalizar la maniobra.

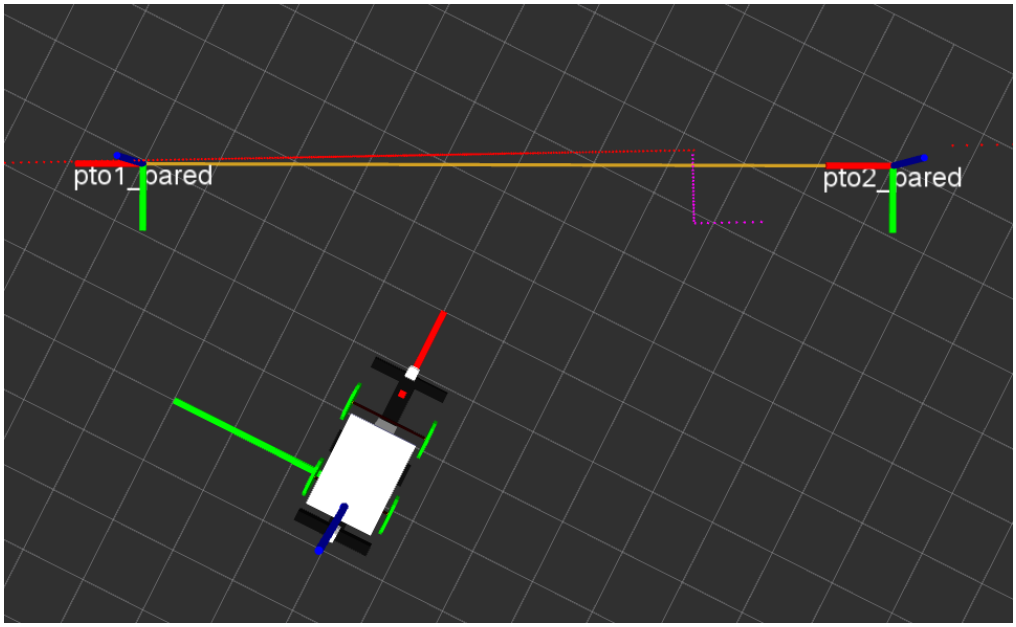


Figura 4.5: Situación inicial antes de comenzar la maniobra para cambio de sentido.

Una vez concluido el cambio de sentido, en la Figura 4.6 quedan reflejadas las maniobras realizadas por el robot, donde se puede comprobar la correcta elección del sentido del giro inicial para el que la maniobra concluye antes, así como el momento en el que el robot puede efectuar la maniobra final sin sobrepasar la distancia de seguridad que se requiere mantener con la pared, ya que en caso contrario el robot habría realizado una maniobra adicional marcha atrás.

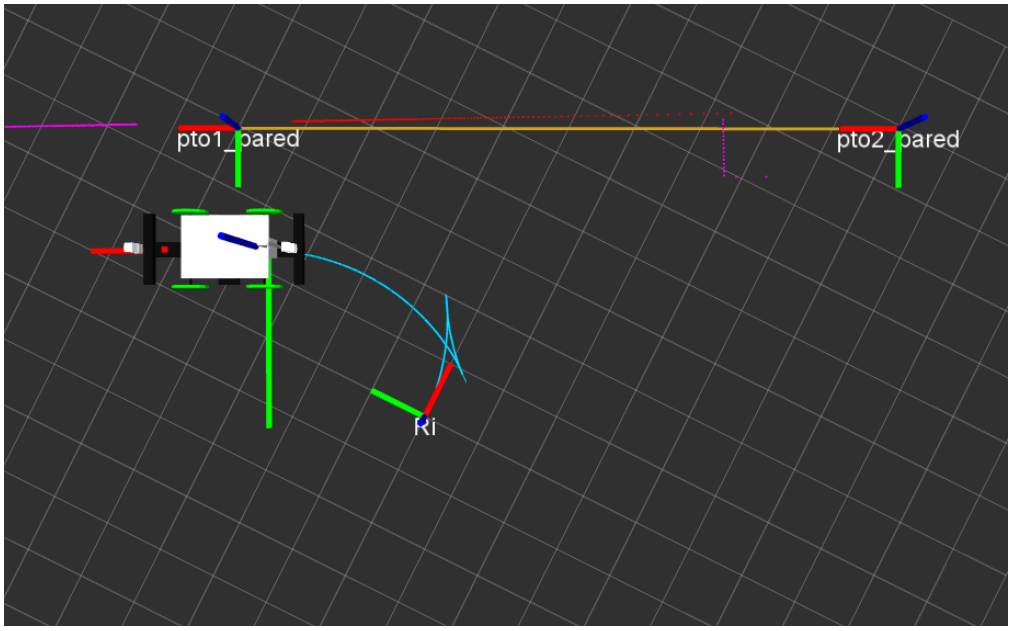


Figura 4.6: Situación final una vez concluido el cambio de sentido

4.2. Pruebas de campo

En el caso de las pruebas con el robot real, el modelo de simulación existente del Robucar-TT ha permitido poner a punto el software y preparar el conjunto de experimentos a realizar con el robot real. Para la experimentación con el robot real, sólo se ha necesitado modificar el nombre del *topic* por el que se transmite la información de la odometría del robot sin necesidad de modificar nada más del programa.

A continuación se va exponer un ejemplo de cada uno de los dos experimentos realizados en este proyecto en una aplicación real, donde analizaremos también la diferencia entre los resultados teóricos, los de simulación y el resultado con el robot real. En el Anexo A se han incluido otro conjunto de pruebas adicionales realizadas para cada experimento.

4.2.1. Trayectorias de Dubins

En este caso se ha realizado un experimento en el que se comprueba el giro del robot hacia ambos sentidos de giro para comprobar su correcto funcionamiento en ambos casos. Para ello se le introduce como localización objetivo la asignada en la expresión (4.2) a la que el robot tiene que llegar.

$$^{R_i}X_{R_f} = (10\,,\ 3\,,\ 135\,) \tag{4.2}$$

En la Figura 4.7 se aprecia las trayectorias realizadas por el robot tanto en el entorno de simulación (rojo), como en un escenario real (verde), además de la trayectoria teórica calculada que representa la trayectoria óptima a seguir (azul). Como podemos observar, el seguimiento de la trayectoria se realiza en ambos casos de manera correcta acumulando un cierto error al finalizarla, siendo éste un error pequeño en comparación a la distancia recorrida.

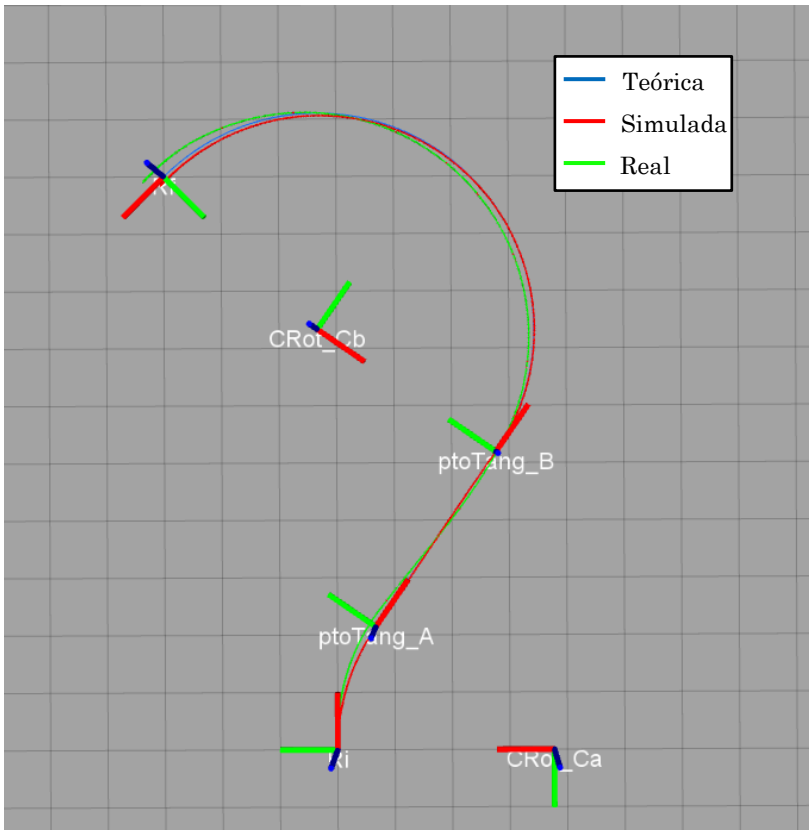


Figura 4.7: Trayectorias de Dubins seguidas, visualizadas en RViz

Con el fin de comprobar la precisión con que se alcanza la localización objetivo indicada, se procede a realizar representación de los datos de localización mediante Matlab, de forma que se visualice las coordenadas de la localización numéricamente. Dicha representación queda reflejada en la Figura 4.8, donde se ha visto que el error producido al finalizar la trayectoria real del robot es de unos 10 centímetros. Este error se puede explicar por el error de estimación de la localización a partir de la odometría del robot real.

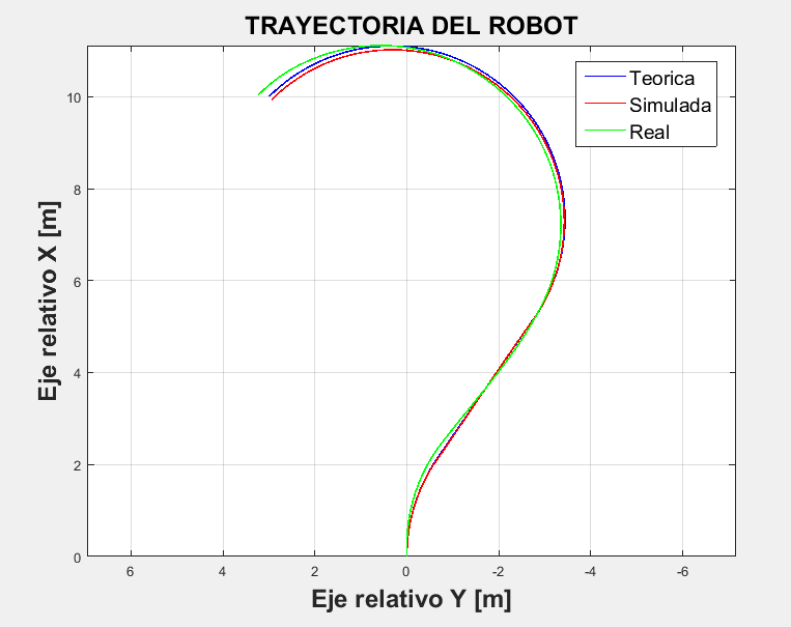


Figura 4.8: Trayectorias de Dubins seguidas, visualizándolas con Matlab.

En la Figura 4.8, se observa el seguimiento realizado con éxito en cualquiera de los casos, aunque con un error de posición del robot respecto de la situación final. Para poder visualizarlo mejor se realiza una ampliación de la localización final en la Figura 4.9, donde se observa que en el caso de simulación el error es de apenas unos 5 centímetros en ambos ejes, y en el caso del robot real el error es de 2.5 centímetros en el eje *x*, y de unos 20 centímetros en el eje *y*.

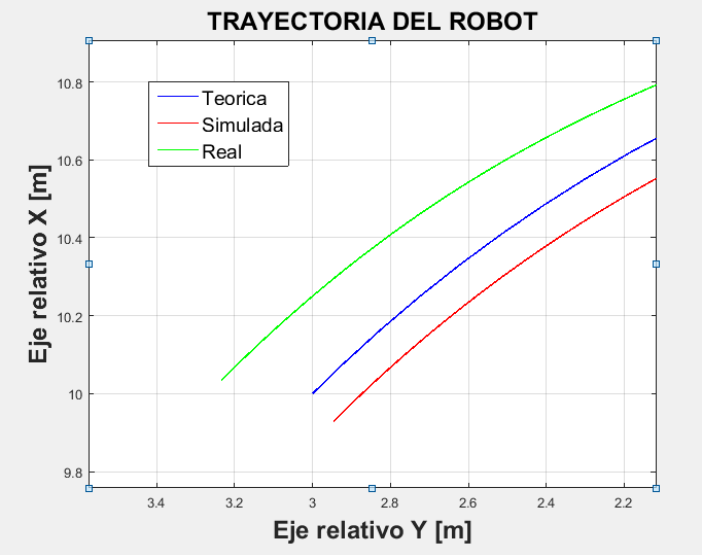


Figura 4.9: Detalle de las trayectorias de Dubins seguidas, visualizadas con Matlab (ampliada).

Finalmente, en la Figura 4.10 se han representado las gráficas de la velocidad y el ángulo de giro de la rueda virtual que habría en el medio del eje de las ruedas delantero, que son las variables controladas en el robot para generar el movimiento.

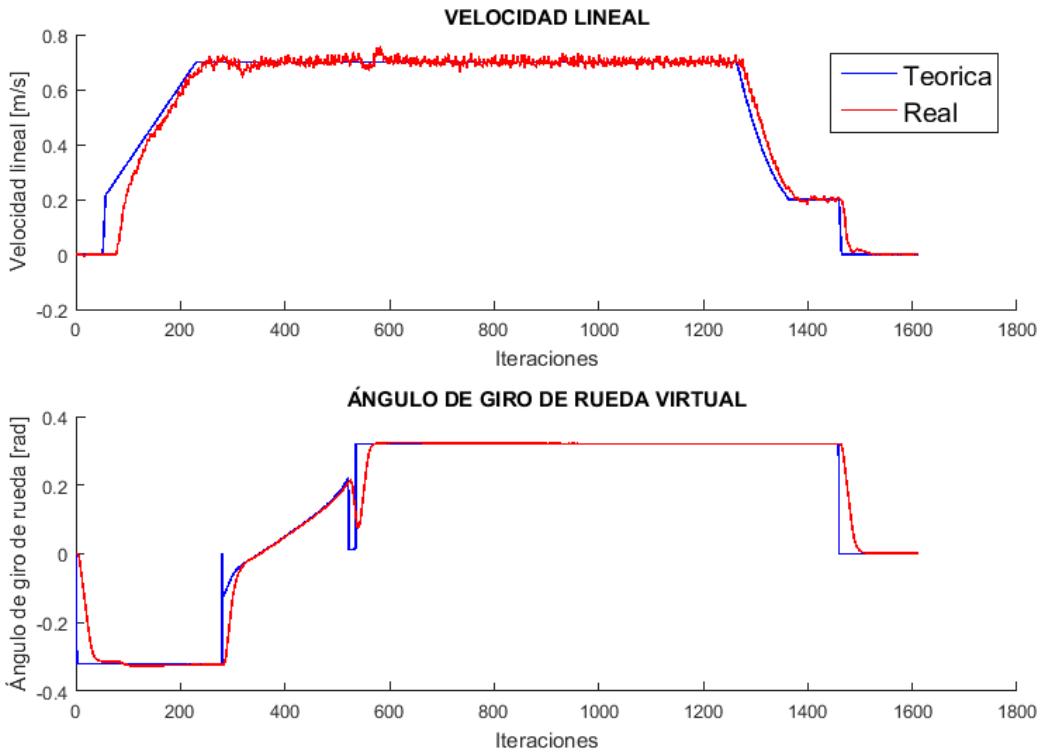


Figura 4.10: Gráficas de velocidad lineal, y ángulo de giro de las ruedas direccionales.

Respecto de la velocidad lineal, en la gráfica se observa los intervalos de aceleración y deceleración realizados. En el caso de la aceleración se introduce una consigna inicial de 0.2 m/s, tras lo que se realiza la aceleración progresiva hasta la consigna máxima mediante una referencia de rampa. En la deceleración se realiza un control de la velocidad proporcional al error normalizado entre 0 y 1, del ángulo de tramo circular final que le queda por recorrer una vez éste es menor que 20 grados.

En la gráfica del ángulo de giro de las ruedas direccionales, se puede observar los distintos tramos de la trayectoria, donde hasta la iteración 300 aproximadamente sería la primera trayectoria circular con dirección hacia la derecha (ángulo de ruedas negativo), y hasta la iteración 550 se produce la actuación del control proporcional mencionado en el capítulo 2, para que el robot se alinee con el tramo recto que debería seguir. A partir de ahí se observa cómo se realiza la trayectoria circular final, siendo la que más tiempo cuesta llevar a cabo en este caso.

4.2.2. Cambio de sentido realizando maniobras

En este experimento se han realizado dos pruebas en las que el robot ha realizado un cambio de sentido efectuando maniobras.

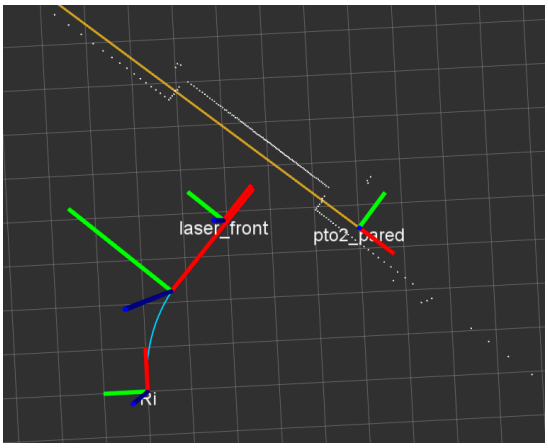
En la primera de ellas, dado el escenario disponible para la prueba de éste nodo, se realiza el cambio de sentido para el caso de que solo haya una pared junto al robot de manera que el cambio de sentido se realice en el menor tiempo posible, y con la menor distancia a la pared que se pueda, teniendo que llevarlo a cabo realizando maniobras. En la Figura 4.11 se visualiza el escenario real utilizado, donde se comprueba como el robot consigue realizar el cambio de sentido a pesar de las irregularidades de la pared que presentan la cochera y la escalera que se muestran en la Figura 4.11 (b).

(a) Situación inicial (b) Situación final

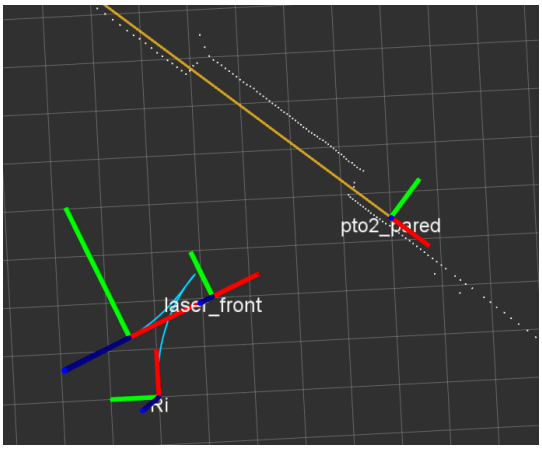
Figura 4.11: Escenario real para el cambio de sentido

En la Figura 4.12 se observa en azul la trayectoria seguida por el robot en las distintas maniobras realizadas, así como la representación de los puntos que mide el láser en todo momento (en blanco) y la recta de regresión de los puntos que mide de la pared en la situación inicial del robot antes de comenzar el movimiento (en naranja). La referencia (triedro) de mayor tamaño observada es la referencia del robot. En la Figura 4.12 (c) se observa que la trayectoria seguida concuerda con los resultados esperados y los obtenidos en simulación (Figura 3.1 y 4.6).

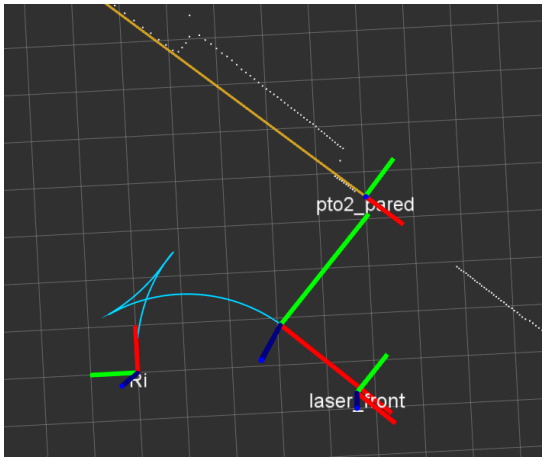
Para comprobar el proceso seguido por las variables de control del robot, se han representado en la Figura 4.13 la gráficas de velocidad lineal y ángulo de giro de las ruedas direccionales, donde se puede ver con claridad las distintas maniobras realizada.



(a) Primera maniobra



(b) Segunda maniobra



(c) Tercera y última maniobra

Figura 4.12: Trayectoria seguida en cambio de sentido con una pared

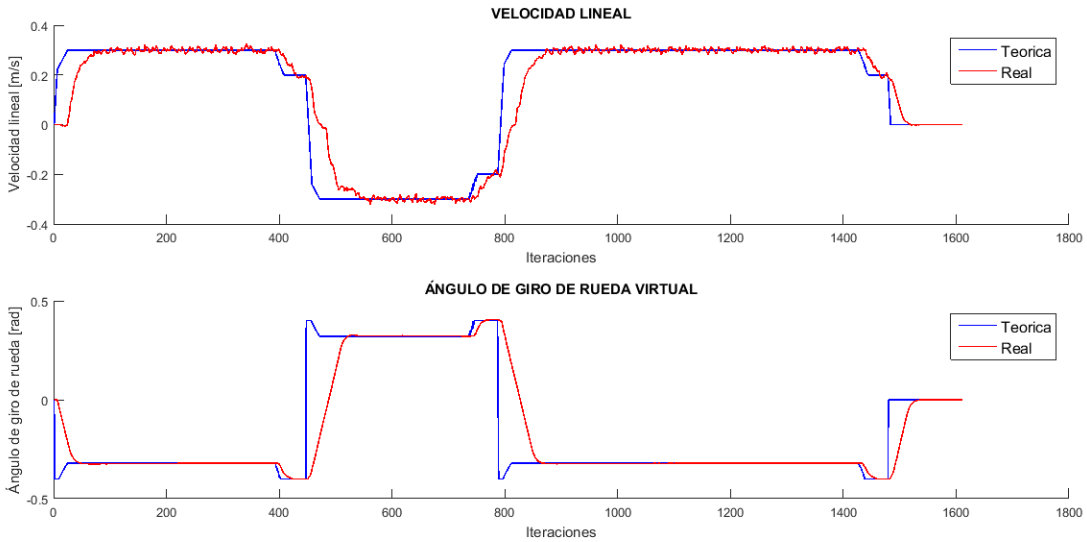
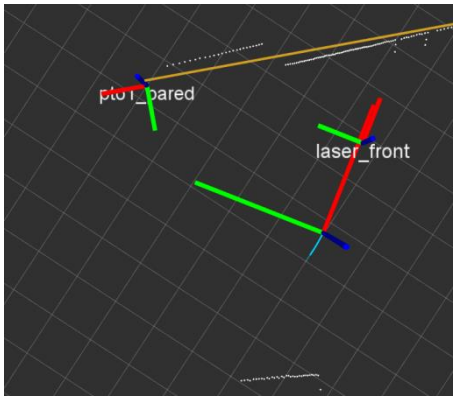
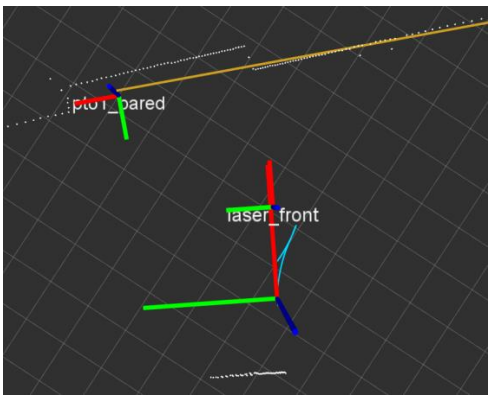


Figura 4.13: Velocidad y ángulo de las ruedas direccionales con una pared

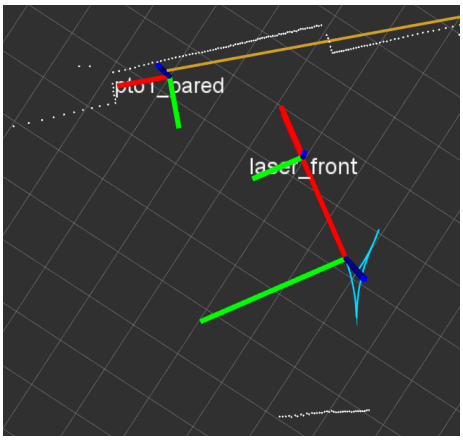
En la segunda prueba realizada se lleva a cabo un cambio de sentido en el mismo escenario que el caso anterior, añadiendo otra pared paralela a la mostrada en la Figura 4.11 mediante unos paneles colocados tras el robot, tal y como se puede apreciar en los puntos obtenidos del láser trasero que se representan en la Figura 4.14.



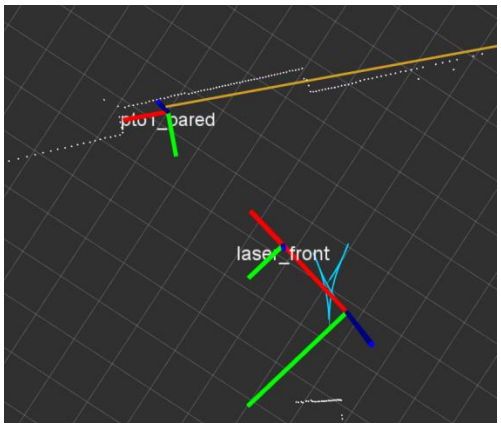
(a) Primera maniobra



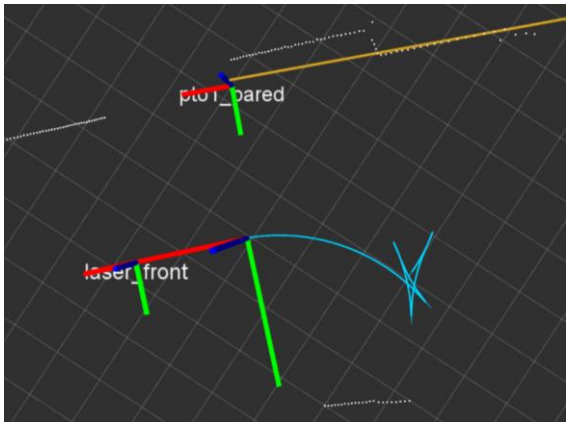
(b) Segunda maniobra



(c) Tercera maniobra



(d) Cuarta maniobra



(e) Quinta maniobra

Figura 4.14: Trayectoria seguida en cambio de sentido con dos paredes

De esta manera se realiza el experimento como si se tratase de un escenario dentro de un túnel en el que el robot tendrá que realizar más de una maniobra marcha atrás para poder efectuar el cambio de sentido. Se han llevado a cabo las maniobras en distinto sentido a la prueba anterior para comprobar su correcto funcionamiento en ambas direcciones.

En la Figura 4.14 (b) y (c) se ve como el robot tiene que realizar otra maniobra ya que detrás se ha detectado que se sobrepasa la distancia de seguridad con la pared trasera. Una vez finaliza la cuarta maniobra en la Figura 4.14 (d) el robot detecta que ya es posible finalizar el cambio de sentido sin necesidad de nuevas maniobras, por lo que procede a realizar la última maniobra (Figura 4.14 (e)).

Igual que en el caso anterior en la Figura 4.15 se representan las gráficas de velocidad y ángulo de giro de las ruedas direccionales donde se puede ver las distintas maniobras realizadas por el robot. En este caso se puede comprobar cómo se han realizado dos maniobras adicionales respecto del caso anterior

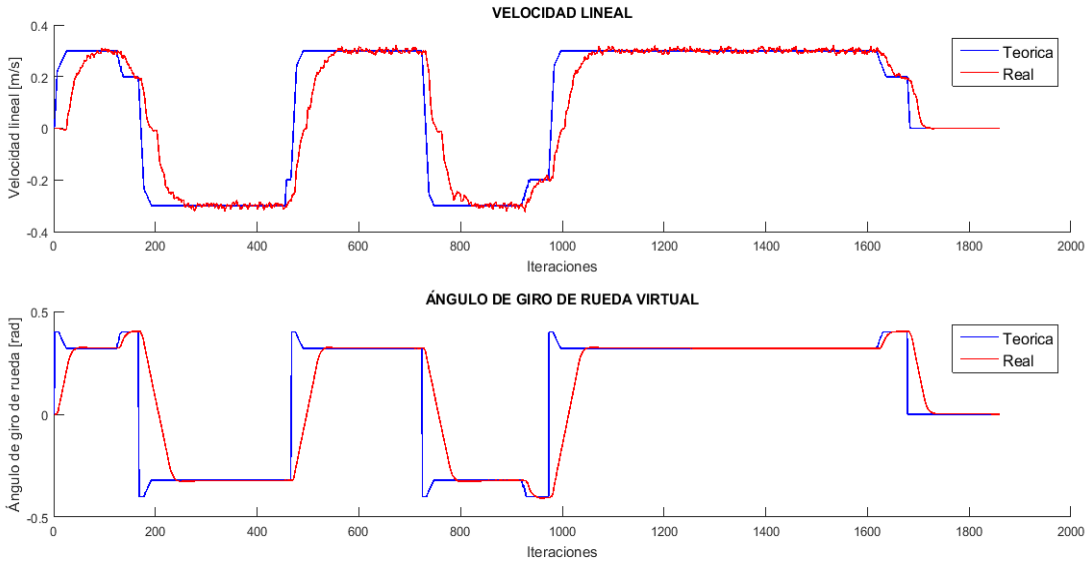
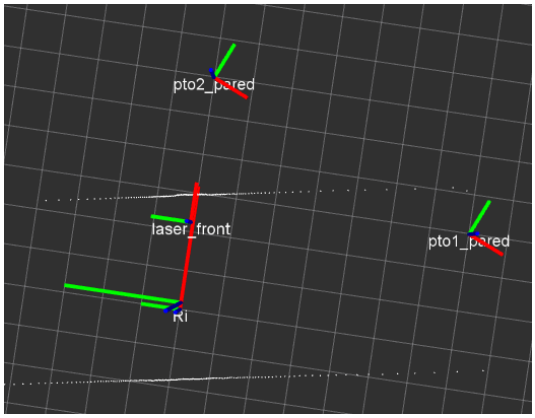


Figura 4.15: Velocidad y ángulo de las ruedas direccionales con dos paredes

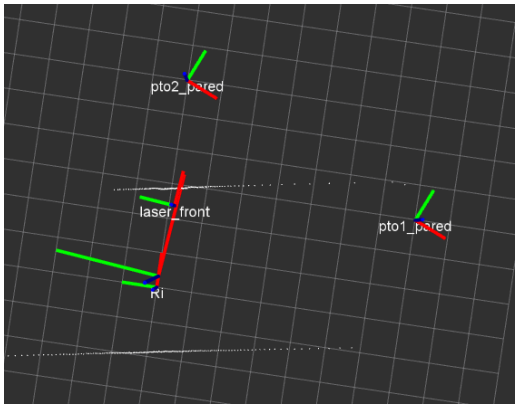
Adicionalmente a las pruebas anteriores, el nodo ROS creado para el cambio de sentido ha sido probado con el Robucar-TT en el antiguo túnel ferroviario de Somport en Canfranc (Figura 4.16) donde, a pesar de no estar el programa en su última versión en el momento en que se realizó la prueba, el robot consigue efectuar un cambio de sentido aunque la orientación final no sea paralela a la de la pared, tal y como aparece en la Figura 4.17. La orientación errónea obtenida de la pared es debido a los datos espurios que no habían sido filtrados, produciendo que la recta que aproxima la pared tenga un error muy grande.

Figura 4.16: Robucar-TT dentro del túnel de Somport

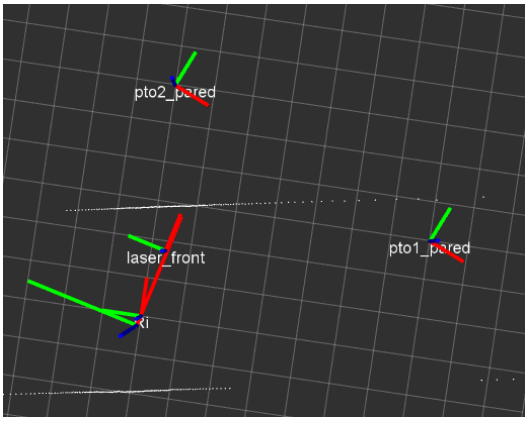
Las gráficas de velocidad y ángulo de giro de las ruedas direccionales se representan en la Figura 4.18, donde se puede comprobar las distintas maniobras realizadas dentro del túnel que aparecen en la Figura 4.17.



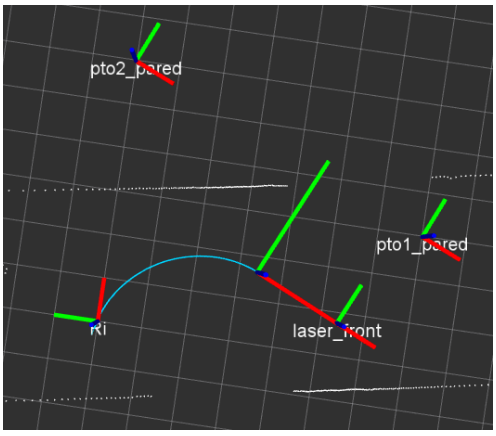
(a) Situación inicial



(b) 1ª maniobra



(c) 2ª maniobra



(d) 3ª maniobra

Figura 4.17: Trayectoria seguida en cambio de sentido en túnel de Somport

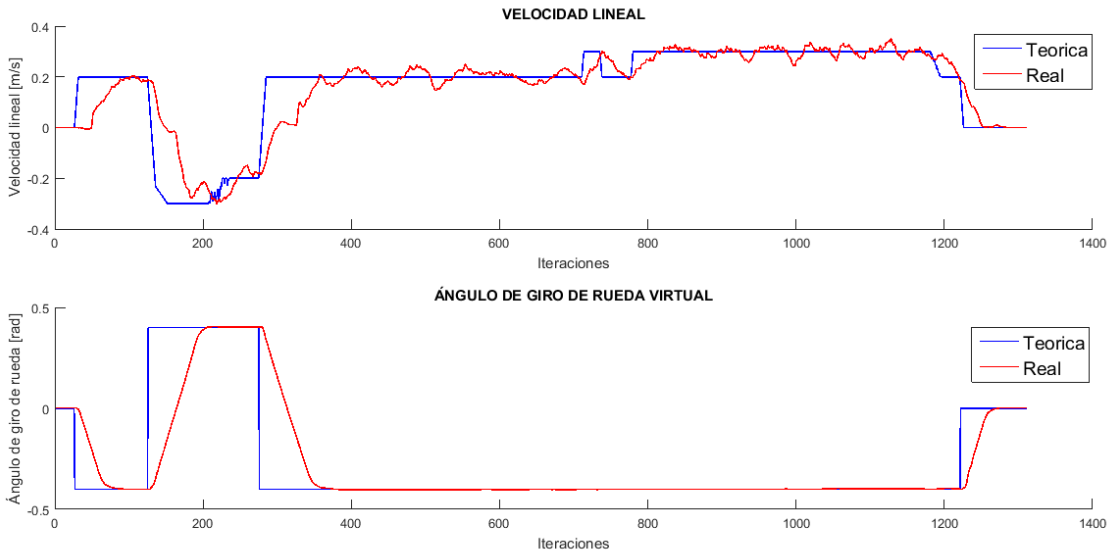


Figura 4.18: Velocidad y ángulo de las ruedas direccionales en túnel de Somport

Capítulo 5
Conclusiones

En este proyecto se ha realizado el control y seguimiento de trayectorias para robots tipo coche mediante una implementación real en el robot Robucar-TT de la Universidad de Zaragoza. Para ello se ha realizado un estudio sobre el funcionamiento de la plataforma ROS, así como de las técnicas necesarias para la implementación de los objetivos marcados.

La aplicación de la trayectorias de Dubins ha supuesto realizar inicialmente el cálculo teórico de las relaciones geométricas necesarias para una aplicación real de las mismas, así como un de un análisis minucioso sobre la elección de la trayectoria óptima de entre todas las posibles, intentando en un principio detectarla sin necesidad del cálculo de todas ellas, sino mediante la relación entre la posición inicial del robot y la localización objetivo. Finalmente, puesto que siempre existían casos especiales en los que el algoritmo de elección fallaba, se abordó el problema mediante la elección de la trayectoria tras el cálculo de todas las posibles.

En el caso de la implementación del cambio de sentido, se ha aplicado un tratamiento de los datos del láser mediante un filtro de mediana, eliminando los datos espurios para evitar así una aproximación errónea de la regresión lineal que dicta la ecuación de la recta que forma la pared respecto de la referencia de coordenadas del robot.

El resultado final del primer experimento, es el preciso cálculo de la trayectoria de Dubins óptima, y un seguimiento aceptable de la misma teniendo en cuenta que la respuesta ante las consignas enviadas al robot tienen un cierto error, sin existir la posibilidad de corregirlas al presentarse el controlador de bajo nivel del robot, en particular la odometría, como una caja negra. Para corregir dicho error en la medida de lo posible se realizó un análisis de sensibilidad a algunos de los parámetros del robot, variando las características geométricas del robot introducidas como parámetro hasta

conseguir una respuesta de radios de giro similares a las consignas enviadas.

En el caso del cambio del sentido se han obtenido buenos resultados, consiguiendo realizarlo en un entorno abierto, con una única pared a pesar de presentar algunas irregularidades. En otra prueba se ha realizado un experimento de la situación del robot dentro de un túnel, mediante un panel situado tras el robot formando otra pared paralela a la del caso anterior, de manera que el robot tuviera que realizar varias maniobras hasta concluir el cambio de sentido. Para la aplicación más realista del caso del túnel, se llevó a cabo una prueba en el túnel de Somport consiguiendo realizar el cambio de sentido del robot en su interior, a pesar de los problemas comentados en el capítulo 4.

Como línea futura de continuación del trabajo realizado en este TFG, se encuentra principalmente la aplicación de las técnicas de Reeds-Shepp Curves junto con las trayectorias de Dubins, para el caso mencionado en el que alguna de las cuatro posibles trayectorias no pueda ser calculada debido a la proximidad entre la localización objetivo y la posición inicial del robot. En ese caso el robot debería realizar una o varias maniobras hasta conseguir llegar a esa localización objetivo. También es posible ampliar el diseño del algoritmo para las maniobras a entornos más complejos, con obstáculos más irregulares, que pueden aparecer en un entorno real de navegación.

Bibliografia

[1] G. Cho, y J. Ryeu, "An Efficient Method to Find a Shortest Path for a Car-Like Robot", *International Journal of Multimedia and Ubiquitous Engineering, Vol. 1, No. 1, March 2008.*

[2] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *The American Journal of Mathematics, vol. 79, no. 3, pp. 497–516, 1957*

[3] ROS, "Robot Operating System". http://www.ros.org/

[4] Gazebo, "Simulador multi-robot 3D". http://gazebosim.org/

[5] RViz, "Herramienta de visualización 3D para ROS". http://wiki.ros.org/rviz

[6] Carlos Gracia, "Modelado y simulación de un robot Robucar TT", Universidad de Zaragoza, CPS, Mayo 2013. https://zaguan.unizar.es/record/10425?ln=es

[7] Librerías de ROS, clase *tf::Transform*. http://docs.ros.org/jade/api/tf/html/c++/classtf_1_1Transform.html

[8] S. LaValle, "Planning algorithms", Cambridge University Press, 2006

Anexo A

Conjunto de pruebas realizadas

Trayectorias de Dubins

Simulación

En simulación se han obtenido muy buenos resultado en todas las pruebas realizadas independientemente de la complejidad de la trayectoria. Algunas de las pruebas realizas se muestran a continuación donde se puede observar el seguimiento de la trayectoria con error prácticamente nulo.

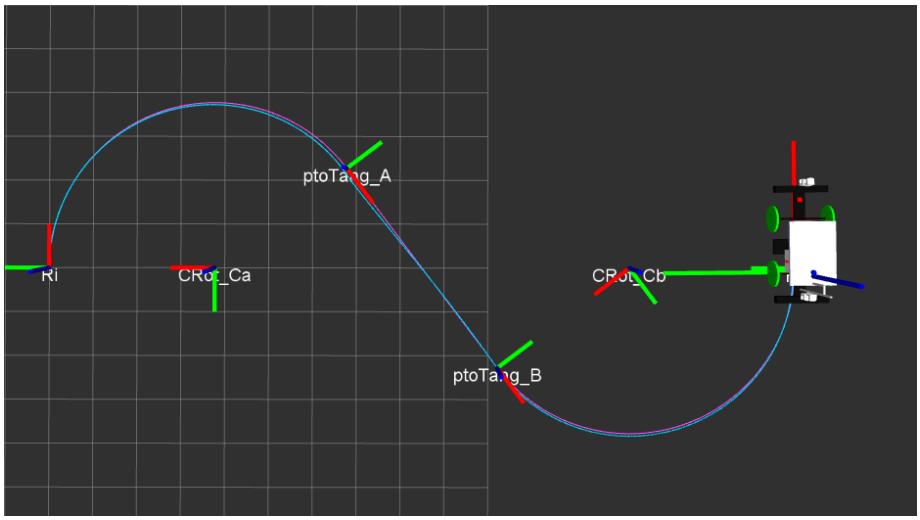


Figura A.1: Trayectoria de Dubins RSL hasta localización (0, -7, 0) .

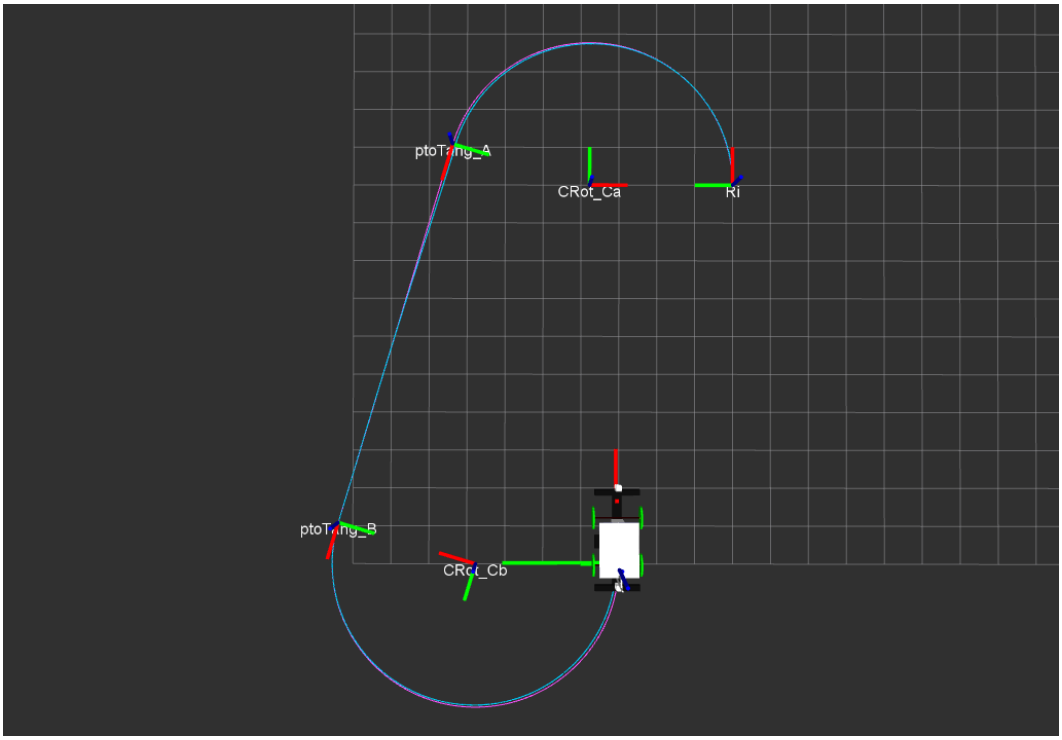


Figura A.2: Trayectoria de Dubins LSL hasta localización (-10, 3, 135) .

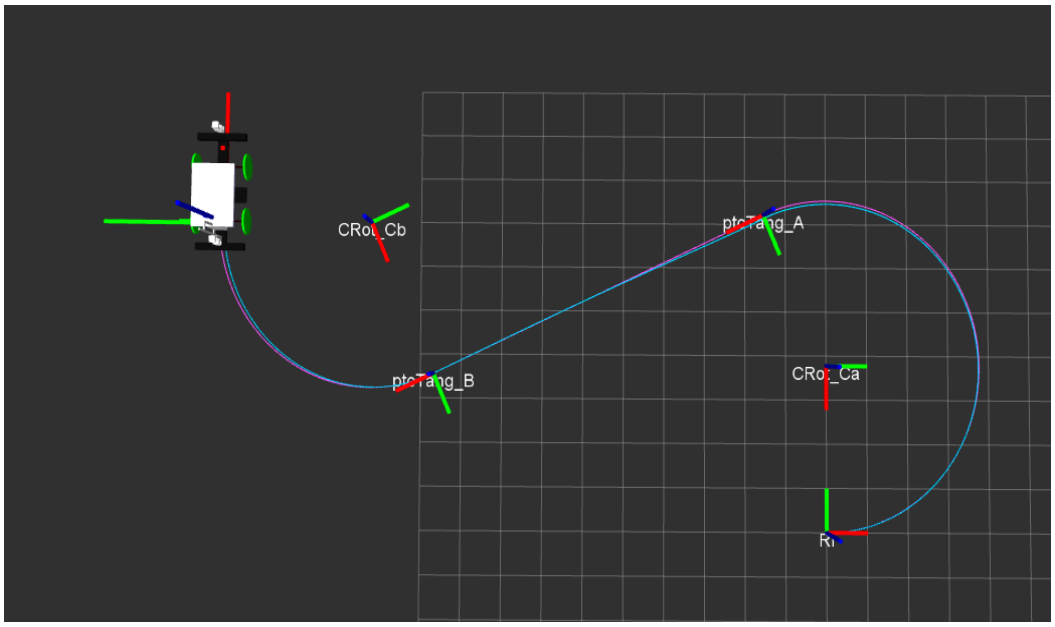


Figura A.3: Trayectoria de Dubins LSR hasta localización (-15, 7, 90) .

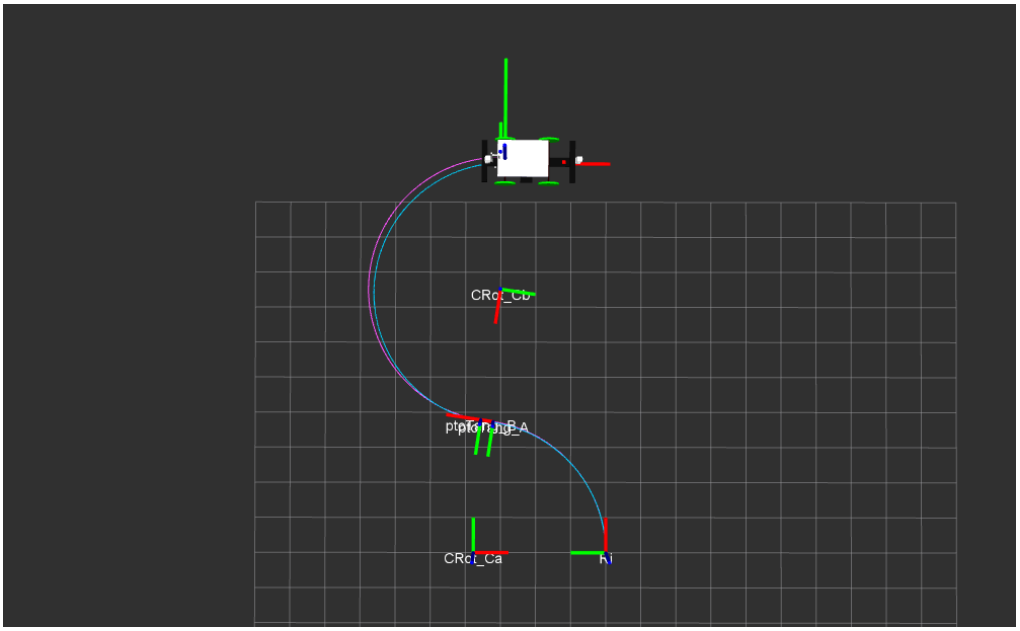


Figura A.4: Trayectoria de Dubins LSR hasta localización (10, 3, -90) .

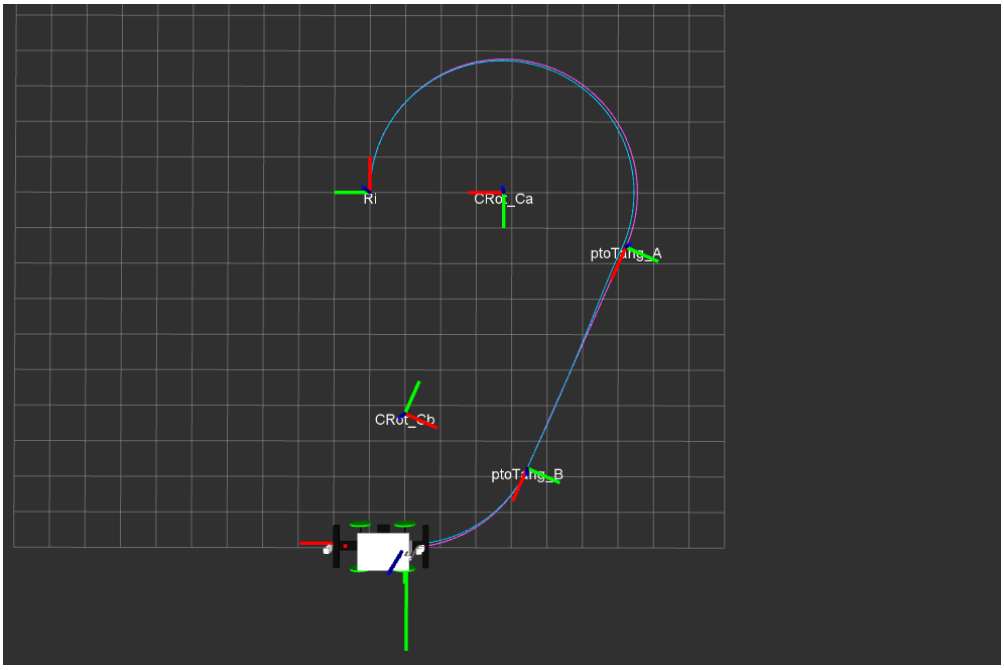


Figura A.5: Trayectoria de Dubins RSR hasta localización (-10, -1, 90) .

Pruebas de campo

En el caso de las pruebas de campo, los problemas derivados del funcionamiento del robot explicados en las conclusiones hacían que el seguimiento de la trayectoria en algunas ocasiones no fuera tan preciso como en simulación. A pesar de ellos se consiguió realizar varias pruebas correctamente. Algunas de las pruebas de campo realizadas quedan representadas a continuación.

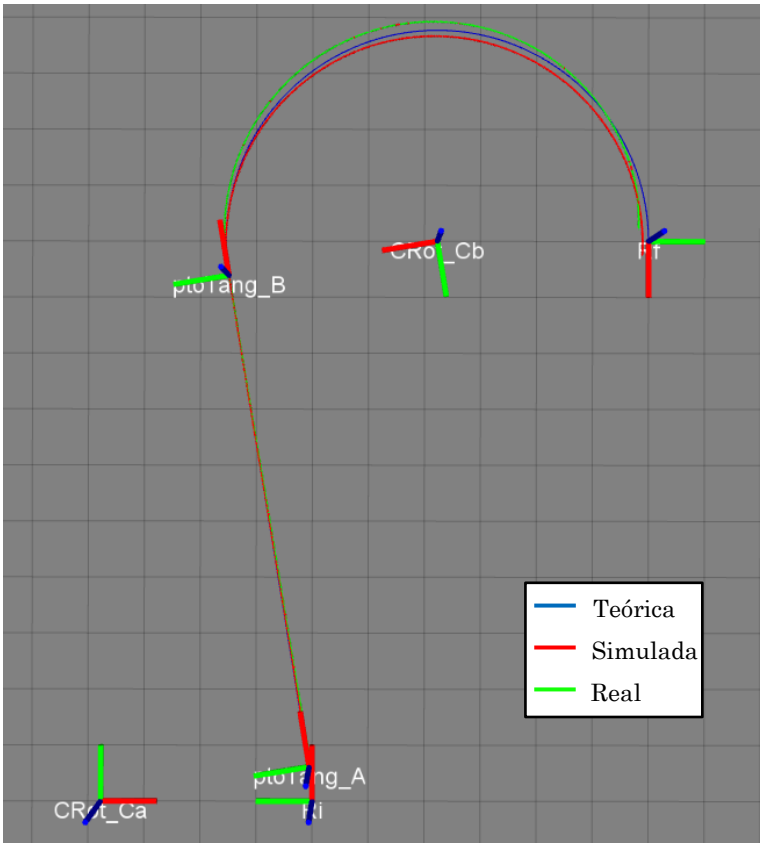


Figura A.6: Trayectoria de Dubins LSR hasta localización (10, -6, 180) .

En la Figura A.7 se observa un ejemplo del correcto funcionamiento del controlador en el tramo recto ya que al finalizar la primera trayectoria circular el error acumulado era notable y ha llegado a la localización B con un error prácticamente nulo.

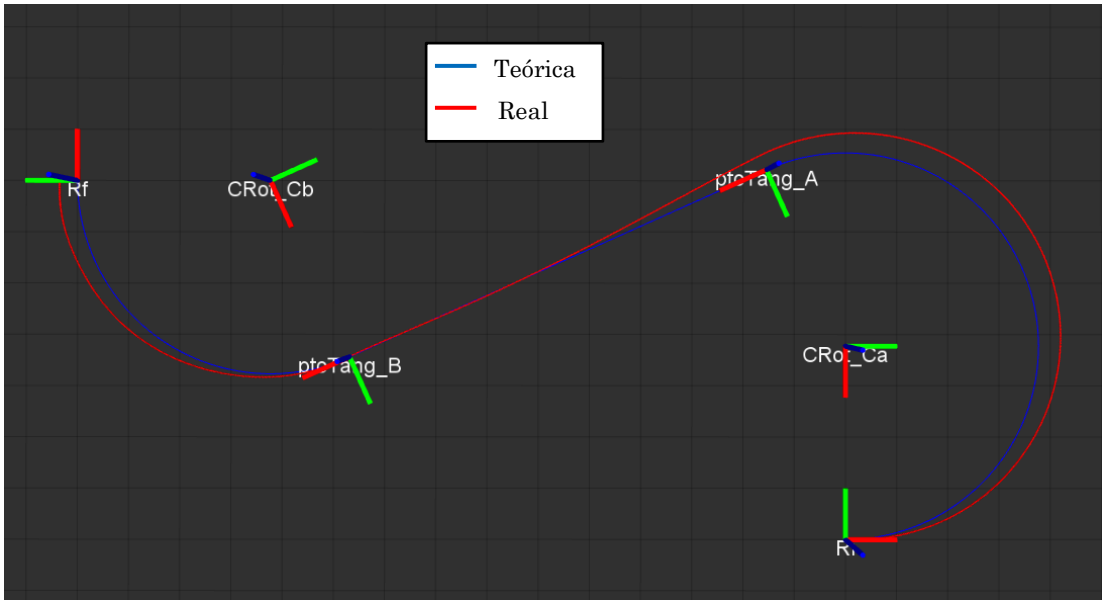


Figura A.7: Trayectoria de Dubins LSR hasta localización (-15, 7, 90) .

Finalmente en la Figura A.8 se muestra un ejemplo del error que comete el robot cuando se le introduce como parámetros de dimensiones del robot las reales.

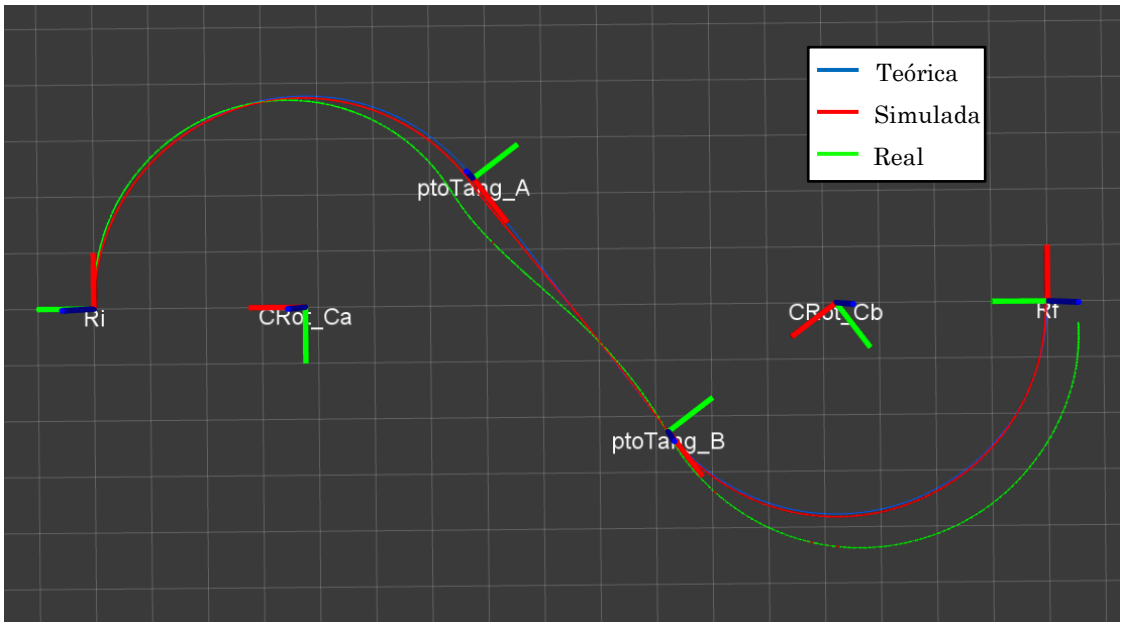


Figura A.8: Trayectoria de Dubins LSR hasta localización (0, -15, 0) .

Anexo B

Descripción software

El software utilizado para llevar a cabo los objetivos del presente trabajo, se ha basado principalmente en la implementación de los mismos en la plataforma de programación *ROS* [3], utilizando como lenguaje C++. También se ha utilizado el entorno de simulación Gazebo [4], que permite comprobar y corregir el correcto funcionamiento de los algoritmos necesarios para el cumplimiento de esos objetivos, además del visualizador 3D *RViz* [5] para poder observar la información captada por los sensores y la trayectoria seguida por el robot.

ROS

ROS, siglas de *Robot Operating System*, es una plataforma de programación, de software libre, que permite la comunicación entre distintos procesos, llamados nodos, mediante el intercambio de mensajes constituidos por determinados tipos de datos, cuya información varía según la función que desempeñe cada nodo. Para recibir o enviar los mensajes el nodo debe subscribirse o publicar respectivamente en un *topic*, que son los canales de comunicación entre nodos. En el caso de este trabajo, los nodos reciben información de algunos sensores del robot, como es la localización del robot o la medida de sus sensores de rango laser, y envían información al robot acerca de la velocidad lineal del robot y del ángulo giro de sus ruedas delanteras para modificar la dirección.

Gazebo

Gazebo es un programa que permite simular varios robots, sensores y objetos en un entorno tridimensional. Puede generar información realista para los sensores e incluye un simulador de dinámica de sólidos para simular comportamientos reales de cuerpos sólidos. De éste modo se realizó, en trabajos anteriores [6], un modelo del Robucar-TT de la Universidad de Zaragoza que simula un comportamiento parecido al del robot real,

incluyendo la información obtenida por sus sensores. Este modelo proporciona la posibilidad de ensayar el funcionamiento deseado de los programas diseñados, sin necesidad de aplicarlos directamente en el robot real, lo que habría supuesto un tiempo adicional innecesario, además de poder poner en riesgo la integridad de personas, y del propio robot, ante comportamientos no deseados del mismo.

Representación de gráficos de comunicación entre nodos

Para el caso de aplicación de las trayectorias de Dubins, en la Figura A.1 aparece un gráfico con los distintos nodos de *ROS* ejecutados y la comunicación entre los mismos en el entorno de simulación, donde se muestra la información que recibe y envía cada nodo y el nombre del *topic* por el que realiza la comunicación. Se muestran colorados en verde los *topics* que reciben información del nodo creado, y en azul los que proporcionan información a dicho nodo. Así entonces, se ve como en el nodo /*car,* que es el que se comunica con Gazebo para realizar las simulaciones, se está enviando la información de odometría al nodo /dubins_node implementado a través del *topic* /*car*/*odometry*, y está leyendo la información de velocidad y ángulo de giro de las ruedas direccionales, que le envía el nodo Dubins a través del topic /*car*/*in*. En el nodo /*dubins* se recibe la información que se quiere representar en el visualizador RViz.

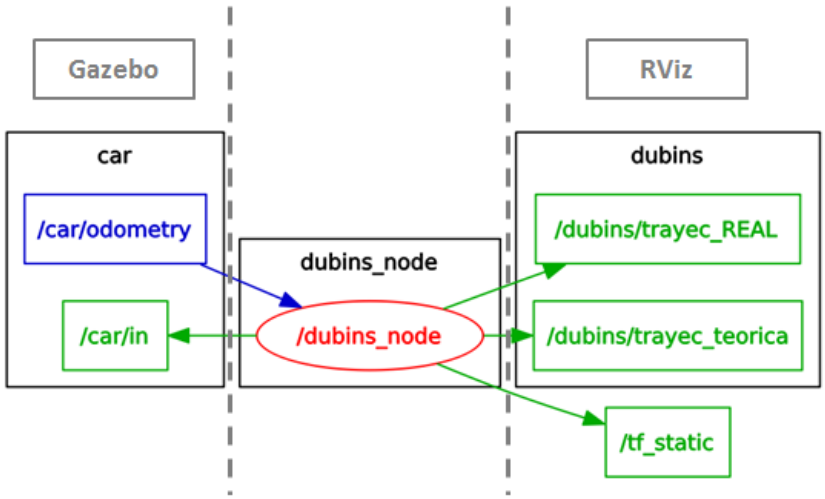


Figura B.1: Gráfico de comunicación de nodos ROS con Dubins_node .

En el caso ejecución del nodo para realizar el cambio de sentido, el gráfico de comunicación entre nodos representado en la Figura A.2 muestra que además de la odometría, el simulador le proporciona también la información recogida por los sensores laser, mientras que en el nodo correspondiente a la visualización de la información enviada por el nodo /maniobra_tunel_node, se visualiza la recta de regresión que representa la recta que forman los puntos del láser que están midiendo la distancia a la pared, y la trayectoria seguida por el robot.

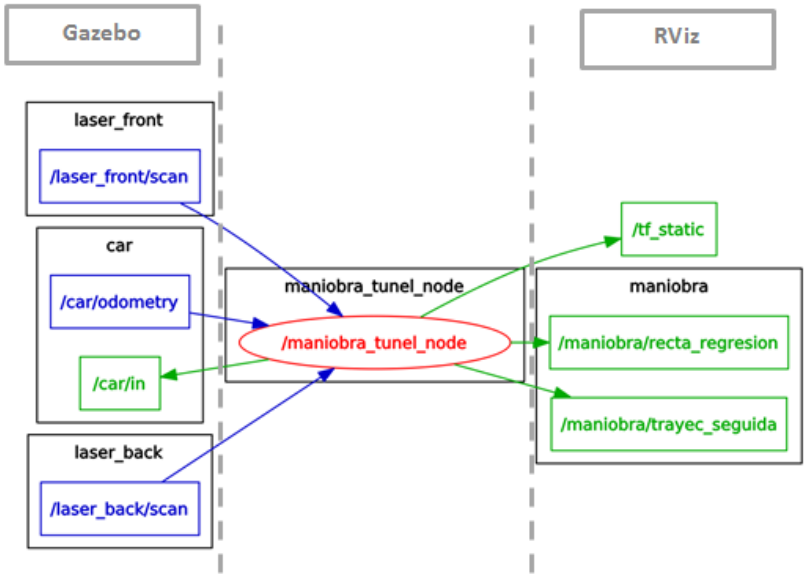


Figura B.2: Gráfico de comunicación de nodos ROS con Maniobra_tunel_node .