Proyecto Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Control y planificación de robots móviles

Autor: Paula Barroso Rodriguez
Tutor: Francisco Rodríguez Rubio

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Proyecto Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Control y planificación de robots móviles

Autor:
Paula Barroso Rodríguez

Tutor:
Francisco Rodríguez Rubio

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Proyecto Fin de Grado: Control y planificación de robots móviles

Autor: Paula Barroso Rodríguez

Tutor: Francisco Rodriguez Rubio

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis maestros

Resumen

El objetivo de este trabajo de Fin de Grado es la comparación de diversos algoritmos de cálculo de trayectorias para distintos robots móviles, de manera que sean capaces de garantizar una circulación sin colisiones a los puntos deseados. Para ello, se han realizado en MATLAB dichos algoritmos, simulándolos, analizándolos y comparando los resultados obtenidos.

Para dicho fin se ha estructurado el trabajo de la siguiente manera:

En primer lugar, se encentra una introducción donde se sitúa el trabajo en el contexto en el que nos encontramos, para así poder comprender porque es interesante su estudio a día de hoy.

Tras ello, se ve la necesidad de crear entornos donde poder generar las trayectorias deseadas. Por ello se ha creado un apartado donde se explica detalladamente los aspectos más relevantes acerca de la simulación de los mismos.

A continuación, se aporta un breve conocimiento sobre los distintos tipos de robots móviles que se usarán para realizar las trayectorias definidas por los distintos algoritmos calculados, junto con sus características principales.

Posteriormente, se introducen y explican los algoritmos usados y no usados para la generación de la trayectoria evitando los obstáculos.

Por último, se muestran los resultados de las simulaciones realizadas, se comparan los diferentes algoritmos que se han usado, y se comprueba cómo reaccionan los diferentes robots a los mismos. Finalizando con una muestra de las conclusiones obtenidas.

Para posibles trabajos futuros, se podría, partiendo de dichas trayectorias generadas, evitar obstáculos móviles que nos fuésemos encontrando gracias a la utilización de sensores.

Abstract

The objective of this Final Grade study is the comparison of various algorithms for calculating trajectories for different mobile robots, so that they are able to ensure movement without collision desired points. In order to that, it has been made in MATLAB these algorithms, simulating them, analyzing and comparing the results.

With that purpose, the work is structured as follows:

First, we find an introduction where we place the work in the context in which we find, in order to understand why it is interesting to study today.

After that, we feel the need to create environments where we can generate the desired trajectories. Therefore we have created a section where we will explain in detail the most relevant about its simulation.

Below, we provide a brief knowledge about the different types of mobile robots that will be used to make the trajectories defined by different algorithms calculated, along with their main characteristics.

Afterwards, we introduce and explain the algorithms used and not used for the generation of the path avoiding obstacles.

Finally, we show the results of the simulations, we compare the different algorithms that we used, and we will check how different robots react to them. Ending with a sample of the conclusions.

For possible future works, we could, based on said generated paths, avoid moving obstacles that we were finding us through the use of sensors.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

En primer lugar, se va a situar el contexto en el que surge el estudio de este trabajo de fin de grado y por qué resulta interesante.

En estos últimos años la robótica móvil está adquiriendo una gran importancia, ya sea en el entorno industrial como en el entorno doméstico o de transporte.

Los motivos son evidentes, por una lado un incremento del interés de la población por la robótica, por otro lado la producción en cadena y por consiguiente la reducción de los costos de producción de los mismos, trabajos de elevada precisión.

Muchos de los trabajos anteriormente citados necesitaban de un humano, pero recientemente la ciencia trata de proporcionar autonomía a dichos robots. En el caso específico de la robótica móvil, los robots han de desplazarse en su entorno, para ello es necesario que éste pueda ser caracterizado, identificar obstáculos y ubicarse con la mayor precisión posible con respecto a un sistema dado. Con estas habilidades el robot debe ser capaz de generar una trayectoria hacia su destino y tratar de seguirla lo más fielmente posible.

Debido a la gran variedad de robots móviles en la actualidad, se puede encontrar un amplio abanico de posibilidades en las que se pueden usar. De todas las aplicaciones para las que los robots móviles son utilizados hoy en día, se van a mencionar las siguientes:

- Vehículos industriales autoguiados:
  Los AGV (Automatic Guided Vehicle), representan un vehículo que se mueve de forma automática, es decir, sin conductor. Estos sistemas están concebidos para realizar tareas repetitivas y con alta cadencia, como el transporte de materiales. Tienen sus orígenes en 1953, cando se decidió realizar la idea de un camión remolque sin conductor. Este primer vehículo funcionaba gracias el campo magnético que creaba un cable enterrado en el suelo y era usado como guía. Hoy en día hay múltiples sistemas de guiado de dichos vehículos, siendo cada uno de ellos el más adecuado para un entorno concreto, entre dichos sistemas de guiado podemos encontrar el seguimiento a trayectorias previamente calculadas, tema del trabajo fin de grado.

Figura 1-1. AGV.

- Robots domésticos:
  Un robot doméstico, conocido también como robot de servicio, es un robot autónomo que se encarga de las tareas del hogar, por eso motivo entendemos que su función principal es la de mejorar la calidad de vida de las personas que habitan en dicho hogar. Debido al avance continuo de la robótica y la domótica, hemos podido ver cómo han ido irrumpiendo en los hogares varias clases de robots. Algunos de ellos se usan con el objetivo de facilitar tareas rutinarias, por otro lado, hay robots que se emplean para el ocio o entretenimiento.

A pesar de que inicialmente los robots domésticos que se conocían eran robots de limpieza y de jardinería, hoy en día podemos encontrar un abanico mucho más amplio, de los que podemos destacar a parte de los mencionados anteriormente, robots para alimentar mascotas, robots sociales y robots chefs entre otros.

Figura 1-2. Robot de limpieza.

Figura 1-3. Robot cortacésped.

Figura 1-4. Robot social.

- Robots de reparto:
  Las grandes empresas de compra online llevan tiempo tratando de diseñar nuevas formas para su servicio de reparto. Inicialmente optaron por el uso de drones, pero debido a una gran cantidad de problemas con las leyes vigentes, muchas empresas como Amazon o Media Mark, incluso el correo en Suiza, han decidido optar por un transporte terrestre. En concreto un robot de seis ruedas que recorrerá las calles en busca de la casa del cliente. Su funcionamiento se realizará mediante una combinación de un sistema de navegación, para determinar las posiciones, y de un sistema de reconocimiento visual del entorno, mediante cámaras. En caso de que el robot posea alguna duda o que algún lugar sea especialmente difícil, un operador tomará el control del robot de manera remota a distancia. Esto que inicialmente puede parecer un fallo no nos resulta tan grave, ya que, quedará almacenado en el robot aumentando su autonomía.

Figura 1-5. Robot de reparto.

- Robots de exploración:
  Debido a la dificultad o peligrosidad de algunas situaciones, durante muchos años se ha usado la robótica para evitar a los humanos correr ciertos riesgos. Los robots de exploración se han usado en muchos ámbitos, tanto como para inspeccionar lugares que al ser humano le sería muy complejo acceder (otros planetas, zonas submarinas), como para labores de salvamento (detección de explosivos, ya sean minas o cualquier otro tipo, robots contraincendios).

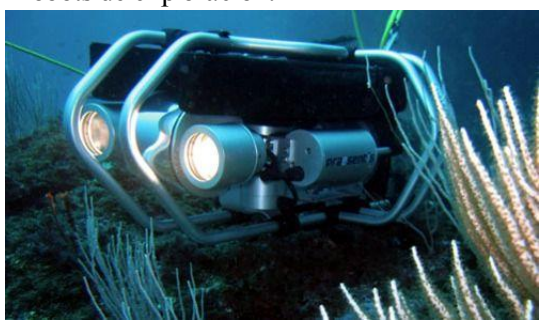


Figura 1-6. Robot submarino.

En todos estos ejemplos se puede ver una característica común. Todas ellas son situaciones en las que un vehículo tiene la tarea de realizar una trayectoria desde un punto origen a un punto destino sin colisionar con su entorno.

Para poder conseguir el propósito anterior se encuentra la dificultad de la navegación, la cual implica resolver

subproblemas a nivel de percepción, localización, planificación y control de movimientos. De forma que se pueda ser capaz de interpretar los datos que suministran los sensores para así extraer información útil, poder determinar su posición en el entorno, decidir cómo actuar para alcanzar el objetivo y gestionar sus actuadores para conseguir la trayectoria deseada.

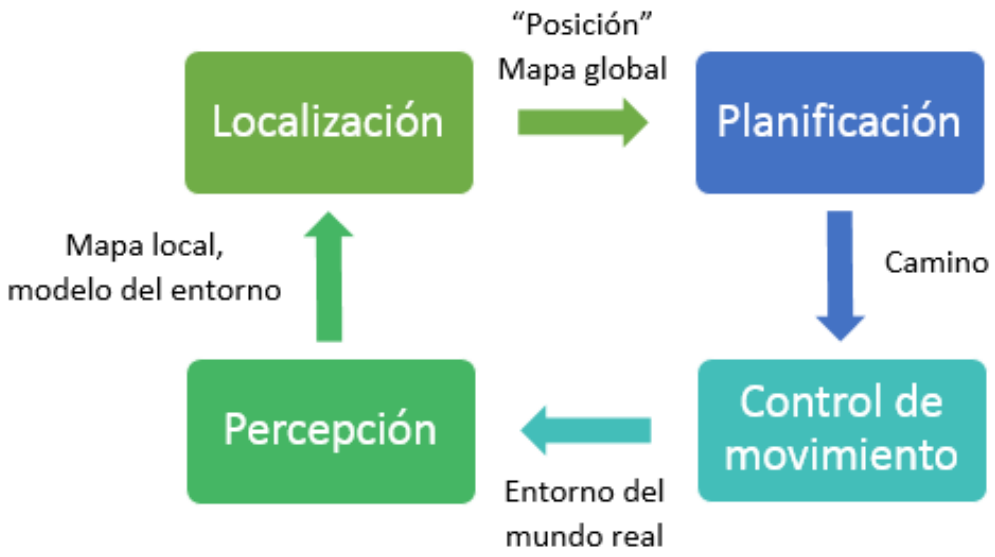


Figura 1-7. Problema de la navegación.

El objetivo de este trabajo es analizar y comparar los distintos algoritmos de generación de trayectorias aplicados a diferentes tipos de robots móviles, para así poder ver tanto las características principales de cada uno de ellos como el funcionamiento en distintos tipos de robots, y así conocer los requisitos mínimos que cada uno de ellos deben tener para poderlos llevar a cabo.

2 MAPAS

En este apartado trataremos de mostrar uno de los puntos más importantes del proyecto, es decir, la generación de los entornos. Durante la realización del trabajo nos hemos centrado en el seguimiento de trayectorias por diferentes tipos de robots, las cuales hemos generado mediante diferentes algoritmos. Pero para poder generar dichas trayectorias era necesario incluir al robot en un entorno, conociendo el punto en el que se encuentra y al que quiere llegar. Además tenemos que tener en cuenta que muchos de los algoritmos que hemos empleado necesitan que se les introduzca un mapa para así posteriormente generar la trayectoria.

Para la creación de entornos hemos dividido los problemas mostrados anteriormente en dos.

El primero de ellos es la generación del entorno propiamente dicho, es decir, en este punto nos hemos encargado de generar una matriz de 100x100 (ya que será el tamaño que tendrá nuestro entorno) compuesta por unos y ceros, en la que los ceros significan celda libre y los unos celda ocupada. La creación de la matriz se ha realizado de forma totalmente aleatoria, de manera que lo primero que decidíamos al azar era cuantos obstáculos tendría el mapa, los cuales pueden variar entre 0 y 9. Cuando ya teníamos el número de obstáculos, necesitábamos conocer cuáles serían sus medidas y que celdas ocuparían, realizando este proceso al igual que el anterior de forma aleatoria, siendo el intervalo de celdas posibles tanto en el eje x como en el eje y de 2 a 99. Junto con la creación de la matriz este problema también incluye el posicionamiento inicial del robot en el mapa y la creación del punto de destino. Los cuales son semialeatorios, es decir, en primer lugar obtenemos su posición en el mapa al azar, tras esto comprobamos si dicha posición corresponde con una celda libre, si no es así volveríamos a calcular su posición y así sucesivamente hasta que esta coincida con una celda libre.

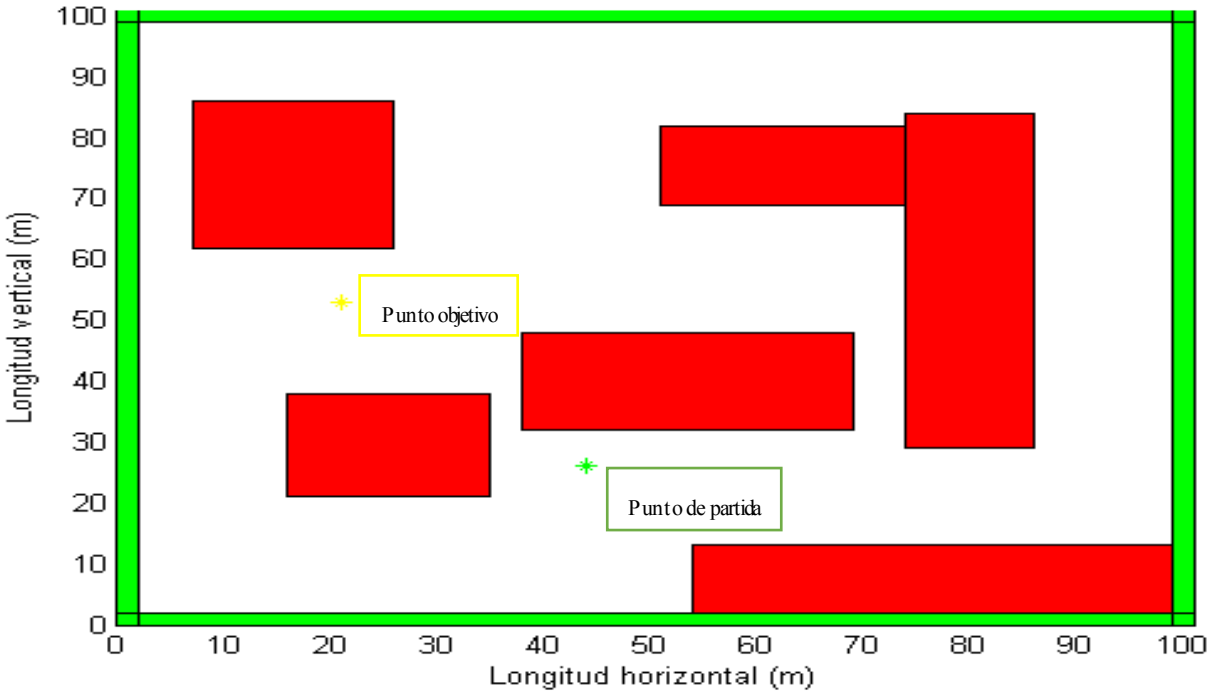


Figura 2-1. Mapa aleatorio 1.

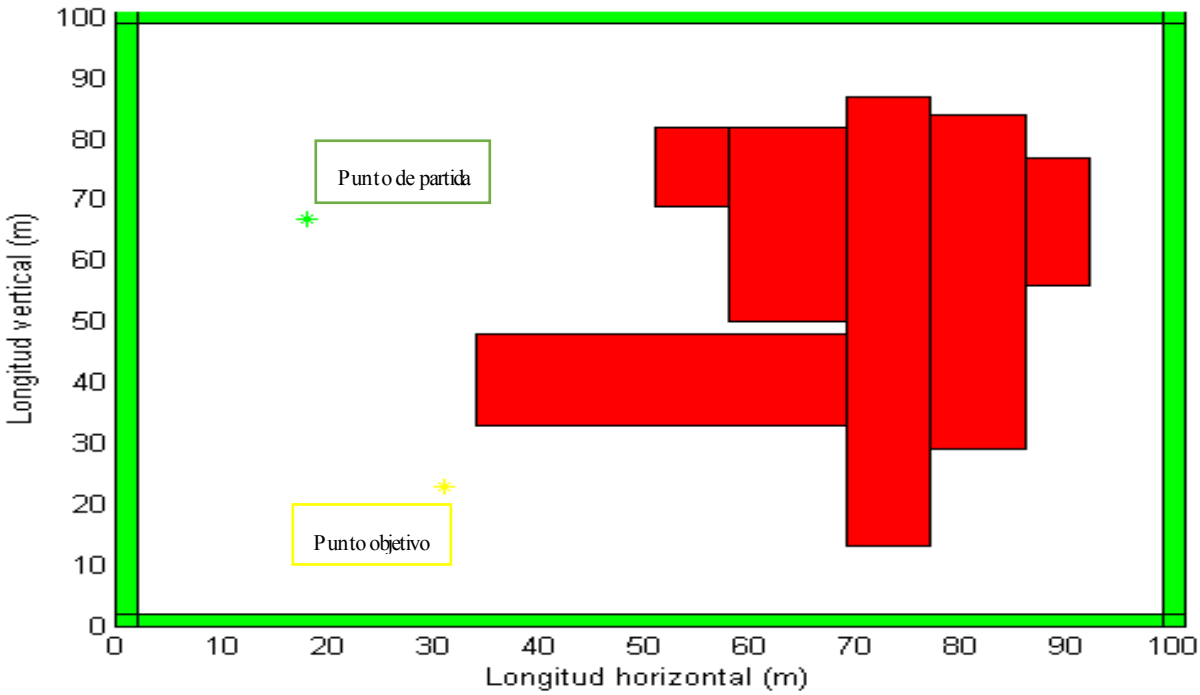


Figura 2-2. Mapa aleatorio 2.

Como podemos ver las imágenes que hemos mostrado anteriormente son muy diferentes, esto es debido a la aleatoriedad en la generación de los mapas. Por ese motivo todos los mapas generados no son óptimos, por lo que hay veces que no merece la pena usarlos. Si observamos ambas imágenes podemos ver como el mapa aleatorio uno es un mapa totalmente funcional, mientras que en el mapa aleatorio número dos vemos como todos los obstáculos se encuentras a la derecha, dejando todo el espacio de la izquierda libre de obstáculos, siendo poco valido para poder comprobar el funcionamiento de los distintos algoritmos y robots que queremos usar.

El segundo de los problemas es el de detectar cuales con los vértices de estos obstáculos para así poder mostrar el entorno por pantalla y entregarle dicha información junto con la matriz de 100x100 a los programas que lo necesiten, consiguiendo así que dichos programas posean una visión más completa del entorno. Los vértices serán entregados mediante una matriz que consta de cuatro columnas, una para cada tipo de vértice. Para clasificar los vértices hemos recorrido la matriz celda por celda comprobando si la celda en la que nos encontramos es un vértice o no. Si se corresponde con un vértice lo almacenamos en la matriz creada en la columna correspondiente con el número de vértice al que pertenezca. De esta forma tenemos todos los vértices almacenados en una matriz separados según el tipo de vértice que sean.

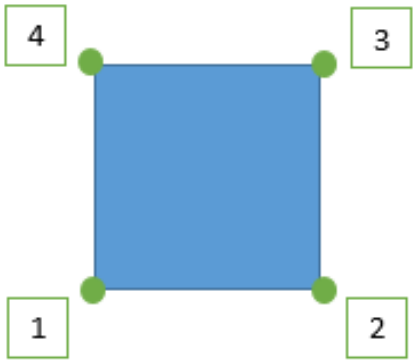


Figura 2-3. Vértices de los obstáculos.

Por último para poder realizar la comparación de los diferentes algoritmos y robots, fue necesario crear dos mapas fijos donde fuese el algoritmo o el robot que fuese el entorno no cambiase y dichos mapas fuesen óptimos,

pudiendo apreciar con mayor facilidad las deficiencias y las virtudes de los elementos implementados.

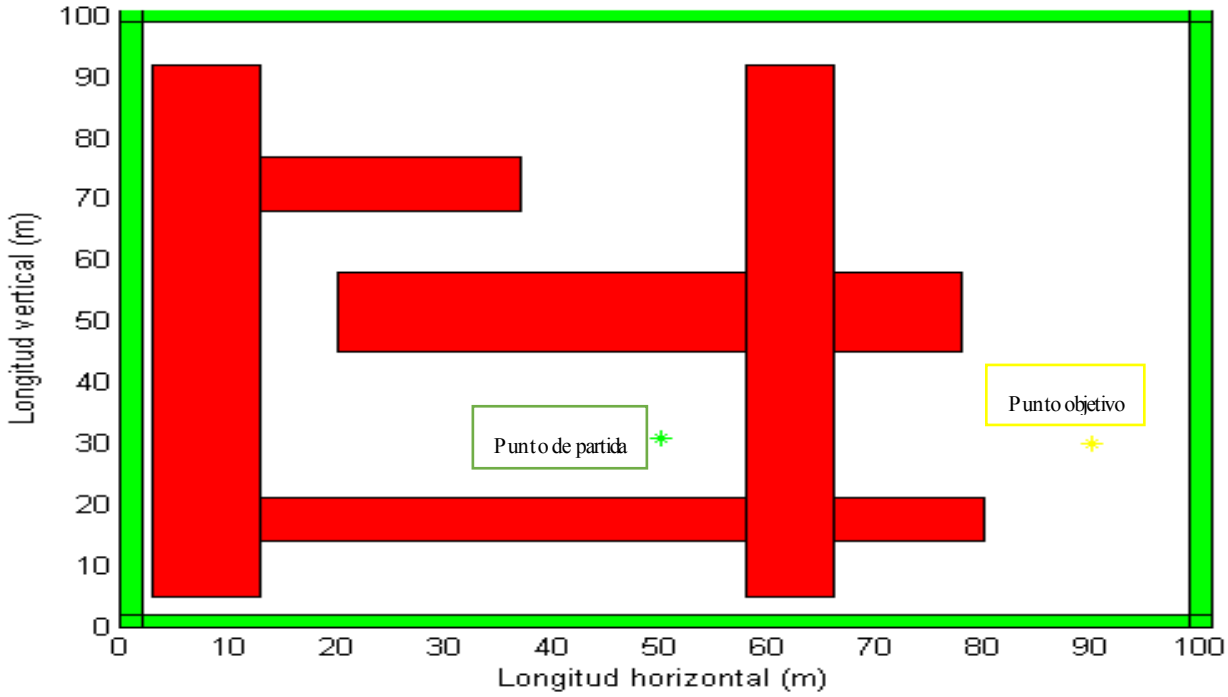


Figura 2-4. Mapa no aleatorio 1.

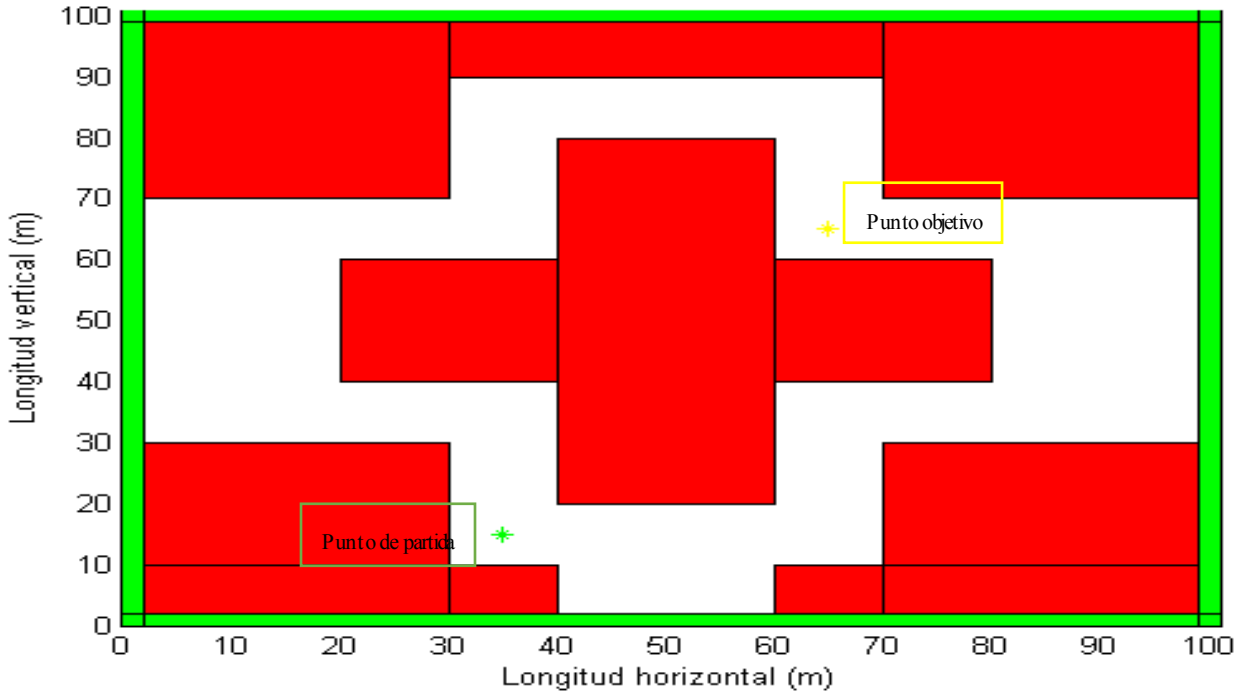


Figura 2-5. Mapa no aleatorio 2.

3 SEGUIMIENTO DE TRAYECTORIAS

Ya que no solo queremos obtener las trayectorias deseadas sino que también queremos ver la respuesta de los robots a dichas trayectorias, hemos tomado cuatro tipos distintos de robots para ver las diferentes reacciones que estos muestran ante las mismas. Para poder observar como dichos robot siguen la trayectoria generada es necesario conocer la cinemática directa de cada uno de ellos y posteriormente enlazarla con el algoritmo de seguimiento de trayectoria al que se le introducirá la trayectoria deseada ya calculada previamente.

En primer lugar, mostraremos los modelos cinemáticos de cada uno de los robots. Tras ello, explicaremos el algoritmo de seguimiento a trayectoria. Finalmente realizaremos la unión entre los dos puntos anteriormente mencionados y aclararemos la forma en la que hemos realizado su implementación en Matlab.

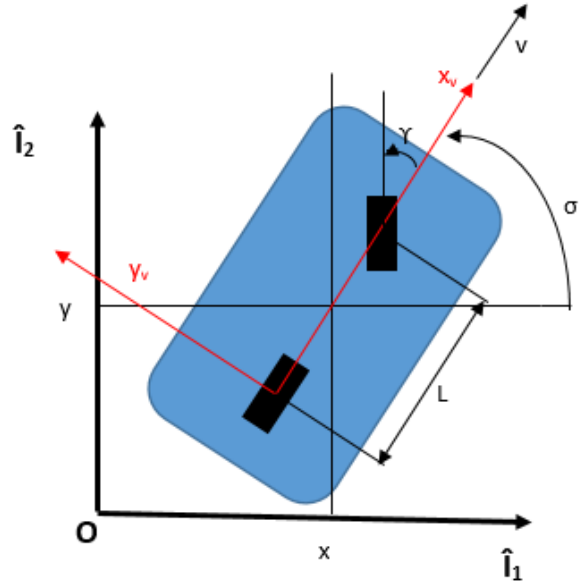
Para cada uno de los robots se ha tomado el radio de la rueda (R) 30cm, distancia de las ruedas que están en paralelo (b) 30cm y distancia de las ruedas de adelante y atrás (L) 30cm. Midiendo el ancho total del robot 40cm. Estas medidas podrían variarse cambiando en la función de Matlab de la cinemática directa de cada robot dichos valores.

3.1 Modelos cinemáticos

En este apartado mostraremos los modelos cinemáticos de cada uno de los robots que usaremos, gracias a los cuales podremos posteriormente simular y observar las diferencias existentes entre ellos.

3.1.1 Biciclo

Este tipo de robot se caracteriza por poseer dos ruedas consecutivas de las cuales únicamente la delantera tiene la capacidad de girar, dicha disposición tiene un inconveniente, es necesario una velocidad mínima y un ángulo de giro de la rueda máximo para que este mantenga el equilibrio y no se caiga, además de esto podemos comprobar que el vector velocidad va en la misma dirección que la orientación del robot. Presentando la siguiente esquema y ecuaciones de cinemática directa.

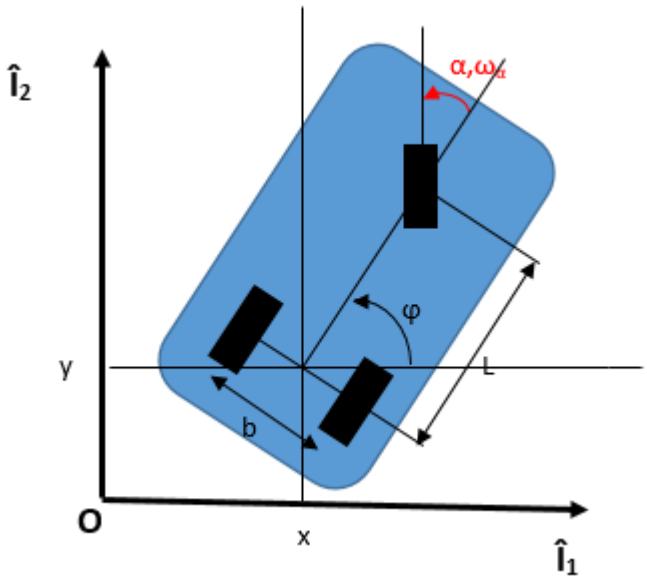


$$\dot{x} = v\cos(\sigma)$$
$$\dot{y} = v\,sin(\sigma)$$
$$\dot{\sigma} = \frac{v}{L}\tan(\Upsilon)$$

Figura 3-1. Robot biciclo.

3.1.2 Triciclo

En este robot observamos que posee tres ruedas, dos traseras paralelas fijas y otra delantera con posibilidad de realizar giros, está disposición a pesar de tener una cinemática directa algo más compleja que en el biciclo tiene una gran ventaja frente a este, y es que la disposición de ruedas hace mucho más dificultoso que el robot se caiga. A continuación mostraremos su esquema y modelo cinemático directo.



$$\dot{x} = R\cos(\alpha)\cos(\varphi)\,\omega_t$$
$$\dot{y} = R\,cos(\alpha)\sin(\varphi)\,\omega_t$$
$$\dot{\varphi} = \frac{R\sin(\alpha)}{L}\,\omega_t$$

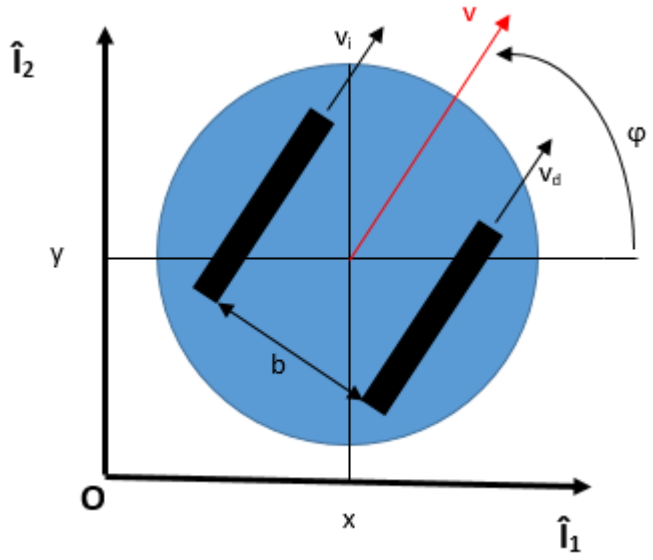
Siendo:

$$\alpha = \Upsilon$$
$$\omega_t = \frac{v}{R\cos(\alpha)}$$
$$\omega_a = \frac{d\alpha}{dt}$$

Figura 3-2. Robot triciclo.

3.1.3 Diferencial

En estos robots vemos como únicamente encontramos dos ruedas, en este caso paralelas, con la característica de que ambas son fijas y cada una de ellas tiene una velocidad independiente. Gracias a esto podemos conseguir su maniobrabilidad. Podemos ver como el vector velocidad del conjunto será siempre una composición de ambos vectores velocidad. El esquema que presenta y su cinemática directa es la siguiente.



$$\dot{x} = \frac{R}{2}\cos(\varphi)\ \omega_i + \frac{R}{2}\cos(\varphi)\ \omega_d$$
$$\dot{y} = \frac{R}{2}sen(\varphi)\ \omega_i + \frac{R}{2}\sin(\varphi)\ \omega_d$$
$$\dot{\varphi} = -\frac{R}{b}\ \omega_i + \frac{R}{b}\ \omega_d$$

Siendo:

$$v = \frac{R(v_i + v_i)}{2}$$
$$\Upsilon = \frac{R(v_d - v_i)}{b}$$

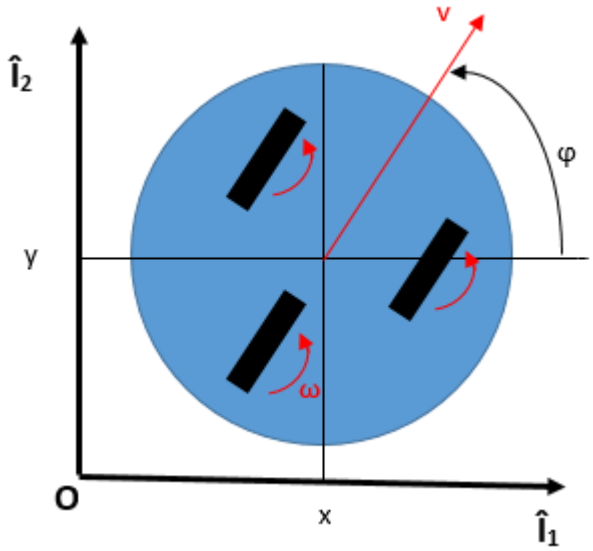
De modo que:

$$\omega_d = \frac{2v + \Upsilon b}{2R^2}$$
$$\omega_i = \frac{2v}{R^2} - \omega_d$$

Figura 3-3. Robot diferencial.

3.1.4 Síncrono

Los robots con esta configuración presentan tres ruedas, dos paralelas y una un poco más atrasada, una característica de los robots síncronos que los diferencian del resto es que todas sus ruedas poseen la capacidad de girar, aunque no libremente, es decir, todas las ruedas giran el mismo ángulo en el mismo instante y con la misma velocidad. En este tipo de robots su vector velocidad posee la misma dirección que la orientación de las ruedas. Su esquema y ecuaciones de cinemática directa son los siguientes.



$$\dot{x} = R\cos(\varphi)\ \dot{\theta}$$
$$\dot{y} = R\ sin(\varphi)\ \dot{\theta}$$
$$\dot{\varphi} = \omega$$

Siendo:

$$\omega = \Upsilon$$
$$\dot{\theta} = \frac{v}{R}$$

Figura 3-4. Robot síncrono.

3.1.5 Implementación en Simulink

Los modelos anteriormente definidos quedan implementados en Simulink de la siguiente forma, siendo el modelo mostrado abajo el del biciclo:

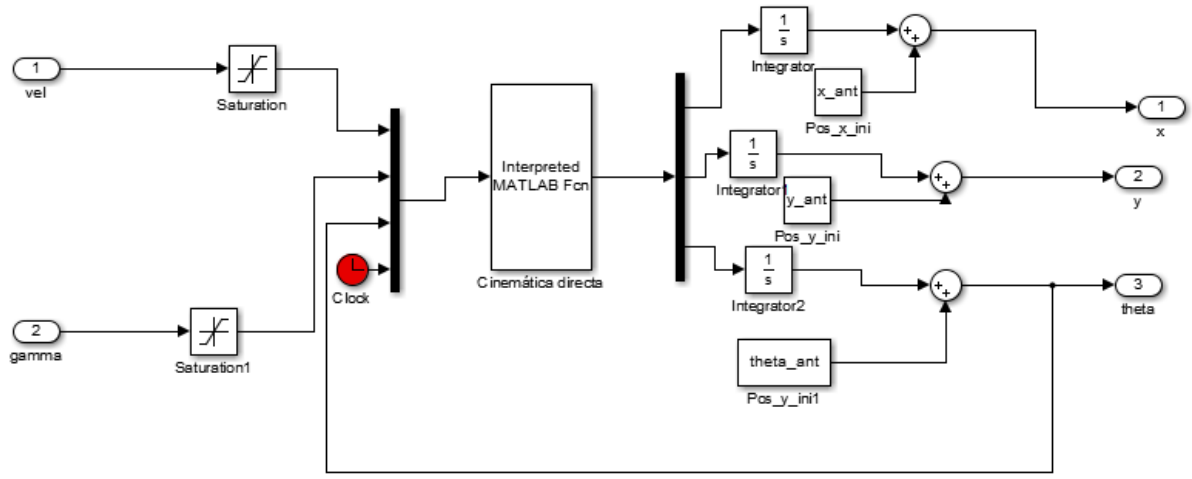


Figura 3-5. Modelo cinemático directo en Simulink.

En la imagen anterior podemos ver un bloque central, que se corresponde con una función de Matlab, en la que encontramos las ecuaciones de la cinemática directa descritas anteriormente, las cuales son diferentes para cada tipo de robot como pudimos observar al inicio de este apartado. También podemos ver como las variables de entrada exterior son la velocidad y la gamma. Además encontramos como estas variables de entrada están saturadas, tanto su valor mínimo como su valor máximo, en el caso de la velocidad a 2 y 0, y en el caso de la gamma a $\pi/4$ y $-\pi/4$. Esto es debido a que como hemos dicho anteriormente los robots no funcionan con cualquier parámetro, si la velocidad es muy alta, este podría caerse al igual que si el ángulo de giro es muy grande. En las formulas de la cinemática directa mostradas anteriormente encontrábamos que las variables x, y, y theta (en el caso del biciclo) no las conocíamos directamente sino que podíamos ser capaces de obtener su derivada. Por ese motivo los valores que obtenemos del bloque central de la cinemática directa son integrados obteniendo así dichas variables, a las que le sumamos las posiciones de la que parten los robot en el momento en el que se llama al archivo de Simulink. Por último vemos que a pesar de que theta no es un valor de entrada exterior, es necesario realimentarlo ya que también es un valor de entrada, el cual es necesario para nuestras ecuaciones cinemáticas.

3.2 Algoritmo de seguimiento de trayectoria

En segundo lugar tenemos que tener en cuenta que la idea es que nuestro robot siga el camino que hemos calculado mediante los algoritmos de generación de trayectorias. Para ello hemos usado el algoritmo de seguimiento de trayectoria, en este algoritmo le vamos introduciendo puntos de destino a medida que el robot los va alcanzando, estos van variando, de manera que los puntos objetivo van cambiando a lo largo del tiempo. Este tipo de algoritmo es de seguimiento puro y en el podemos elegir la distancia a la que queremos seguir la trayectoria (d* que en nuestro caso es cero).

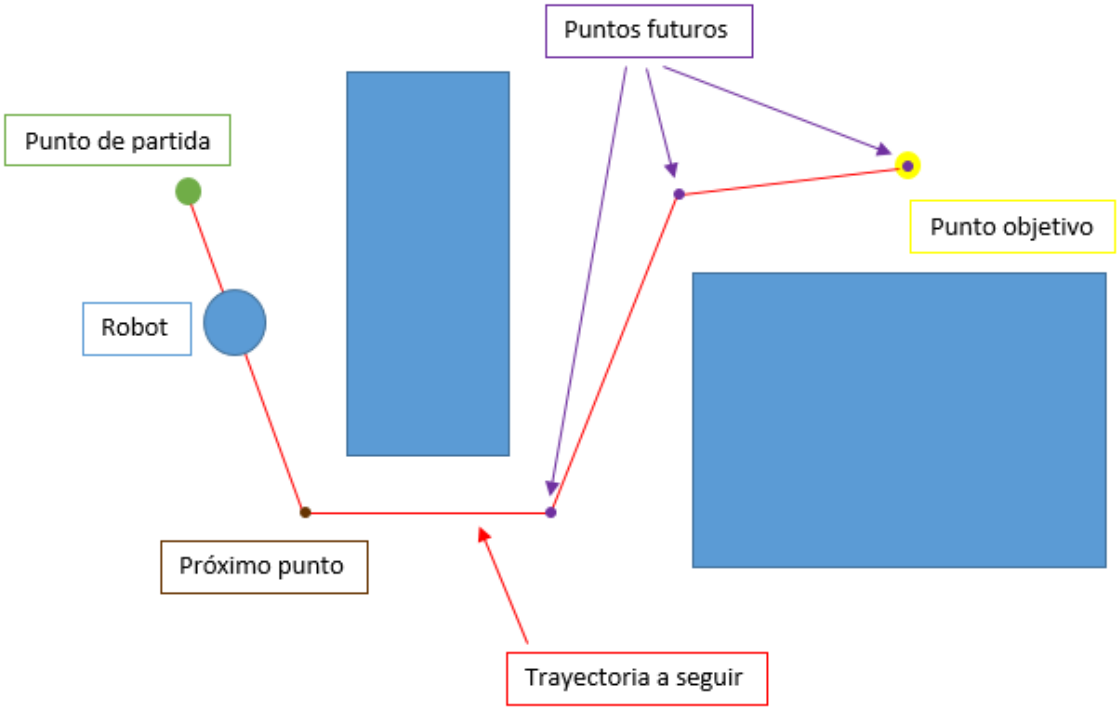


Figura 3-6. Algoritmo de seguimiento de trayectoria.

En este tipo de algoritmos para obtener la velocidad a la que debe moverse nuestro robot es necesario usar una ley de control proporcional más integral (PI), mientras que para calcular la orientación solo es necesario una ley de control proporcional (P).

$$v = K_v e + K_i \int e dt$$

$$\Upsilon = K_h(\theta^* - \theta), \quad siendo\ K_h > 0$$

Donde:

$$e = \sqrt{(x^* - x)^2 + (y^* - y)^2} - d^*$$

$$\theta^* = \tan^{-1}\frac{y^* - y}{x^* - x}$$

Siendo x* e y* el punto objetivo en cada instante, x e y el punto donde nos encontramos en dicho momento, θ* el ángulo relativo del vehículo al objetivo y K_h,K_i y K_y son constantes de posición, integración y velocidad respectivamente.

Dichas constantes las iremos definiendo con unos valores u otros según el método de generación de trayectoria y el robot usado.

La implementación de dicho algoritmo en Simulink sería la siguiente:

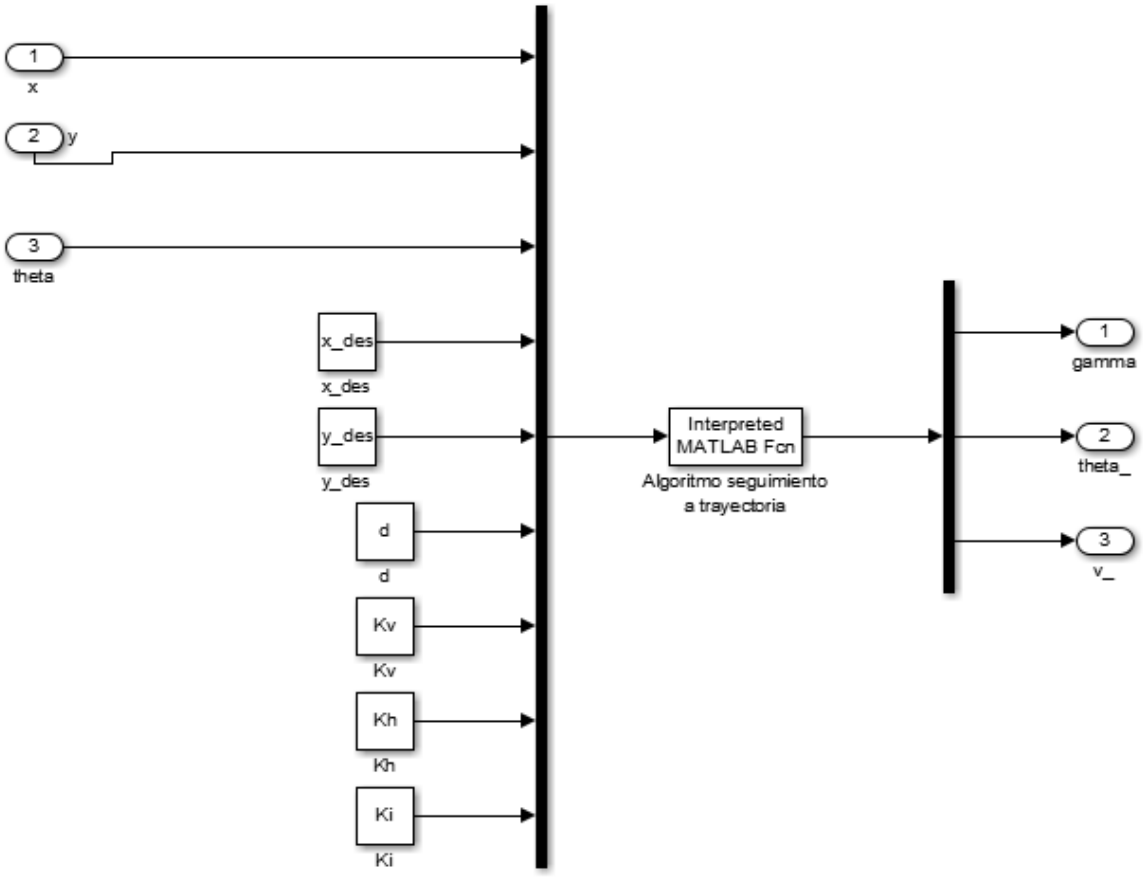


Figura 3-7. Algoritmo de seguimiento de trayectoria en Simulink.

Podemos ver como tenemos nueve valores de entrada, de ellos seis son constantes y tres son variables. Las cuatro constantes finales (d,Kv,Kh,Ki), variarán como hemos dicho antes según el robot y el algoritmo usado y serán introducidas externamente, mientras que las otras dos, también introducidas externamente, se corresponden con el punto que queremos alcanzar en cada instante y estas irán cambiando a medida que se vayan alcanzando los puntos de la trayectoria. Dicho algoritmo lo hemos logrado comparando el punto en el que nos encontramos, el cual es conocido ya que tenemos sus coordenadas x e y, con el punto objetivo de ese instante. En el momento que coincidan dichos valores se pasa al siguiente punto objetivo. También encontramos en la parte central una función de Matlab en la que se realizan las operaciones del algoritmo de seguimiento de trayectoria mostradas anteriormente, para así conseguir las variables de salida necesarias.

3.3 Acoplamiento de los dos apartados anteriores

Por último tan solo nos quedaría unir los dos puntos mencionados anteriormente, la cinemática directa de cada robot y el algoritmo se seguimiento de trayectoria. Para así lograr que cada tipo distinto de robot sea capaz de seguir la trayectoria deseada. Tras ello aclararemos el modo en el que se llamará a dicha función de Simulink y como se le pasarán las variables necesarias. Siendo el esquema usado en Simulink el que mostramos a continuación para el robot diferencial:

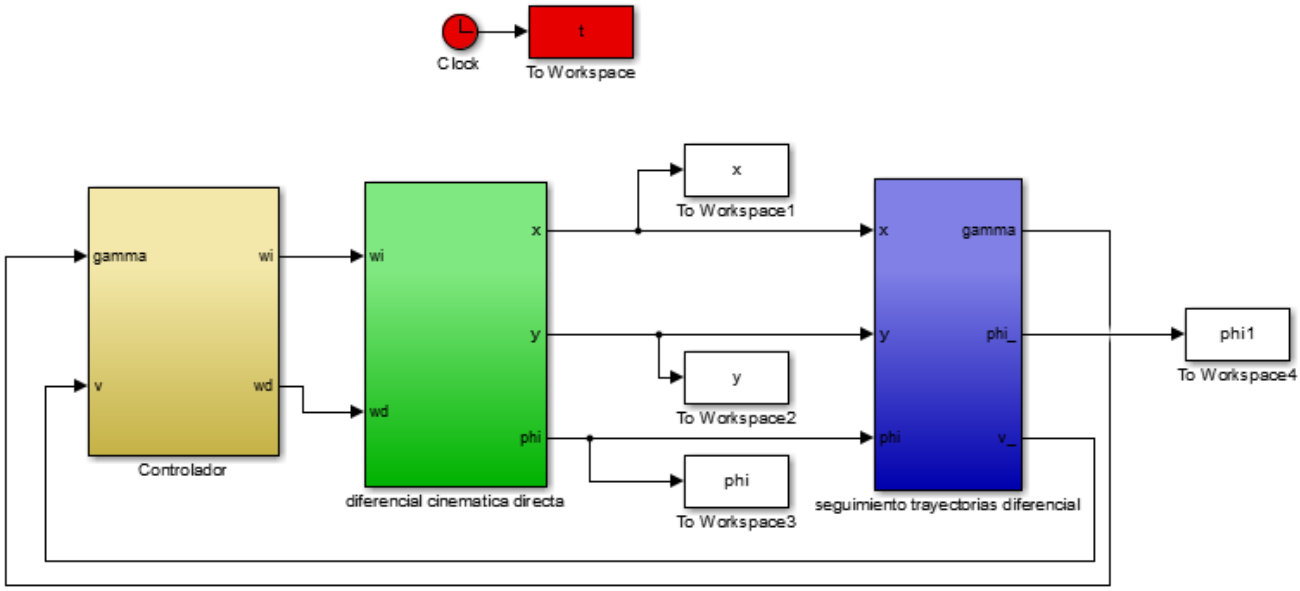


Figura 3-8. Seguimiento de trayectoria de robot específico en Simulink.

En este esquema vemos los dos bloques mencionados anteriormente, el verde es el modelo cinemático directo y el azul el algoritmo se seguimiento de trayectorias. Además de estos, vemos un tercer bloque llamado controlador de color amarillo, el cual se encarga de convertir las unidades de salida del bloque de seguimiento de trayectorias a las unidades de entrada de la cinemática directa. En este caso el esquema se corresponde con el robot diferencial, hemos elegido este robot ya que en él encontramos los tres bloques mencionados, a diferencia de en el robot biciclo en el que debido a que las entradas de la cinemática directa son las de salida del seguimiento de trayectoria no le hace falta un controlador que realice un cambio de variables. En la imagen podemos ver como las posiciones x e y que salen del bloque del modelo cinemático son las que entran en el seguimiento a trayectoria, junto con el ángulo girado phi. Por otro lado vemos como los valores de entrada del controlador, ángulo de giro de la rueda (gamma) y velocidad, son los valores de salida del seguimiento de trayectorias. Por último comprobamos como las variables wi y wd de salida del controlador son las que entran en el bloque de cinemática directa. De esta forma los tres bloques quedan completamente conectados, siguiendo cada uno de los cuatro robots elegidos la trayectoria deseada.

Es de mencionar que para poder llevar a cabo esta unión de forma apropiada es necesario combinar Matlab con Simulink, de forma que Matlab sea el que le pasa tanto la x_ant, la y_ant, la theta_ant del modelo cinemático directo, como la x_des y la y_des del seguimiento de trayectorias. A modo de aclaración de como se ha realizado dicho procedimiento, en el capítulo 5 se muestra un diagrama con una explicación del mismo.

4 GENERACIÓN DE TRAYECTORIAS

A continuación pasaremos a explicar una serie de algoritmos, los cuales son comúnmente usados para el cálculo de trayectorias. Estos algoritmos los diferenciaremos en los que hemos usado poder obtener nuestro fin, la obtención de una trayectoria y la evitación de obstáculos, y los que no hemos implementado.

4.1 Algoritmos implementados

A continuación mostraremos una serie de algoritmos que usaremos para poder obtener nuestro fin, la obtención de una trayectoria evitando los obstáculos. En este punto llevaremos a cabo una presentación de los mismos, mientras que más adelante mostraremos los resultados de sus ejecuciones con los diferentes tipos de robots móviles empleados.

4.1.1 Grafos de visibilidad

Los grafos de visibilidad proporcionan un enfoque geométrico útil para resolver los problemas de planificación. Este tipo de algoritmo modela los obstáculos mediante polígonos partiendo de que toma un entorno bidimensional. En este tipo de algoritmo para generar un grafo se introduce el concepto de visibilidad, según el cual se definen los puntos de su entorno como visibles si se pueden unir mediante un segmento rectilíneo que no colisione con ningún obstáculo.

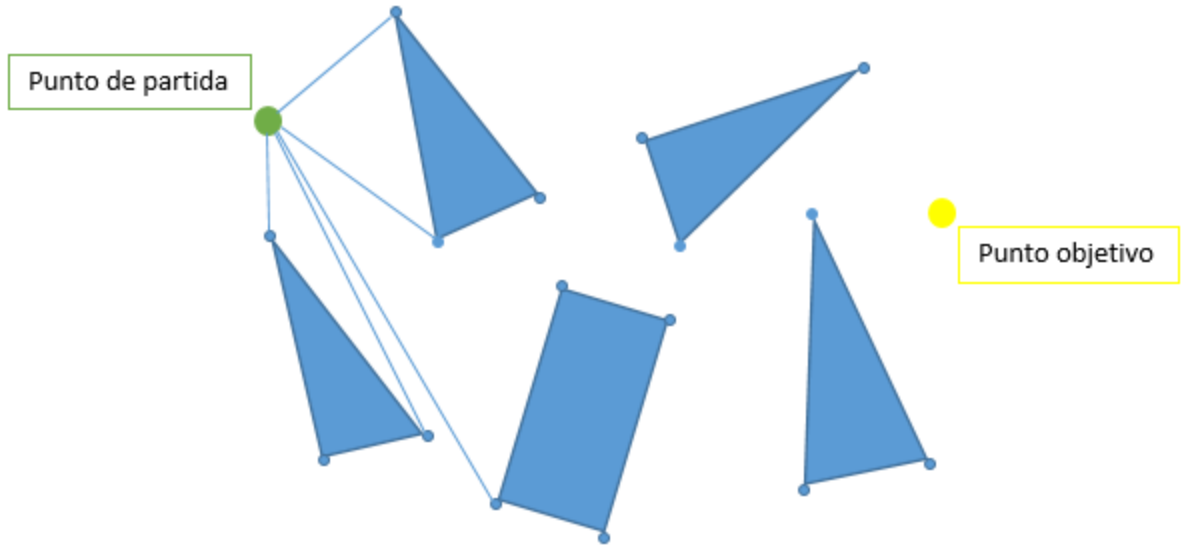


Figura 4-1. Concepto de visibilidad.

Así se considera como nodos del grado de visibilidad la posición inicial, la final y todos los vértices de los obstáculos de su entorno, el grafo resultante será la unión de todos los nodos que sean visibles mediante arcos.

Por último gracias a un algoritmo de búsqueda en grafos elegimos la ruta que una la posición inicial con la final, y que esta sea la más apropiada según el criterio elegido por el programador, normalmente suele ser el de mínimo

número de saltos.

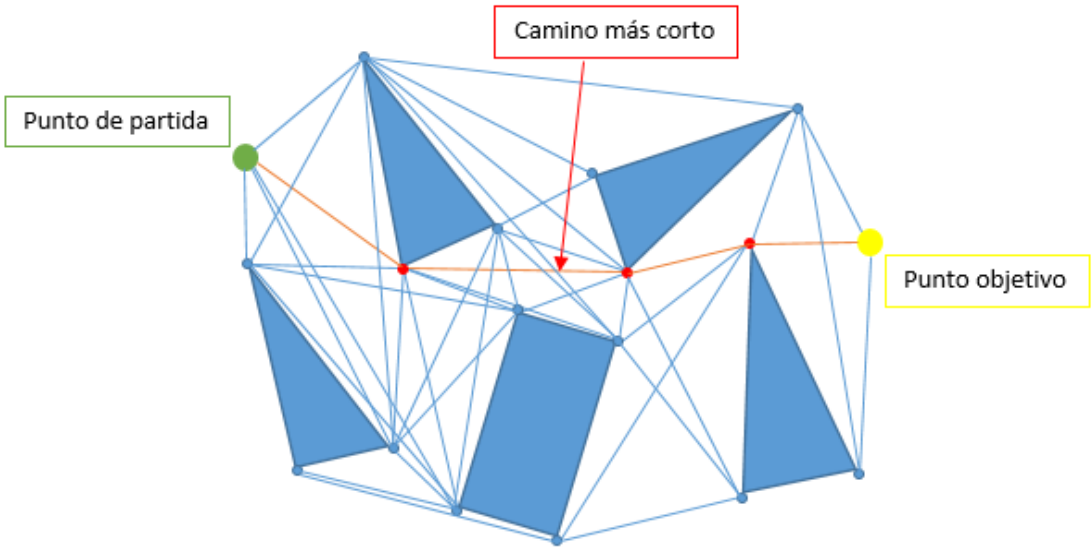


Figura 4-2. Grafo de visibilidad.

En el grafo mostrado podemos observar cómo solo están unidos los nodos visibles, de forma que el conjunto de arcos estará formado por las aristas de los obstáculos junto con el resto de líneas que relacionan los vértices de los polígonos.

Tras la selección del camino es necesario alejar al robot de los obstáculos, ya que sino nuestro robot colisionaría debido a que el camino calculado mediante este algoritmo es el formado por los vértices y las aristas de nuestros obstáculos, en mi caso hemos separado el robot de la trayectoria 3 metros de esta forma como nuestro robot mide 30cm no colisionará.

El coste computacional de este algoritmo se eleva de forma importante con el número de obstáculos que encontremos en nuestro mapa, es decir, con el número de vértices.

4.1.2 Descomposición en celdas

Los métodos de la descomposición por celdas se caracterizan por dividir el problema en dos niveles: Uno global que resuelve la trayectoria por zonas y otro a nivel local. Consiste en descomponer el espacio libre en regiones más pequeñas, a las que llamamos celdas. Una vez hayamos discretizado el espacio de configuraciones libres de colisión en un conjunto de celdas completo, se procede a resolver su conectividad a nivel superior, de tal forma que un camino entre dos configuraciones en una misma celda se va a dar mediante un grafo de conectividad, de forma que el camino estará representado por la sucesión de celdas por las que se tiene que pasar para llegar a la configuración final. Existen dos subtipos principales:

- Descomposición exacta: Usamos este término cuando la descomposición genera un conjunto de celdas cuya unión resulta idéntica al espacio de configuraciones libres de colisión. Es decir, las celdas deben estar definidas de tal manera que se adapten a la configuración de los obstáculos. Por ejemplo, como nosotros hemos puesto en práctica, se pueden trazar rectas verticales en cada vértice, ya que tanto el entorno como los obstáculos tienen límites poligonales. De esta manera, todo el espacio de configuraciones libres queda fragmentado en celdas trapezoidales. A partir de este punto como hemos mencionado anteriormente se genera el grafo de conectividad y se halla la secuencia de celdas que formarán la trayectoria. Para ello simplemente será necesario calcular los puntos medios de los segmentos, los cuales serán los puntos de paso. Uniendo los puntos de paso de las celdas seleccionadas obtendremos la trayectoria. A este tipo de fragmentación en celdas se le denomina “Descomposición e celdas verticales”.

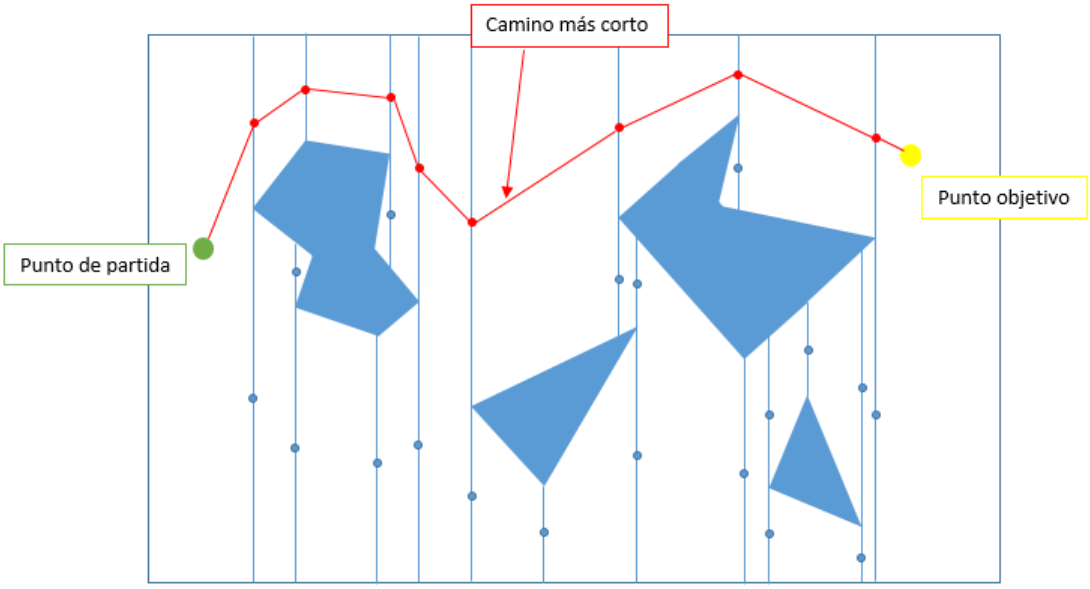


Figura 4-3. Descomposición en celdas verticales.

Este tipo de algoritmo precisa de una ordenación previa de los vértices, lo cual implica un tiempo de cómputo. Tras esto el trazado de las líneas verticales, la designación de las celdas y la identificación de las que contienen la posición origen y destino también repercute en el tiempo de cómputo.

Otro ejemplo de descomposición en celdas exactas lo forma la triangulación de Delaunay. En este tipo de algoritmos se consideran inicialmente el conjunto de puntos formado por los vértices de los obstáculos. Posteriormente se une mediante una recta los pares de puntos mediante los cuales es posible hallar una circunferencia la cual los contenga y en la que no haya ningún punto más. Este tipo de exclusión implica que los puntos del par son los más cercanos a la circunferencia hallada, y que el centro de la misma equidista de los puntos a unir. De estos resultados se deduce que este centro pertenecerá al diagrama de conectividad, el cual se calcula como hemos mencionado anteriormente.

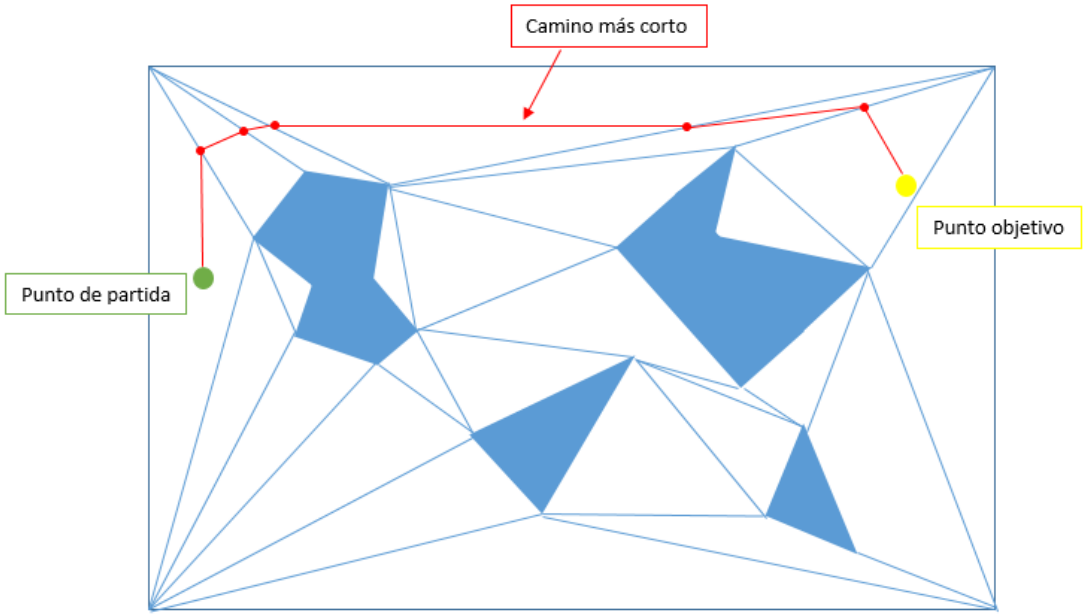


Figura 4-4. Descomposición en celdas triangulares.

Como podemos observar este método resulta más complejo aunque posee la ventaja de que los puntos medios de los segmentos de las celdas se distancian de una forma más eficiente de los obstáculos, y por tanto la solución que obtenemos nos ofrece mayor margen de seguridad en cuanto a evitar colisiones.

- Descomposición aproximada: En este tipo de descomposición se utilizan celdas de diseño predefinido, normalmente rectangulares. Al descomponer el entorno en dichas celdas, algunas de ellas solaparán completamente o en parte con algún obstáculo. Éstas serán descartadas, las demás serán las que formarán el grafo de adyacencia. El resto del proceso es muy similar al anterior. La ventaja de este método con respecto al anterior la encontramos en lo sencillo que es implementarlo. A continuación se describen dos ejemplos para clarificar la explicación: el método de división en cuadrantes, también conocido como "quadtree", y el método de frente de ondas, también llamado "marching method".
  El método conocido como "quadtree" parte de un entorno el cual incluimos en un rectángulo. El siguiente paso sería dividir el entorno en rectángulos iguales mediante el método de trazar mediatrices en la base y la altura original, obteniendo así cuatro rectángulos. Estos rectángulos se clasificarán en rectángulo lleno, rectángulo vacío o rectángulos en los que encontremos ambas situaciones. Únicamente estos últimos volverán a sufrir una fragmentación usando el mismo sistema que anteriormente. Los cuales de nuevo se volverán a etiquetar como llenos, vacíos y mixtos, siendo los mixtos los que volverán a sufrir el proceso de fragmentación.

  En cada una de estas etapas se actualiza una estructura de datos tipo árbol de grado cuatro. Es decir, de cada nodo del árbol descienden otros cuatro.

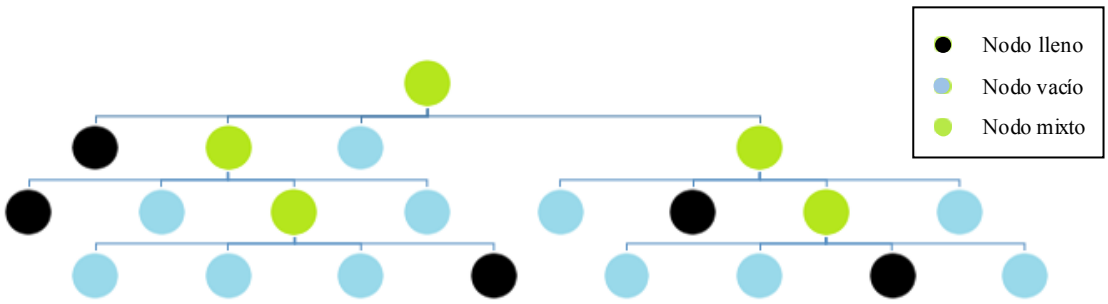


Figura 4-5. Estructura en árbol de las particiones.

Como vemos en la imagen en la que el negro representa que está lleno, el azul que está vacío y el verde que es mixto. Usando este árbol, se comprueba en cada iteración si existe una secuencia de celdas vacías que conecte la celda origen con la celda destino. Si la hay, se genera un camino utilizando los puntos medios de los rectángulos vacíos. Si no la hay, se vuelve a hacer una iteración. Existe un tamaño mínimo de celda, por lo que si tras alcanzarse ese tamaño en las iteraciones no se encuentra camino, se devuelve que no ha sido posible encontrar una solución.

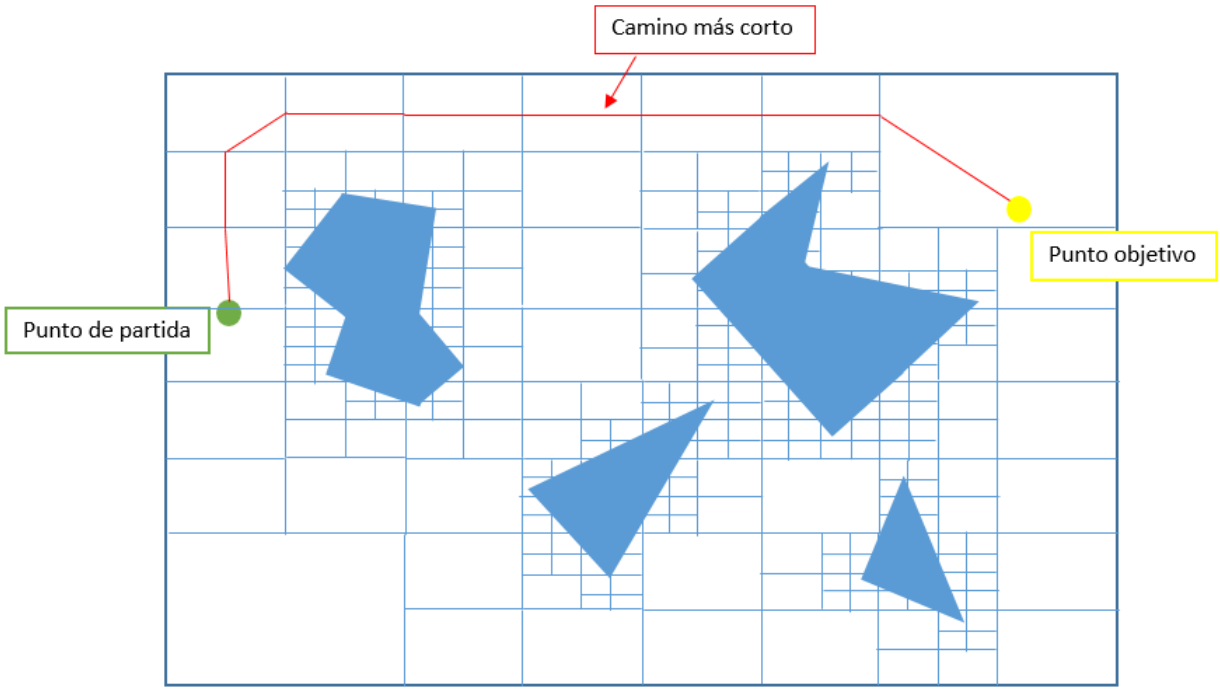


Figura 4-6. Método quadtree.

Existen otros métodos que parten de una descomposición en celdas del mismo tamaño. Uno de estos métodos es el "fast marching method". Su funcionamiento consiste en descubrir la trayectoria de un rayo de luz emitido desde la posición origen hasta la de destino, aplicando la teoría de ondas.

La solución que este método nos proporciona es similar a la de los grafos de visibilidad. Este tipo de soluciones suele estar pegada a los obstáculos, lo que no es deseable. Sin embargo normalmente este método suele usarse combinado con otros para así ofrecer una mayor calidad.

4.1.3 Diagramas de Voronoi

Los Diagramas de Voronoi son en realidad grafos, los cuales dan nombre al método. Estos diagramas poseen la ventaja de que maximizan la distancia entre el robot y los obstáculos. Este método no solo es usado directamente, sino también como base de otros métodos. En este método es necesario al igual que en otros modelar los obstáculos como polígonos. Esto nos permite adoptar la analogía del robot puntual, es decir, la solución que obtenemos será la seguida por un punto en el plano, el cual representa el robot.

Un diagrama de Voronoi generalizado (lo nombramos de esta forma para diferenciarlo del original, el cual reducía los obstáculos a puntos) muestra los puntos que equidistan de elementos vecinos[1], estos pueden ser obstáculos o bordes del entorno.

Teniendo en cuenta que el lugar geométrico que nosotros hemos usado está formado por obstáculos poligonales, el diagrama de Voronoi estará compuesto por rectas y parábolas, esto es debido a que los puntos que equidistan de un punto y una recta forma una parábola y los puntos que equidistan de dos rectas generan una recta. No obstante los diagramas de Voronoi también pueden generarse a partir de entornos formados por objetos curvos, aunque la forma del diagrama no serán estructuras predefinidas, sino que dependerá de la forma de los elementos en cada caso concreto.

El siguiente paso tras haber creado el grafo es el unir el punto origen con el punto destino. Para ello en primer lugar comprobamos si el punto se encuentra en la vecindad de dos segmentos, si esto es así se calcula la recta ortogonal que une el punto al segmento más cercano. Esta recta junto con su intersección con el grafo será lo que proporcione la unión del punto y el grafo. Si por lo contrario el punto se encuentra entre la vecindad de una

[1] Los elementos vecinos son aquellos entre los que no hay ningún otro elemento

recta y un vértice, se calcula la recta que pasa por el vértice y el punto, obteniéndose así la intersección con el grafo la cual proporcionará la unión del punto y dicho grafo. A partir de este momento el problema consiste en unir ambos puntos de intersección (el del punto de salida y el punto de llegada con el grafo) siguiendo una secuencia de vértices en el grafo. La elección de los vértices dependerá del criterio que el programador elija, en nuestro caso al igual que en el resto de algoritmos nuestra preferencia es que haya el menor número de saltos.

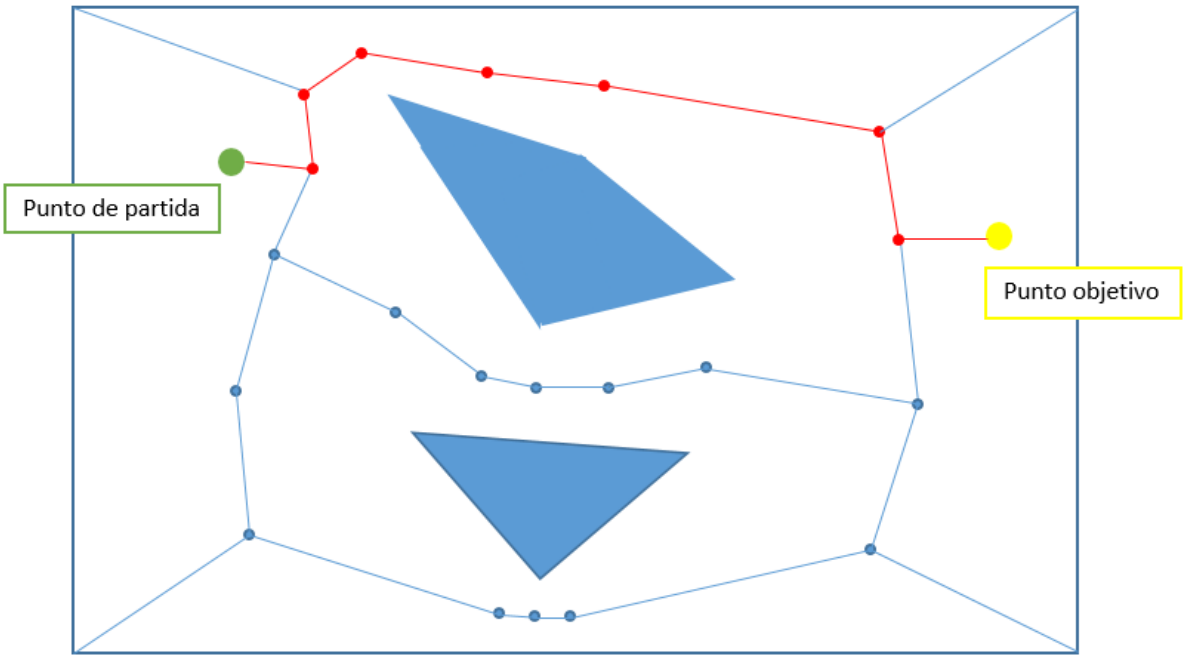


Figura 4-7. Diagramas de Voronoi.

Un posible inconveniente de este método podría ser que antes de implementarlo es necesario comprobar si el robot que usaremos es capaz de maniobrar según las exigencias de la trayectoria generada, ya que a veces este algoritmo presenta giros abruptos en los vértices. En contraposición este método posee una gran ventaja con respecto a otros y es la capacidad de alejarse de los obstáculos. Esto puede ser de gran utilidad en espacios donde haya muchos obstáculos o los pasos entre ellos sean muy estrechos. Sin embargo esto podría convertirse en un inconveniente si nos encontramos en un entorno con escasos obstáculos, ya que la trayectoria generada podría no ser la más óptima, eligiendo una que nos haga dar grandes rodeos.

4.1.4 Algoritmos Bug

La familia de los Bugs son un conjunto de algoritmos que son usados para la construcción de rutas de navegación en entornos reales con presencia de obstáculos. Estos algoritmos poseen una gran eficacia en la navegación basada en sensores, además de poseer una elevada facilidad de implementación. Este tipo de algoritmos no necesita una visión global del entorno, ya que su funcionamiento se basa en un conocimiento local del mismo. El objetivo es simple, encontrar una ruta libre de colisiones, mediante las restricciones dadas por los obstáculos y circunnavegando dichos objetos. Los algoritmos bugs actúan bajo tres hipótesis. El robot es un punto, La localización debe ser perfecta y es necesario que los sensores sean precisos. El conjunto de los algoritmos bug se componen entre otros de bug1 y bug2.

- Bug1: Este tipo de algoritmo al formar parte de la familia de los bugs, traza una línea que una el punto inicial y el de destino y comienza su seguimiento hasta el momento en el que detecta un obstáculo. En este momento el robot circunnavega el obstáculo completamente recordando cual es el punto que está más cerca de la meta. Tras ello el robot bordea el obstáculo hacia el punto que hemos detectado anteriormente como más cercano a la meta por el camino más corto. En este punto volveríamos a calcular la recta que une el punto en el que nos encontramos y la meta, volviendo a circunnavegar si encontramos un obstáculo, calculando el camino más corto, y así sucesivamente hasta que lleguemos al punto deseado.

Por todo lo anteriormente expuesto podemos comprobar como el algoritmo de bug1 es un algoritmo de búsqueda exhaustiva, en el que se examinan todas las opciones posibles antes de comprometerse con na de ellas. Debido a este motivo el algoritmo de bug1 precisa de una capacidad computacional elevada y el tiempo necesario para generar la trayectoria es mayor que en otros algoritmos de su misma familia.

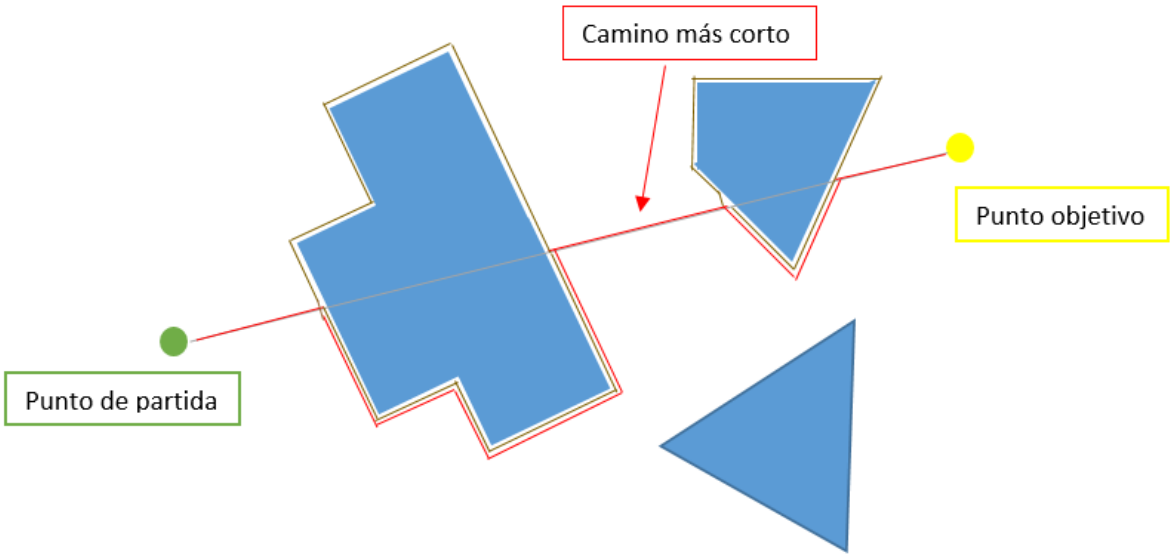


Figura 4-8. Algoritmo Bug 1.

- Bug2: Estos algoritmos inicialmente general una línea que parte del punto inicial y llega al punto final, esto es una característica típica que encontramos en todos los algoritmos pertenecientes a la familia de los bugs. En segundo lugar el robot se va desplazando por dicha recta hasta que se encuentre un obstáculo en su trayectoria, en dicho momento el robot bordea el obstáculo hacia uno de los dos lados aleatoriamente hasta volver a encontrarse con la recta originalmente calculada, de forma que volveremos a repetir el procedimiento descrito anteriormente cada vez que el robot detecte un objeto. Hasta llegar al punto deseado.

  Este algoritmo como describimos anteriormente es un algoritmo voraz, ya que no se plantea que camino será mejor tomar, simplemente toma uno aleatoriamente. Por ese mismo motivo la capacidad computacional de este algoritmo es mucho menor que en el algoritmo de bug1, además de ser considerablemente más pequeño el tiempo usado para general dicho algoritmo.

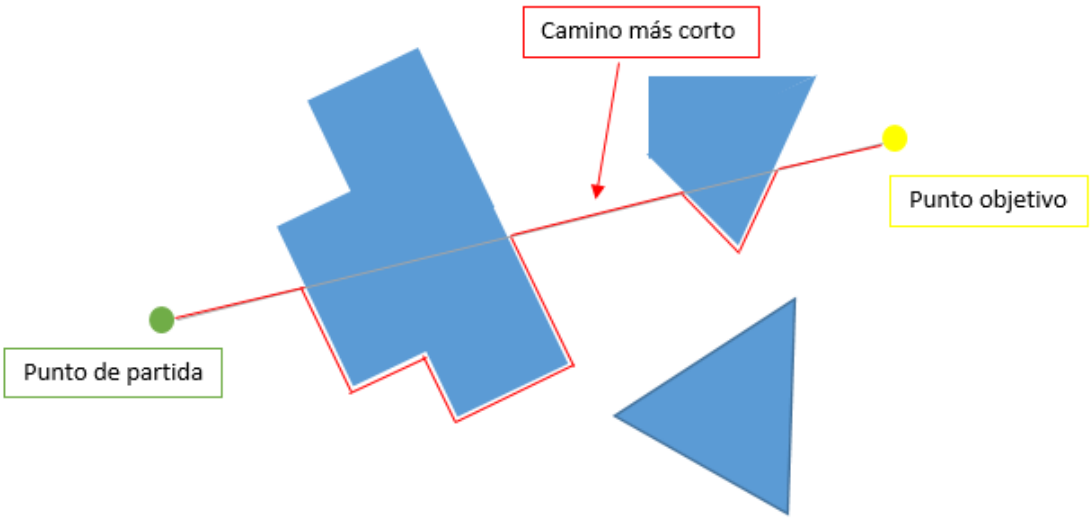


Figura 4-9. Algoritmo Bug 2.

Por los motivos mencionados anteriormente podemos llegar a la conclusión de que el algoritmo Bug2 supera en muchos casos al algoritmo Bug1 y por este motivo ha sido el que hemos implementado. A pesar de esto podemos comprobar como el algoritmo Bug1 tiene un rendimiento global más predecible lo cual para algunos casos podría ser más interesante.

4.1.5 Planificador de frente de ondas

Este método convierte el entorno en el que se encuentra el robot en una matriz, en la que los valores de cada elemento de dicha matriz varían según sea un obstáculo, el destino o la proximidad a este. En primer lugar debemos distinguir las celdas que son obstáculos de las que no, para ello le asignamos el valor cero a cada elemento que esté libre y un uno a cada elemento que represente un obstáculo. Tras esto definimos la celda que represente el punto al que queremos llegar con un valor dos.

Punto de partida

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2

Punto objetivo

Figura 4-10. Distinción de celdas obstáculo.

Una vez diferenciado entre obstáculo y camino libre podemos pasar a ir dándole valores a las celdas según estas estén más o menos próximas al punto de llegada deseado. Para ello comenzando por el punto de destino vamos actualizando todas las celdas conexas (las ocho celdas que estén conectadas con la celda actual) con valor 0, al valor actual más uno. Este proceso lo vamos repitiendo constantemente hasta que alcancemos el origen del que partiremos.

Si alguna celda se ha quedado a cero tras haber realizado todas las iteraciones significa que dicha celda es una zona no alcanzable por el robot, es decir, que se encontrará en el interior de obstáculos o en situaciones similares.

Finalmente para obtener el camino deseado, partiendo desde el origen, nos vamos moviendo a celdas con un valor numérico inferior hasta llegar al punto destino. Según la métrica establecida, normalmente el valor de cada celda, es aproximadamente la distancia del punto en el que os encontramos al punto de destino.

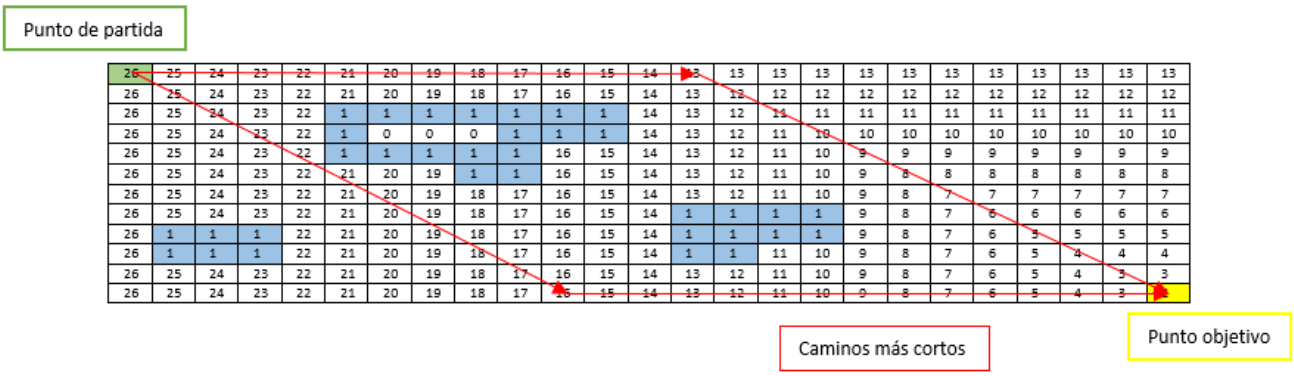


Figura 4-11. Planificador de frente de ondas.

Una desventaja que encontramos en este tipo de algoritmos es que al haber alrededor del punto en el que nos encontramos varios puntos posibles a los que movernos, no siempre los puntos elegidos para llegar a nuestro objetivo son los más óptimos para el movimiento de nuestro robot, ya que en múltiples situaciones nos podremos encontrar con caminos en zigzag.

Por otro lado el este algoritmo es un algoritmo bastante sencillo de implementar.

4.2 Algoritmo no implementado

4.2.1 Campos de potencial

Este método trata al robot como una partícula, la cual se encuentra sometida a diferentes fuerzas. Por un lado tenemos una fuerza de atracción hacia el punto final, al cual queremos llegar, y por otro tenemos las fuerzas de repulsión que ejercen los obstáculos sobre el propio robot para que este no colisione con los mismos. Para definir estas fuerzas consideramos un campo potencial que esté asociado al entorno en el que se encuentra el robot, en el cual su valor dependa de la proximidad con los obstáculos y la distancia al punto de destino.

El planificador de trayectorias genera una trayectoria en función de un vector gradiente derivado del campo potencial. Se calcula el vector en el origen y se añade a la trayectoria un segmento, el cual debe ser de un tamaño fijado previamente, cuya dirección sea la designada por el gradiente. En el extremo del segmento mencionado volvemos a calcular el gradiente, añadiendo nuevamente otro segmento en su extremo en la dirección que el vector indique. Este proceso lo repetimos constantemente generando así la trayectoria de forma que el robot siga siempre la dirección que minimice el valor del campo potencial. Nuestro objetivo final será el de llegar al mínimo absoluto que será el punto al que queremos llegar. Sin embargo, el entorno en el que el robot se encuentra puede contener mínimos locales.

Este tipo de métodos poseen diversos problemas, por los cuales hemos decidido descartar su realización a lo largo de este trabajo.

El problema principal, por el cual no hemos usado este algoritmo en el proyecto, lo encontramos cuando tenemos estos mínimos locales, en estos casos la fuerza resultante es nula, pero el punto al que hemos llegado no es el que se deseaba alcanzar. Una de las formas por las que podríamos resolver este inconveniente sería recurrir a la generación de trayectorias aleatoria, mediante la cual tras planificar ciertos movimientos aleatorios se consiga salir se ese mínimo local. Tras salir de dicho mínimo continuaremos con el procedimiento que hemos explicado inicialmente.

Además del inconveniente anterior, este método posee algunos otros problemas. El principal es calcular el tiempo que transcurrirá en la generación del camino deseado, el cual debe ser lo suficientemente largo como para ser capaz de abandonar cualquier mínimo local que se encontrase, pero tampoco demasiado ya que sino consumiría demasiado tiempo de computo. Normalmente la solución que se le da a dicho problema es calcular el tiempo de cómputo mediante una función aleatoria. Otro de los inconvenientes se debe a los caminos aleatorios que hemos usado para conseguir salir de los mínimos locales, ya que pese a ser muy sencillos de calcular generan una densidad de probabilidad no uniforme.

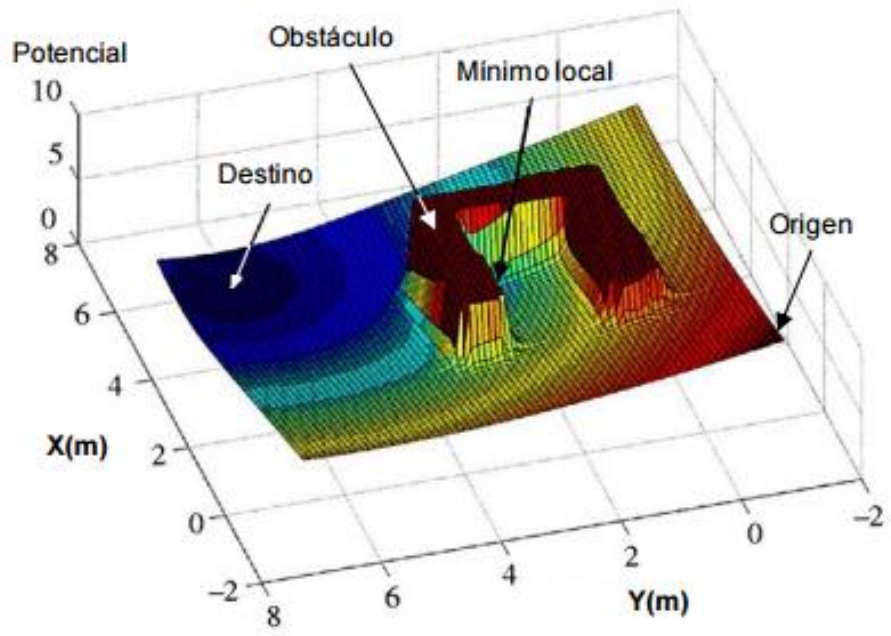


Figura 4-12. Campos de potencial.

5 SIMULACIONES Y COMPARACIONES

Tras la anterior explicación de los algoritmos que hemos implementado, en este apartado pasaremos a analizar el funcionamiento de cada uno de ellos en los dos mapas diferentes que teníamos creados, para así poder ver las diferencias y similitudes de cada uno de ellos. Además mostraremos como cada tipo de robot reacciona a cada una de las trayectorias generadas por los diferentes métodos.

Antes de pasar a exponer los resultados obtenidos, se mostrará a modo de aclaración el proceso mediante el cual los programas son llamados y por consiguiente los datos que se pasan de un programa al posterior. En primer lugar tenemos un programa principal, en el cual se determinan los valores d, Kv, Kh, Ki, se elige el mapa que queramos implementar, se llama a tres programas secundarios de Matlab de forma secuencial y por último muestra los resultados obtenidos. El primer programa llamado es el encargado de generar el mapa elegido previamente, dicho programa se rige según lo explicado en el apartado 2, y este pasa como datos el propio mapa generado, los vértices de los obstáculos y el punto inicial y final de la trayectoria. El segundo es el programa que genera la trayectoria deseada según el algoritmo que queramos implementar, dichos algoritmos son los explicados en el punto 4.1, este programa pasa como datos los puntos pertenecientes a la trayectoria obtenida. El último de ellos es el programa que se ocupa del seguimiento de la trayectoria ya conocida por un robot elegido previamente. Para poder llevar a cabo este objetivo es necesario que dicha función de Matlab vaya llamando constantemente al programa de Simulink mostrado en el apartado 3.3, el cual requiere dos variables externas, la x_des y la y_des, que iremos actualizando repetidamente antes de llamarlo. Dichas variables se modificarán en la actualización siempre que el punto en el que nos encontremos coincida con el valor de las mismas, en cuyo caso pasarán a tener el mismo valor que el próximo punto de la trayectoria deseada. Este bucle se finalizará cuando el punto en el que nos encontremos coincida con el punto final de la trayectoria generada.

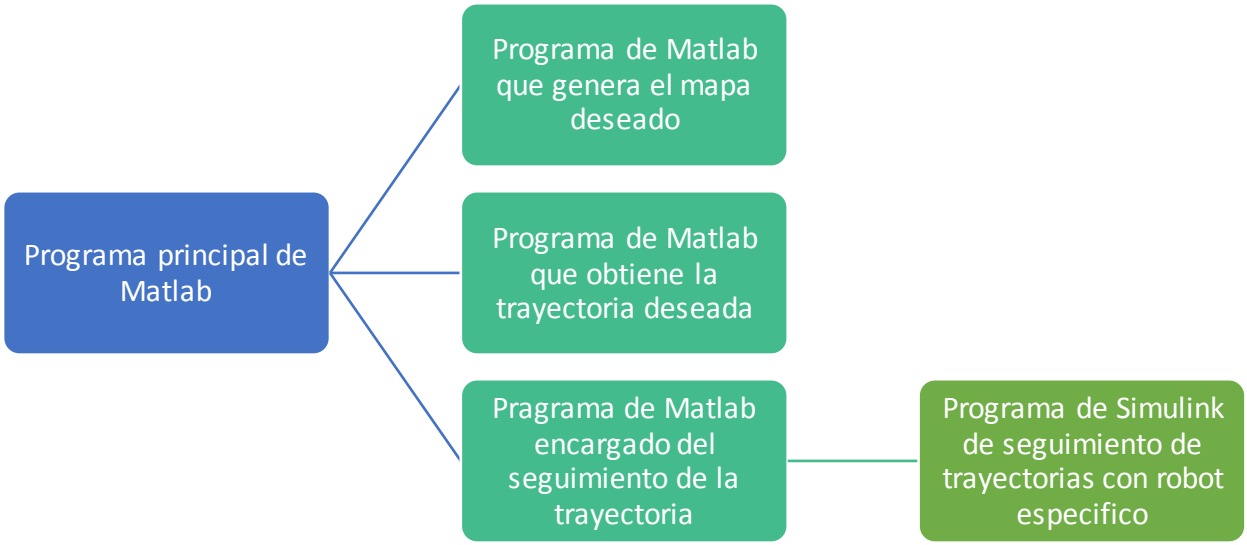


Figura 5-1. Diagrama del proceso de llamada a los programas.

5.1 Trayectorias generadas

En primer lugar mostraremos las trayectorias generadas por los diferentes algoritmos en ambos mapas. En los cuales los asteriscos verdes representan el punto inicial y el final y la recta azul es la trayectoria generada.

5.1.1 Grafos de visibilidad

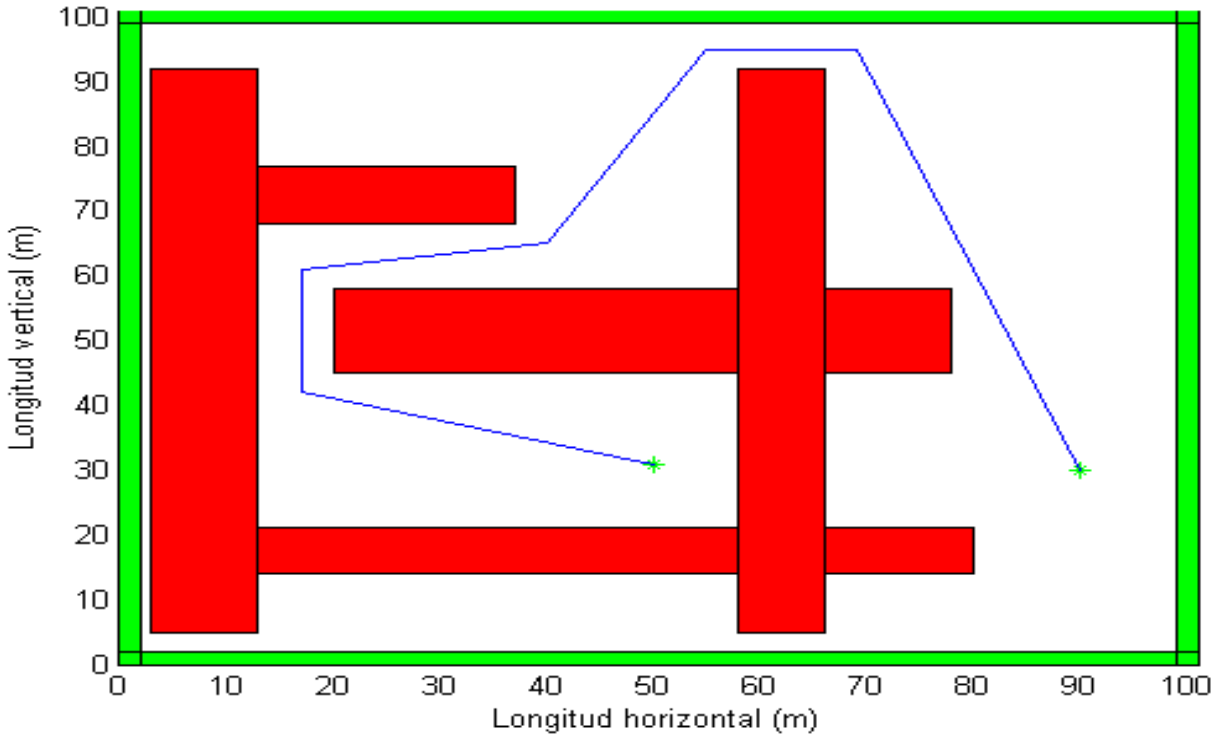


Figura 5-2. Trayectoria generada por grafos de visibilidad en mapa 1.

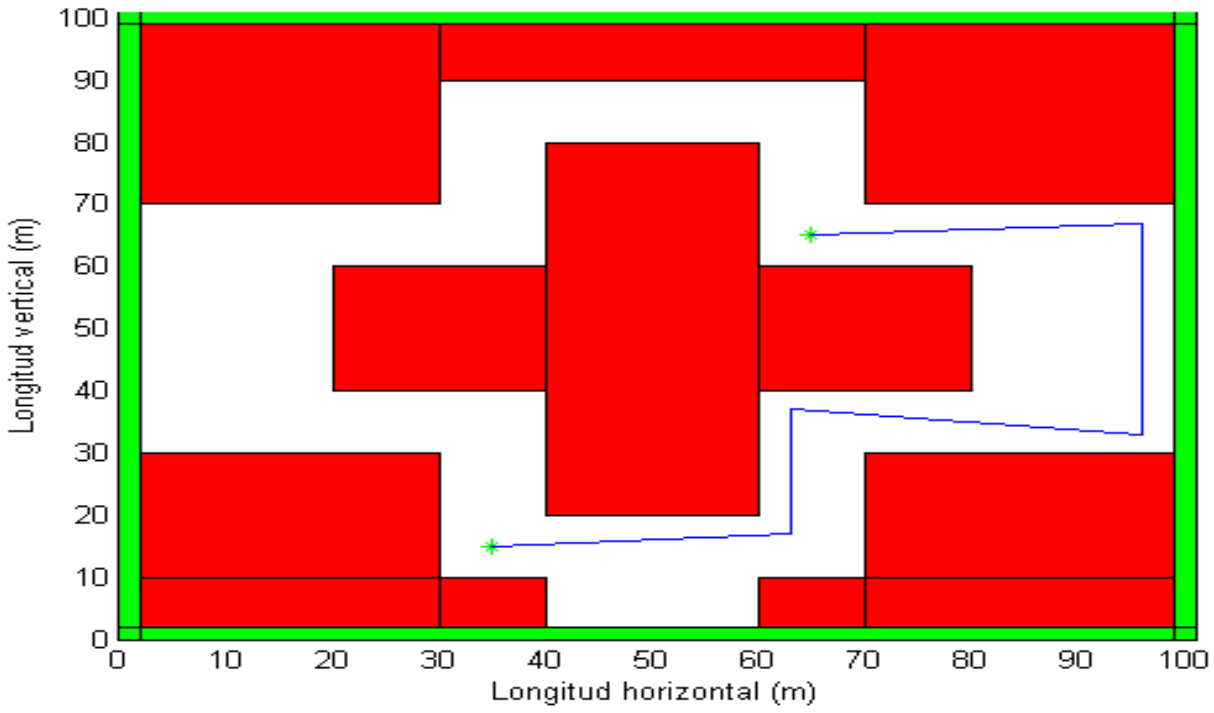


Figura 5-3. Trayectoria generada por grafos de visibilidad en mapa 2.

En ambas imágenes podemos ver como el camino seguido es un camino apropiado, en el que no se produce ningún tipo de colisión con los obstáculos presentes en el entorno, esto es debido a que tras la obtención de nuestra trayectoria hemos separado al robot tres metros de los puntos de la trayectoria, de forma que como el robot es de cuarenta centímetros de diámetro es prácticamente imposible que se produzca algún choque. Por último podemos comprobar como la trayectoria obtenida es la de menor número de saltos, esto es debido a que es nuestro criterio de elección.

5.1.2 Descomposición en celdas verticales

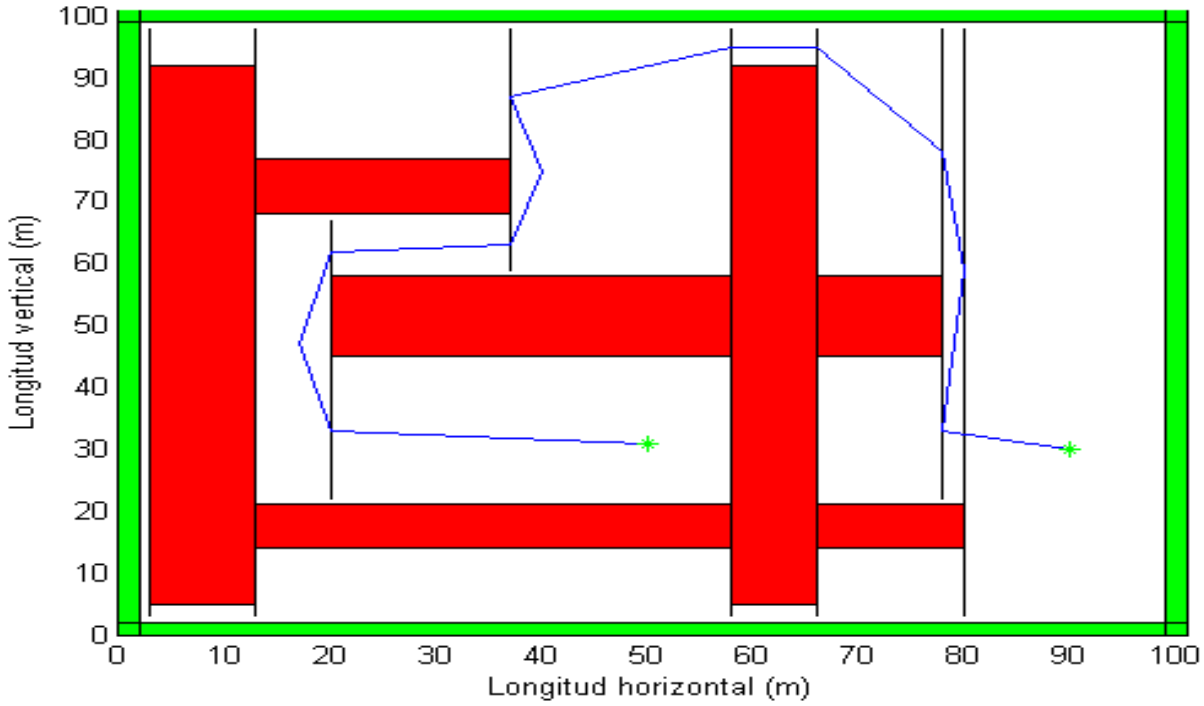


Figura 5-4. Trayectoria generada por desc. en celdas verticales en mapa 1.

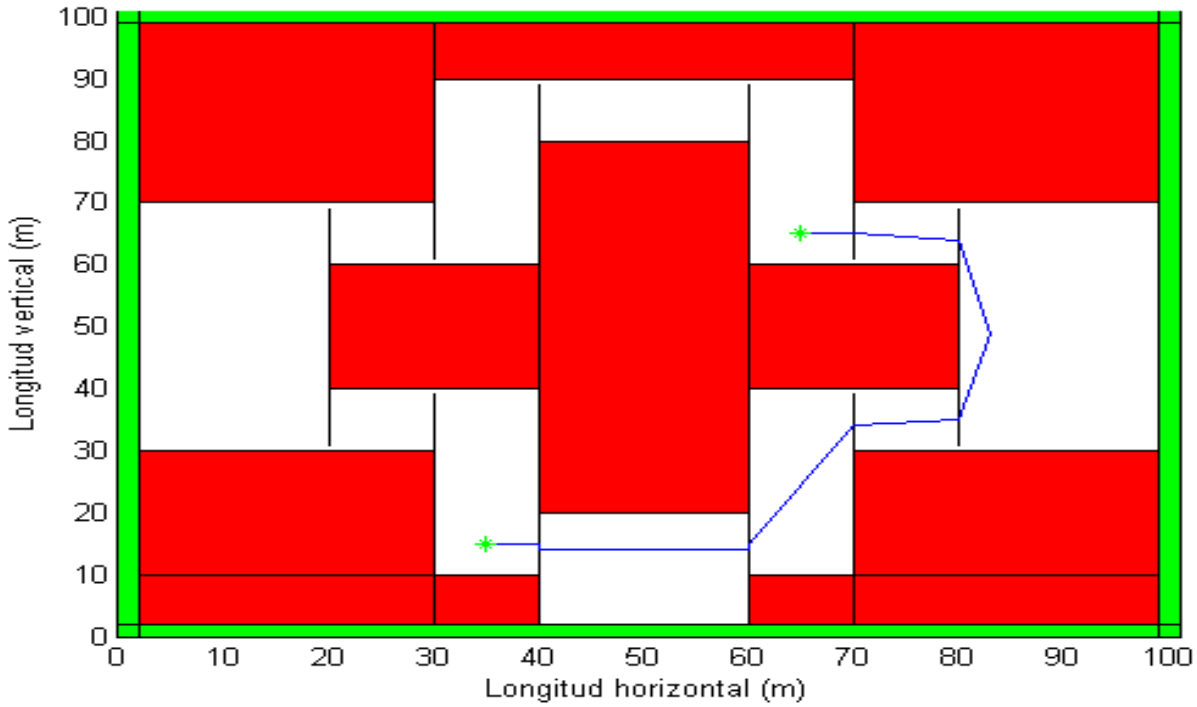


Figura 5-5. Trayectoria generada por desc. en celdas verticales en mapa 2.

En las imágenes anteriores vemos como la trayectoria obtenida es una trayectoria valida en la que no se produce ninguna colisión. Esto gracias a que en los algoritmos de descomposición en celdas verticales los puntos del camino son los pertenecientes a los puntos medios de las rectas que conforman las celdas y este tipo de obtención de puntos nos permite tener la suficiente distancia entre el robot y las paredes que le rodean. El inconveniente que encontramos es cuando dos puntos contiguos de la trayectoria coinciden en el eje x, en este caso ha sido necesario calcular un punto intermedio que esté separado de la pared tres centímetros para que no se produzca ningún choque en los desplazamientos verticales.

5.1.3 Diagramas de Voronoi

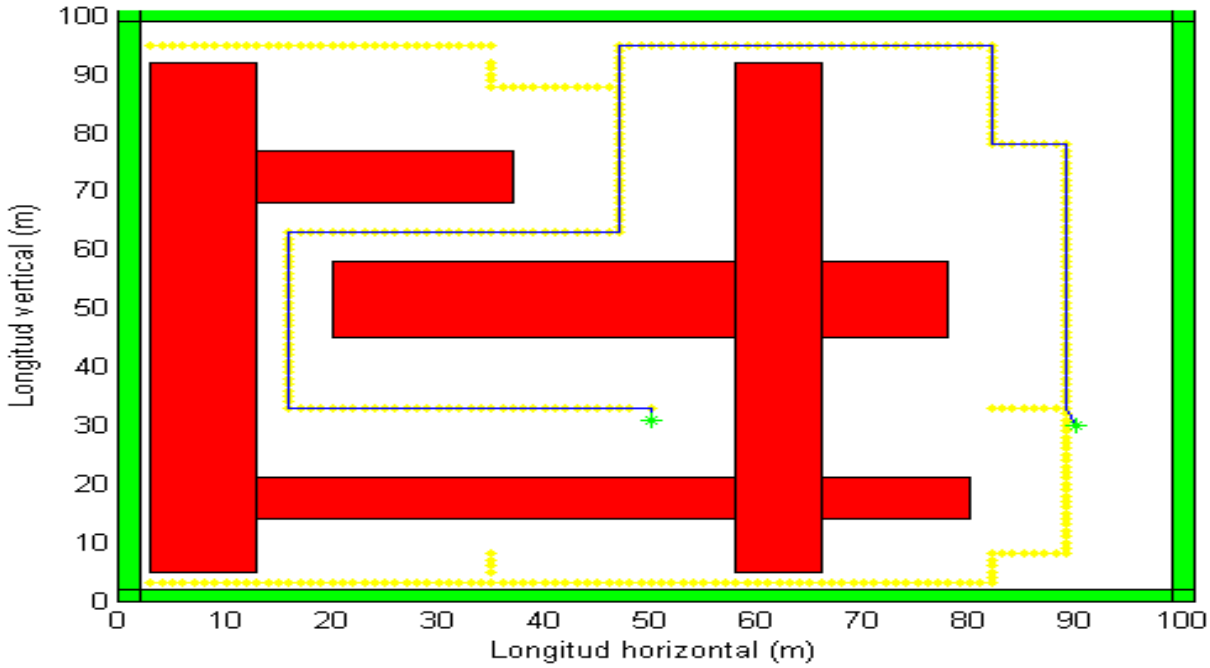


Figura 5-6. Trayectoria generada por diagramas de Voronoi en mapa 1.

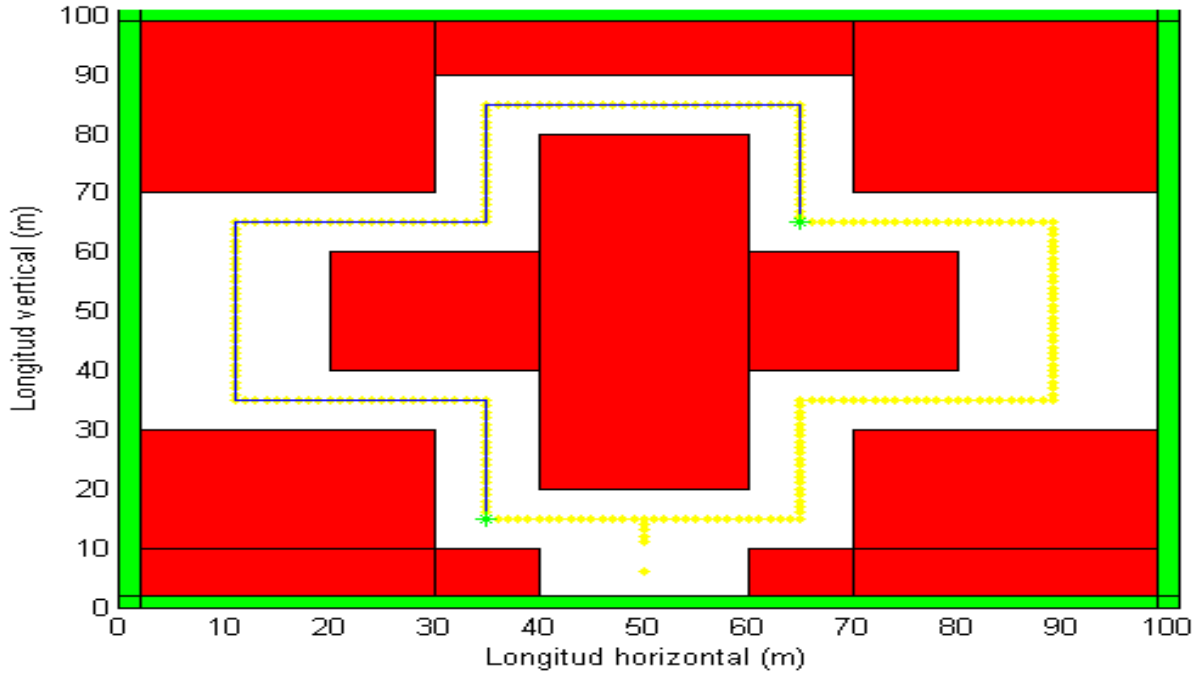


Figura 5-7. Trayectoria generada por diagramas de Voronoi en mapa 2.

Las imágenes anteriores nos muestran como las trayectorias a seguir generadas por el diagrama de Voronoi son unos caminos totalmente válidos en los que no se produce ningún tipo de colisión. De hecho la trayectoria a seguir, como ya habíamos predicho anteriormente al explicar el método, se encuentra en los puntos medios de los obstáculos, situando así al robot en los puntos más alejados de posibles choques. El único inconveniente que podríamos encontrar sería la posibilidad de encontrar caminos más cortos igualmente válidos, lo cual no es suficiente como para descartar el método.

5.1.4 Algoritmo Bug 2

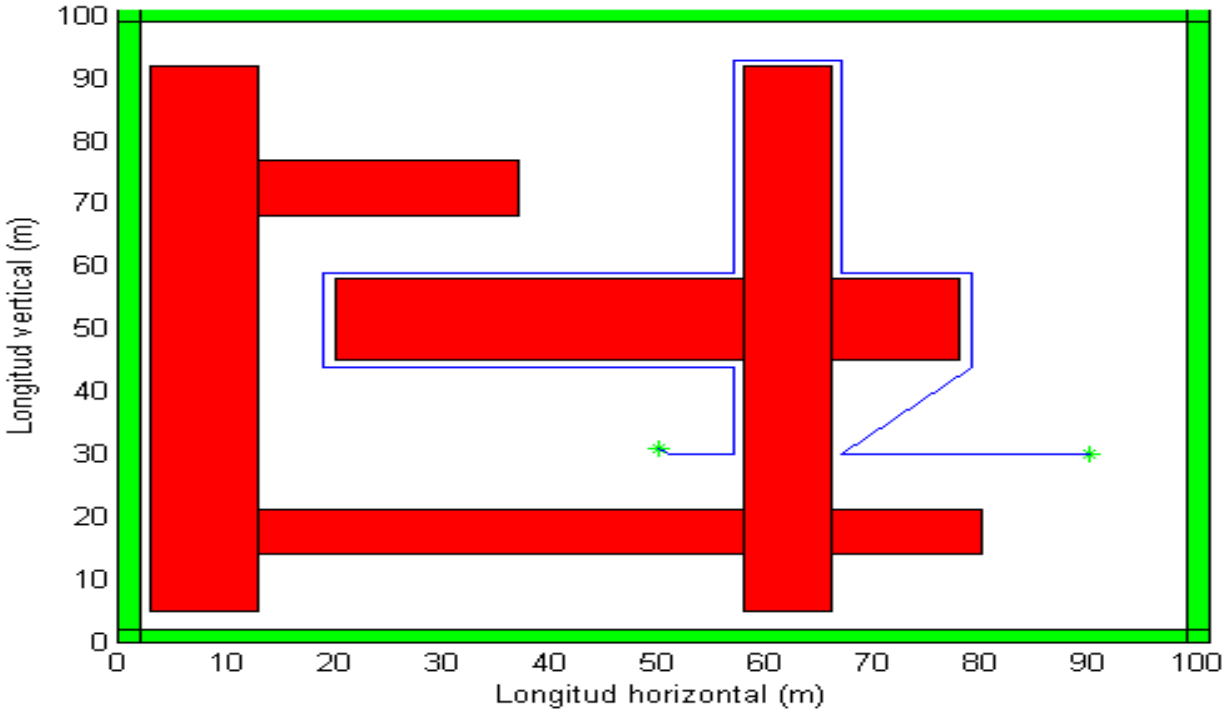


Figura 5-8. Trayectoria generada por Bug 2 en mapa 1.

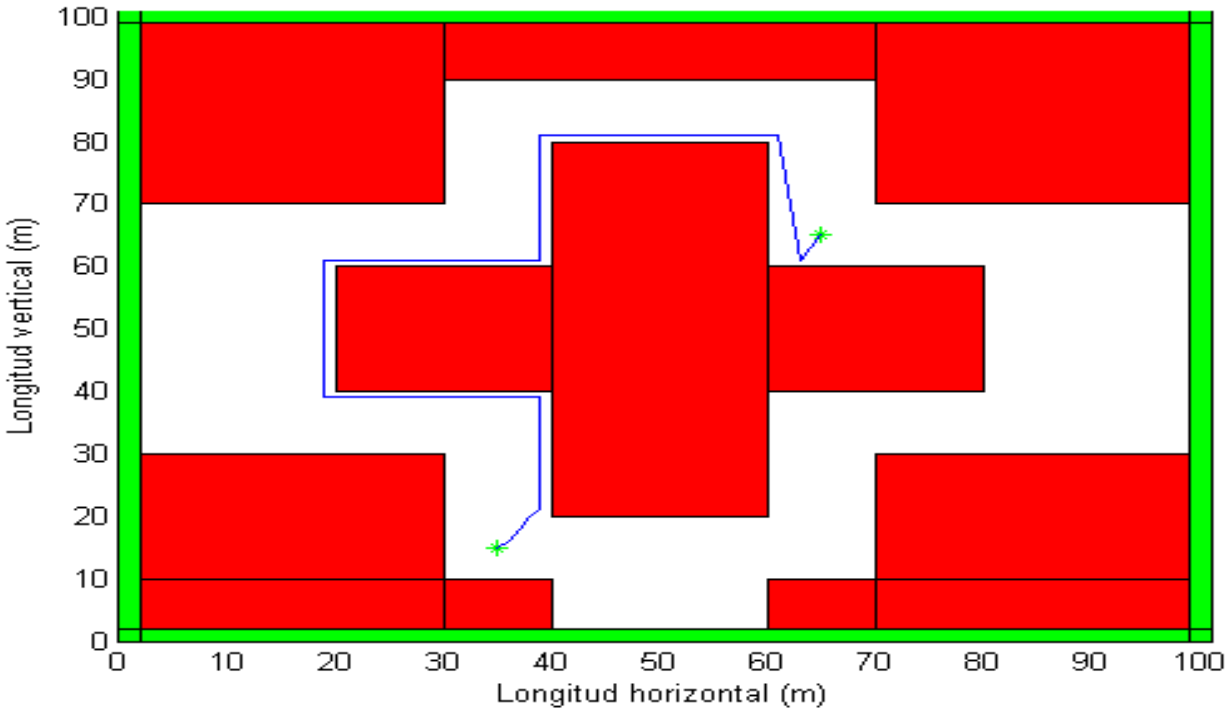


Figura 5-9. Trayectoria generada por Bug 2 en mapa 2.

Vemos en las imágenes anteriores como el camino generado es válido ya que no se produce ningún choque con los obstáculos. Como se puede observar la trayectoria comienza siguiendo la línea que une el punto inicial con el punto final hasta el momento en el que se encuentra un obstáculo. En dicho momento la trayectoria comienza a cambiar y a seguir a las paredes del obstáculo, de forma que el punto próximo del camino será uno de los puntos más próximos a donde nos encontramos que esté pegado al mismo obstáculo. Comprobamos así como no es necesario recorrer todo el objeto como en el algoritmo de Bug 1, por lo que ahorramos carga computacional. A pesar de que en estas trayectorias encontramos puntos cercanos a los objetos, la distancia que hay entre ambos es mínimo de 1 metro, lo cual es suficiente como para que el robot sea capaz de circular sin problema.

5.1.5 Planificador de frente de ondas

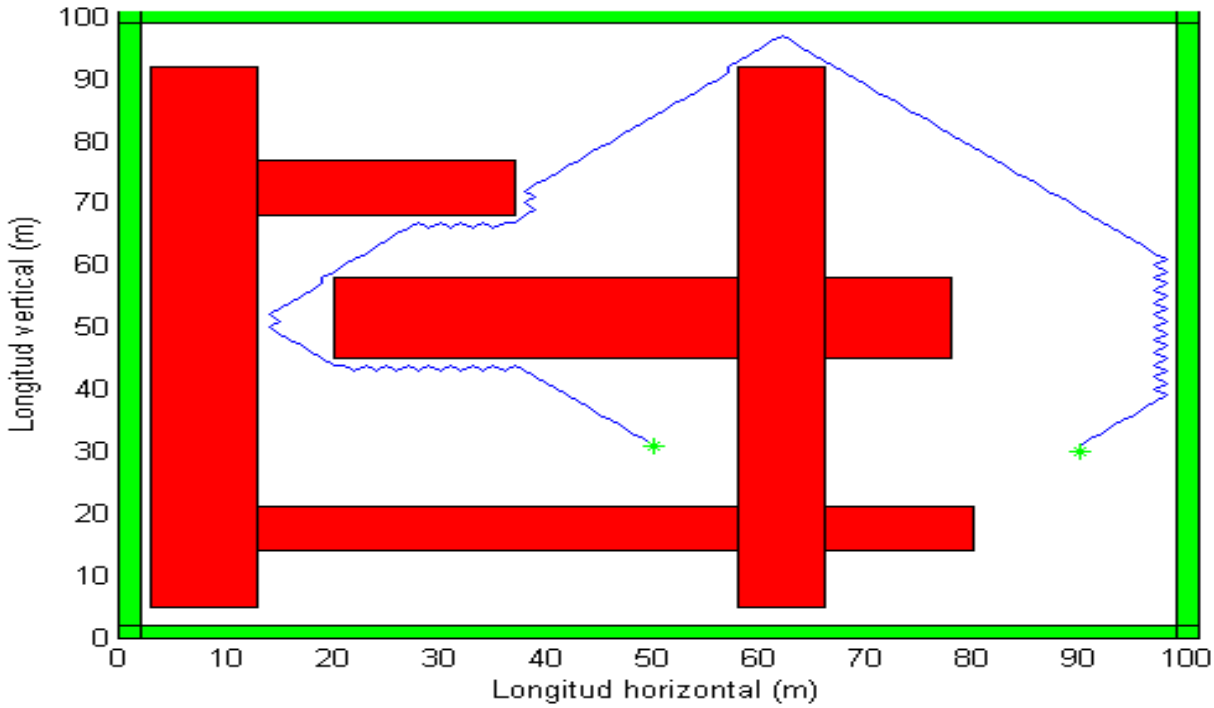


Figura 5-10. Trayectoria generada por planif. de frente de ondas en mapa 1.

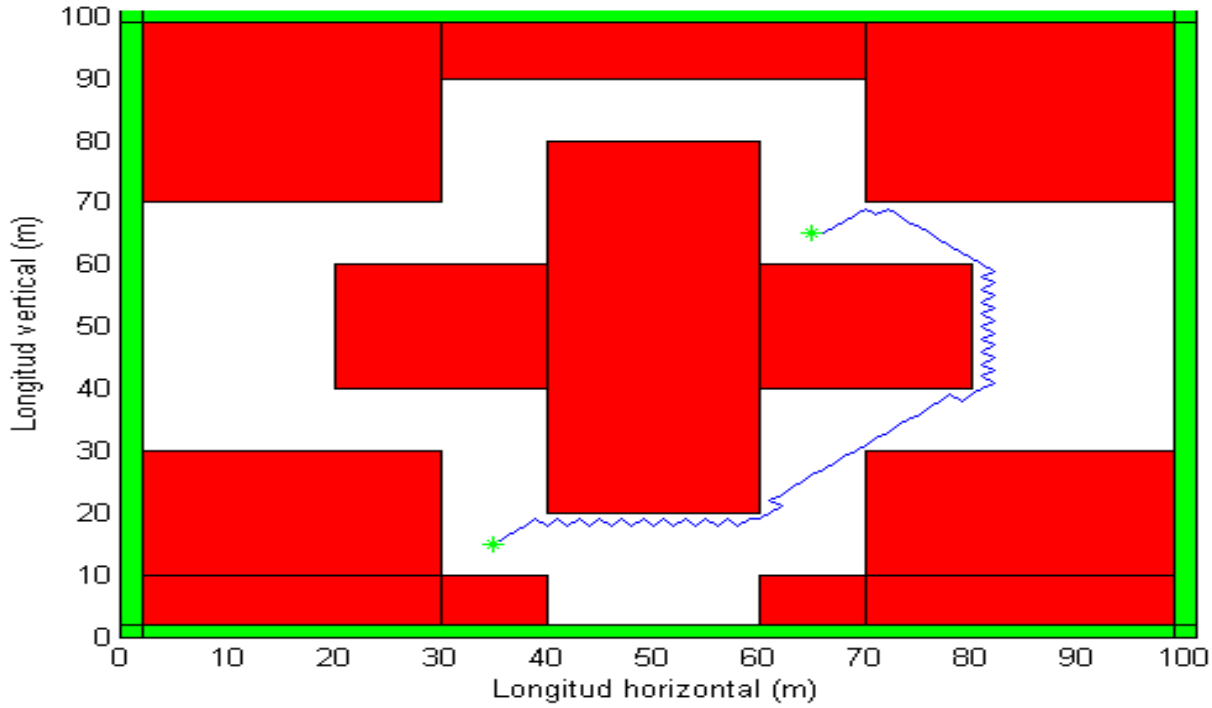


Figura 5-11. Trayectoria generada por planif. de frente de ondas en mapa 2.

En las imágenes anteriores vemos como las trayectorias son aceptables, no produciéndose ninguna colisión. Dichas trayectorias se han formado como esperábamos, encontrando en ellas varios zigzag indeseables para el posterior seguimiento de los robots, esto se produce como ya hemos explicado anteriormente debido a que los valores que rodean a un punto son puntos accesibles, y por ello es posible que se tome cualquiera de ellos. A pesar de que en este tipo de trayectoria encontramos puntos próximos a los objetos, la distancia que hay entre ambos es suficiente como para que el robot sea capaz de circular sin problema.

5.2 Seguimiento de las trayectorias

En segundo lugar mostraremos como los diferentes robots que vamos a usar siguen las trayectorias generadas mostradas en el apartado anterior para ambos mapas. En los cuales los asteriscos verdes representan el punto inicial y el final, la recta azul es la trayectoria generada por el algoritmo, la recta morada es la trayectoria seguida por el robot y el circulo que encontramos en el punto final es el que hemos usado para simular nuestro robot, el cual está sobredimensionado, ya que, este mide 2m de diámetro mientras que nuestro robot mide 0.4m de diámetro. Esto se ha realizado de dicha forma para que fuese más fácil de ver a la hora de realizar las simulaciones. Para cada uno de los robots, como mencionamos en el apartado 3.2, es necesario introducir de manera externa las constantes d, Kv, Kh y Ki, las cuales variarán para cada uno de ellos y en cada método empleado.

5.2.1 Grafos de visibilidad

- Robot biciclo

  Para que el robot sea capaz de seguir la trayectoria de manera fiable, en necesario que sus constantes sean las siguientes:

Tabla 5–1. Constantes del robot biciclo para GV

d	Kv	Kh	Ki
0	0.75	100	1

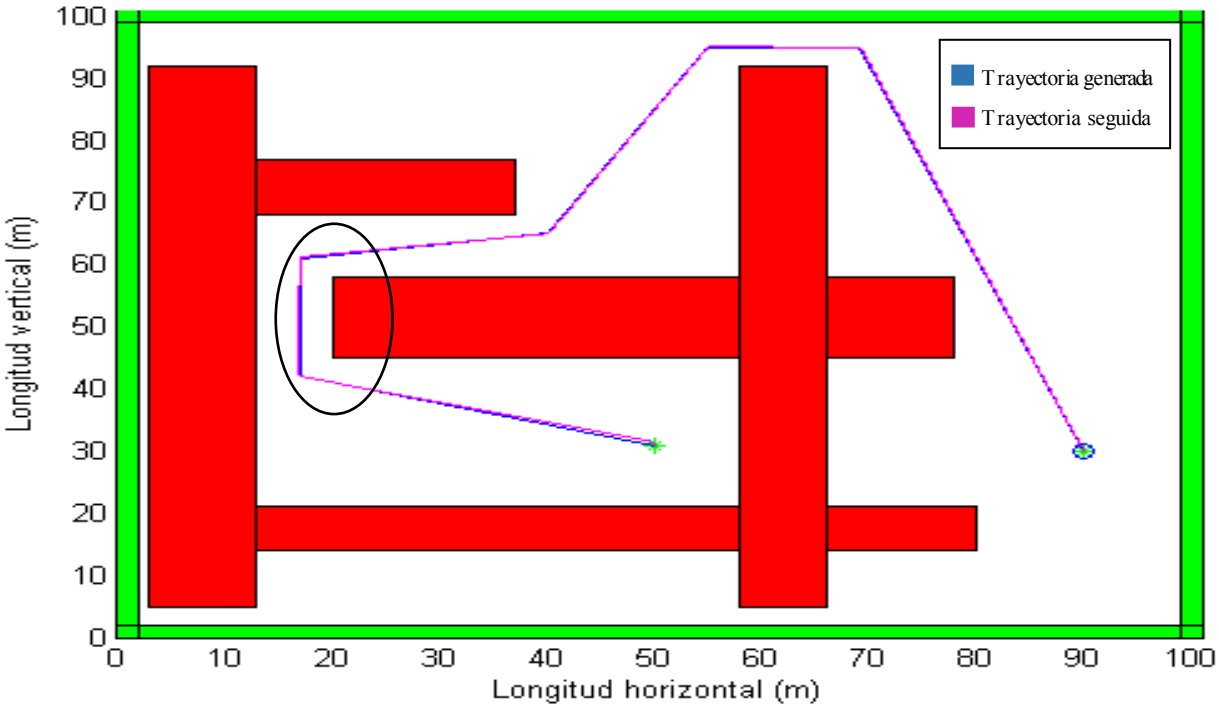


Figura 5-12. Seguimiento de trayectoria con robot biciclo en mapa 1 (GV).

En la imagen anterior vemos como el robot sigue de manera bastante precisa y rápida la trayectoria que hemos generado mediante los grafos de visibilidad, por tanto aceptamos como válidas las constantes elegidas. Ya que el mapa es de 100X100 metros, para poder apreciar mejor hasta qué punto el robot está siguiendo la trayectoria de manera fiable se ha realizado un zoom en la zona que se ha visto con mayores variaciones entre la trayectoria generada y la seguida, dicha zona es la que vemos en la imagen anterior seleccionada.

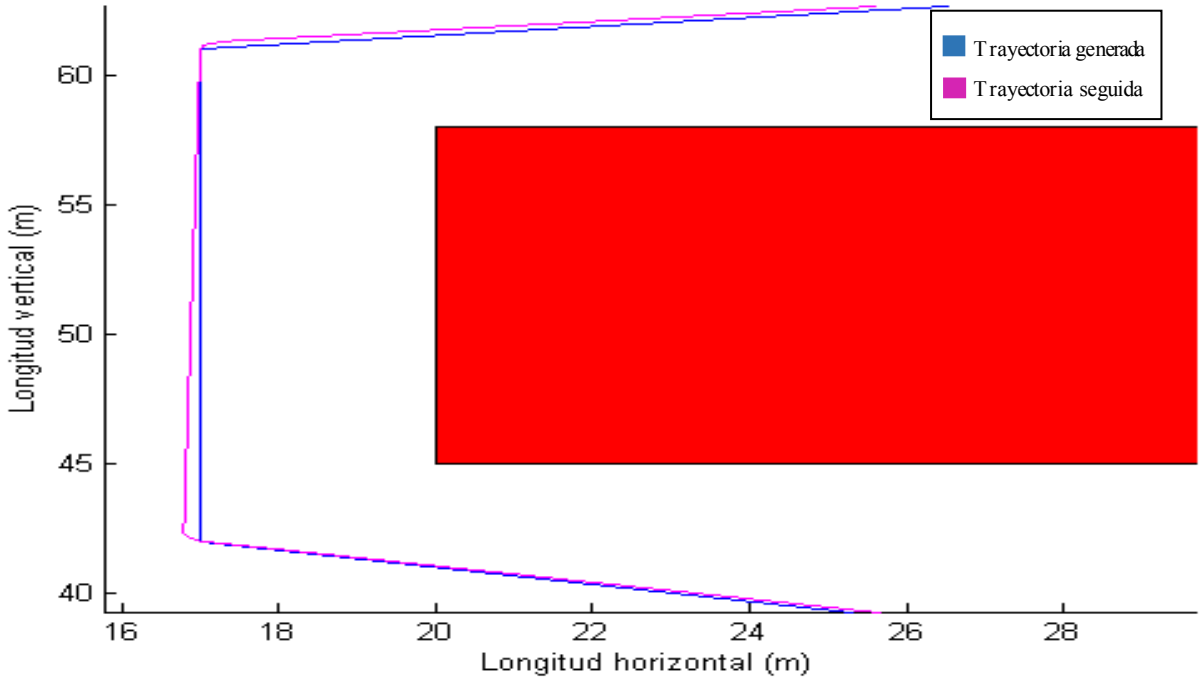


Figura 5-13. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 1 (GV).

Como podemos ver, tras ampliar la imagen vemos que a pesar de que en la curva mostrada exista ciertas

variaciones entre la trayectoria generada y la trayectoria seguida, dichas variaciones son mínimas, por lo que no nos afectaría de forma que pudiese invalidar el movimiento del robot pudiéndose provocar algún choque.

A continuación veremos los resultados obtenidos para el mapa 2 con el mismo robot y las mismas constantes.

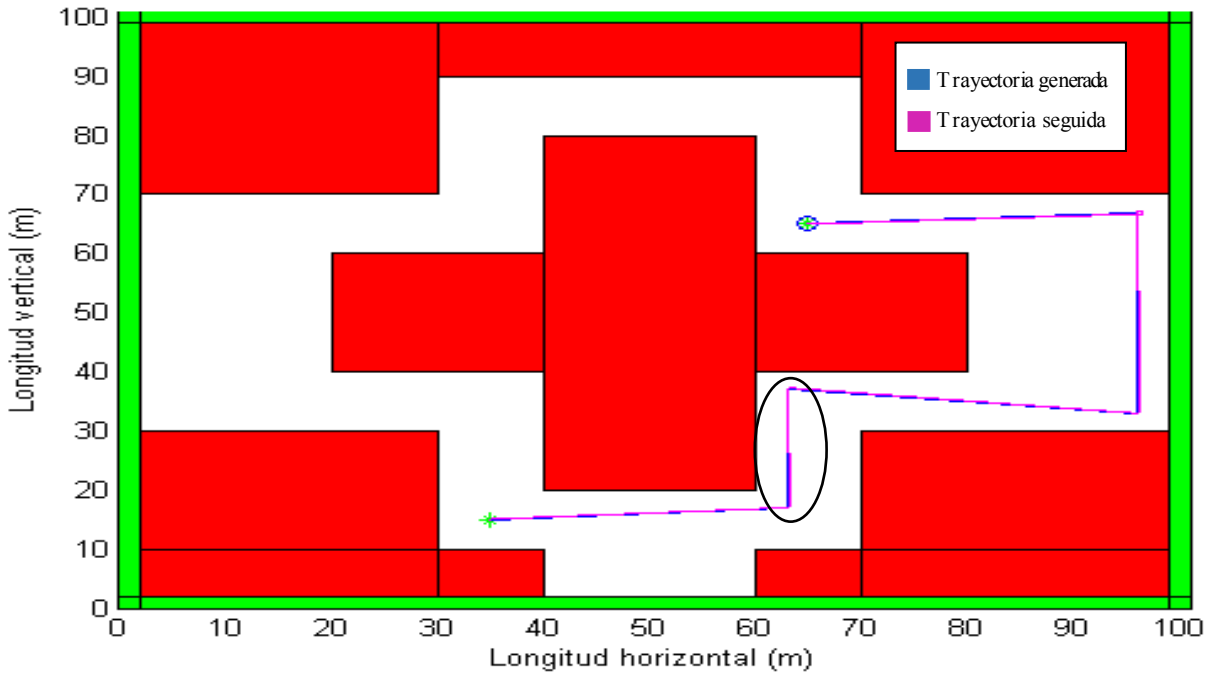


Figura 5-14. Seguimiento de trayectoria con robot biciclo en mapa 2 (GV).

En esta imagen podemos ver como al igual que en el mapa 1 el robot sigue la trayectoria que hemos generado de forma ajustada y rápida, por tanto podemos confirmar como las constantes que habíamos elegido son las apropiadas para dicho robot en el algoritmo de grafos de visibilidad. Al igual que en el mapa anterior también hemos hecho zoom sobre la zona que vemos que la trayectoria formada por el robot y la trayectoria generada es más inexacta. Dicha zona la vemos marcada en la imagen anterior mediante un círculo negro.

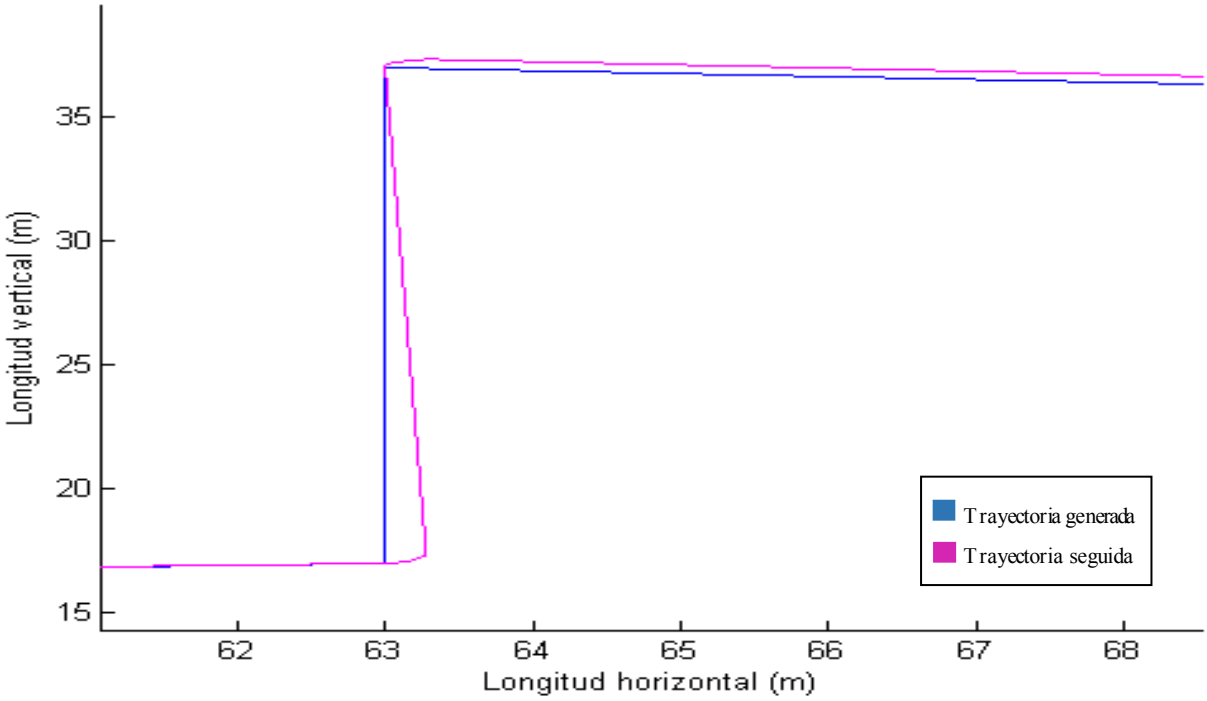


Figura 5-15. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 2 (GV).

Tras la ampliación de la imagen vemos que cuando se producen ciertos giros la trayectoria seguida por el robot tiene pequeñas diferencias con la trayectoria generada, estas diferencias son mínimas aproximadamente de 0.2 metros, por lo que no es un asunto por el que debamos invalidar las constantes elegidas para el robot.

- Robot diferencial

  A pesar de que el algoritmo es el mismo, es necesario variar las constantes ya que es un robot diferente. Las constantes elegidas para que el robot siga la trayectoria de forma adecuada son las siguientes:

Tabla 5–2. Constantes del robot diferencial para GV

d	Kv	Kh	Ki
0	0.005	50	200

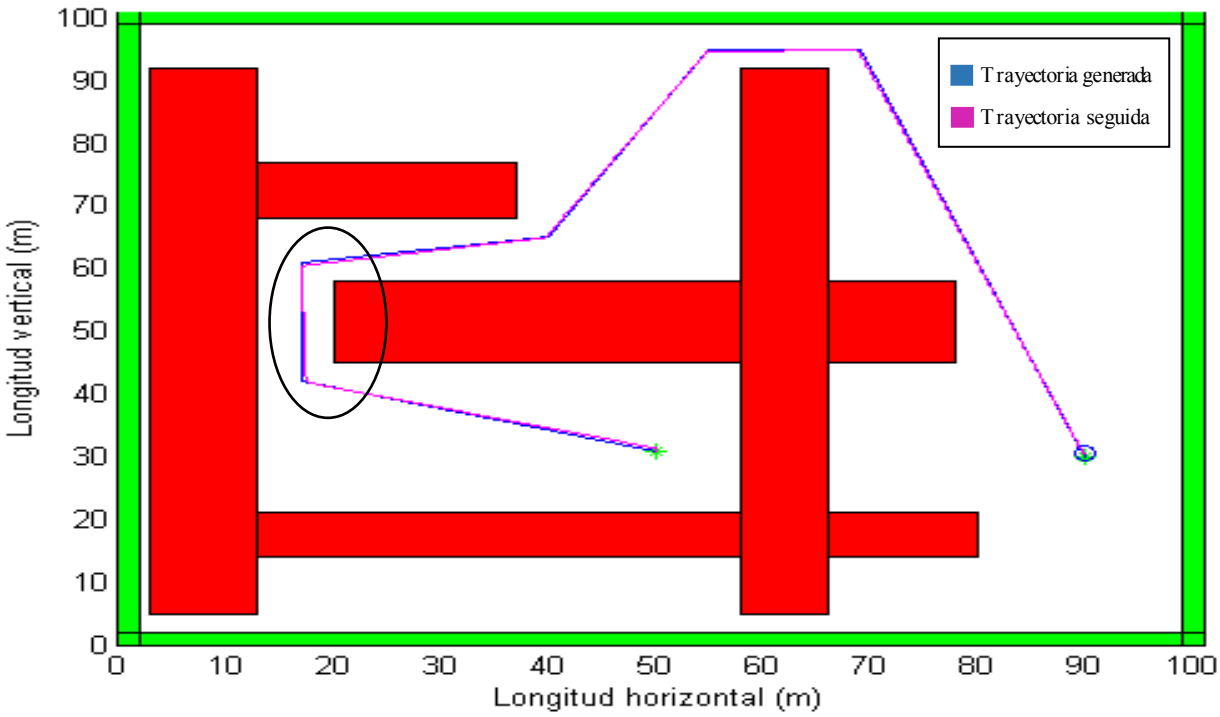


Figura 5-16. Seguimiento de trayectoria con robot diferencial en mapa 1 (GV).

Como vemos en la imagen anterior el robot sigue la trayectoria que hemos generado de manera casi exacta, pero en esta ocasión no se realiza de forma tan rápida. Estos hechos hacen que aceptemos las constantes seleccionadas. Por el mismo motivo que en el robot biciclo, realizaremos un zoom sobre la zona seleccionada, ya que es la que hemos considerado más conflictiva, para así poder ver si cuando observamos el resultado más de cerca, este sigue siendo válido.

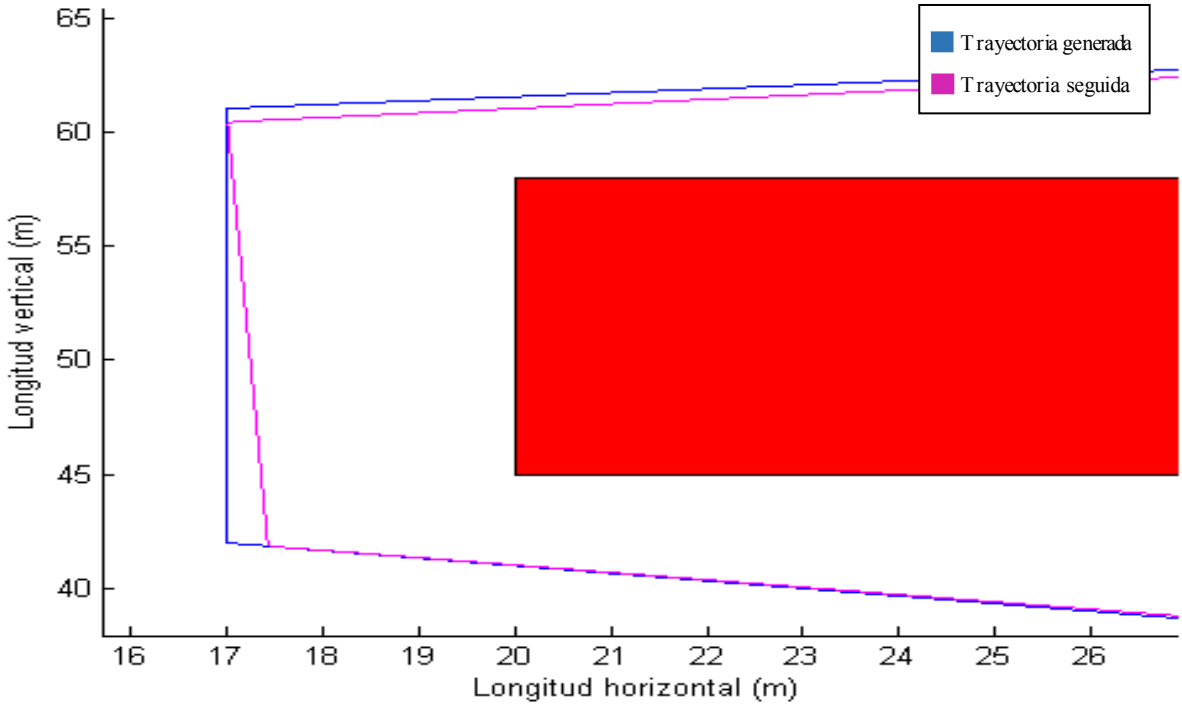


Figura 5-17. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 1 (GV).

Cuando ampliamos la imagen podemos ver como al producirse los giros, el robot tiene alguna dificultad

para seguir fielmente a la trayectoria generada. A pesar de ello y siendo incluso más apreciables en este robot que en el biciclo, lo que difieren ambas trayectorias son aproximadamente 0.5 metros, lo cual no es suficiente para no aceptar las constantes que hemos seleccionado.

Al igual que en el robot biciclo las constantes que usaremos para seguir la trayectoria generada en el mapa 2 serán las mismas que hemos usado en el mapa 1, para de esta forma poder aceptar o no dichas constantes como constantes generales del robot diferencial en el algoritmo de grafos de visibilidad.

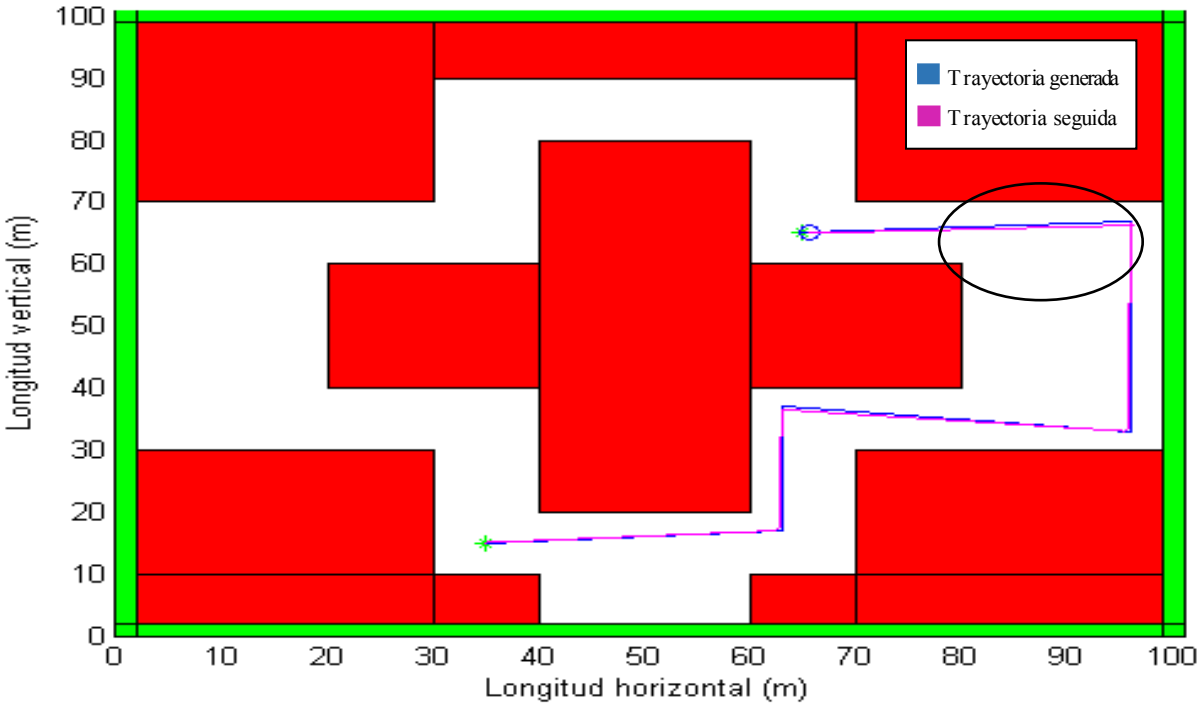


Figura 5-18. Seguimiento de trayectoria con robot diferencial en mapa 2 (GV).

Con esta imagen podemos confirmar que las constantes que hemos decidido usar para el robot diferencial en los grafos de visibilidad son válidas, ya que el camino seguido por el robot es una trayectoria, que a pesar de que en ciertos puntos difiera de la trayectoria generada, estas diferencias parecen a simple vista despreciables. Más adelante confirmaremos ampliando la zona marcada con un círculo negro, ya que es la que parece que posee mayores diferencias, que nuestras deducciones son correctas y que podemos despreciar dichas desigualdades.

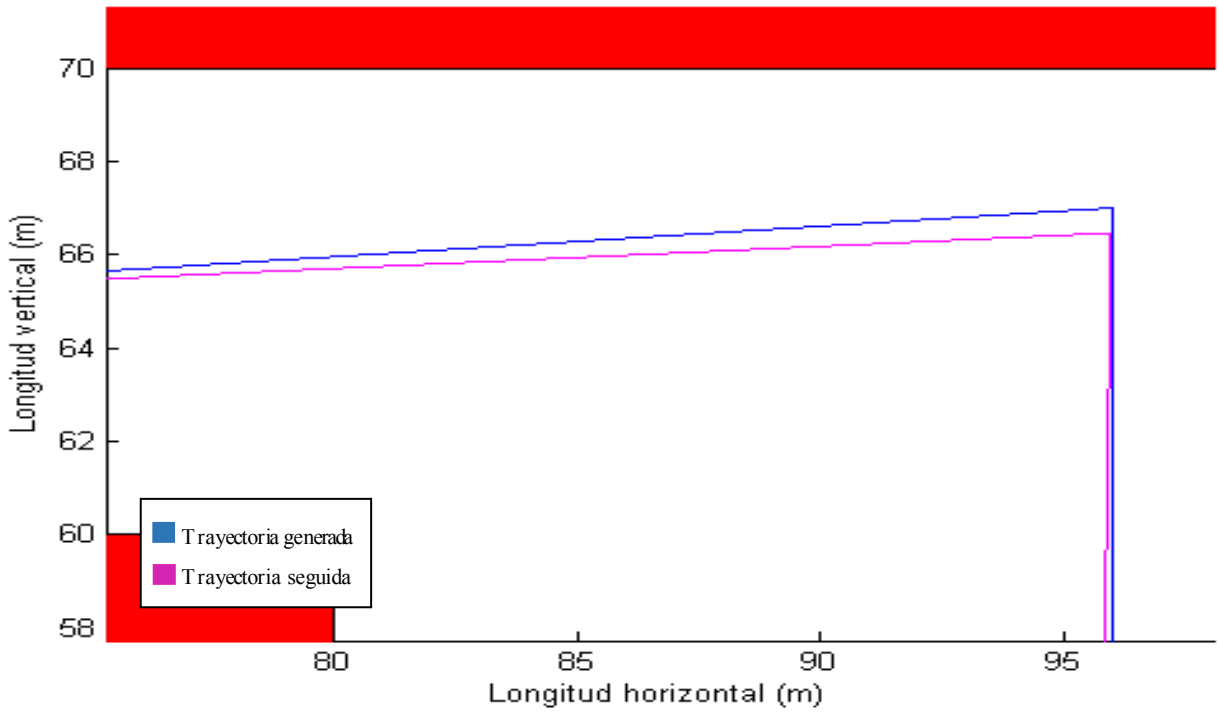


Figura 5-19. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 2 (GV).

Podemos ver tras haber ampliado la imagen anterior como apenas se separa de la trayectoria que este tiene que seguir, distanciándose un máximo de 0.5 metros. De todos modos esto no sería un inconveniente por el cual modificar las constantes asignadas.

- Robot síncrono

Como ya hemos dicho anteriormente, es necesario que cada robot posea unas constantes específicas en cada método, por ello las constantes que hemos decidido usar para que el robot síncrono siga la trayectoria generada por los grafos de visibilidad son las siguientes:

Tabla 5–3. Constantes del robot síncrono para GV

d	Kv	Kh	Ki
0	0.005	50	200

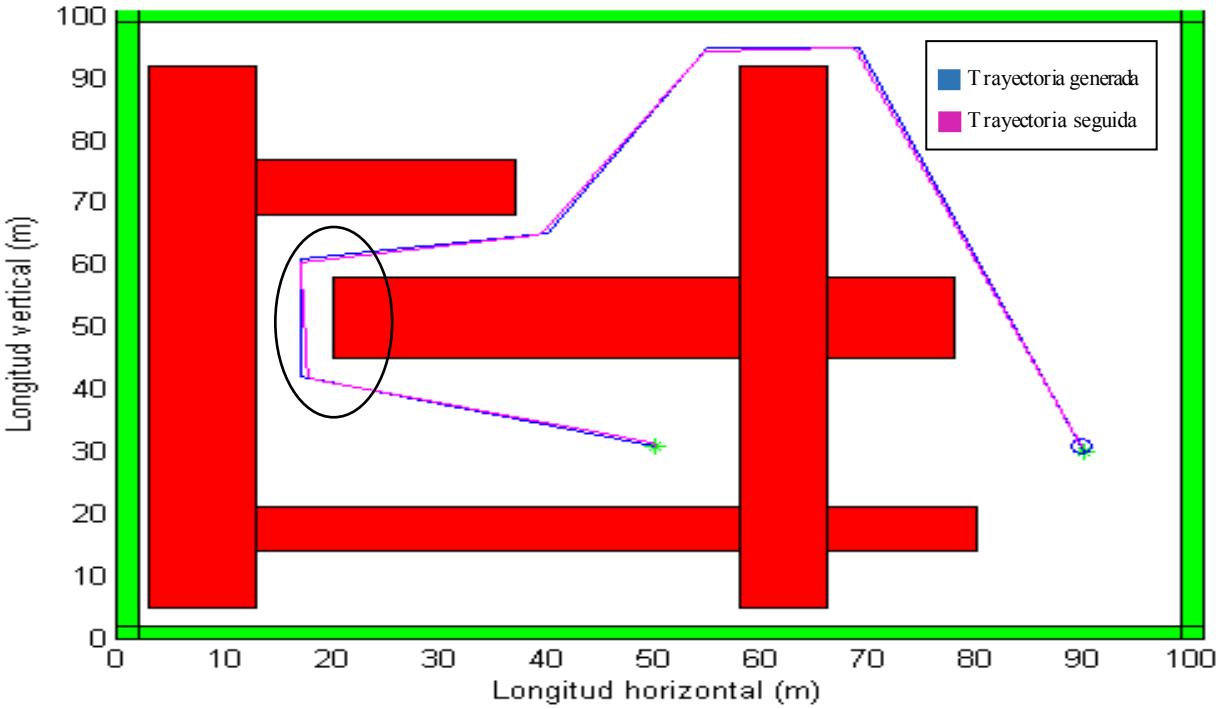


Figura 5-20. Seguimiento de trayectoria con robot síncrono en mapa 1 (GV).

La imagen anterior nos muestra como el robot sigue la trayectoria que hemos generado mediante los grafos de visibilidad con pequeñas variaciones y una velocidad similar a la obtenida en los robots diferenciales. Dichas variaciones no parecen a simple vista demasiado grandes como para que no sean válidas las constantes que hemos decidido usar, pero para asegurarnos de ellos ampliaremos la zona seleccionada, la cual es la que posee mayores variaciones y ahí comprobaremos si nuestras conclusiones iniciales son acertadas.

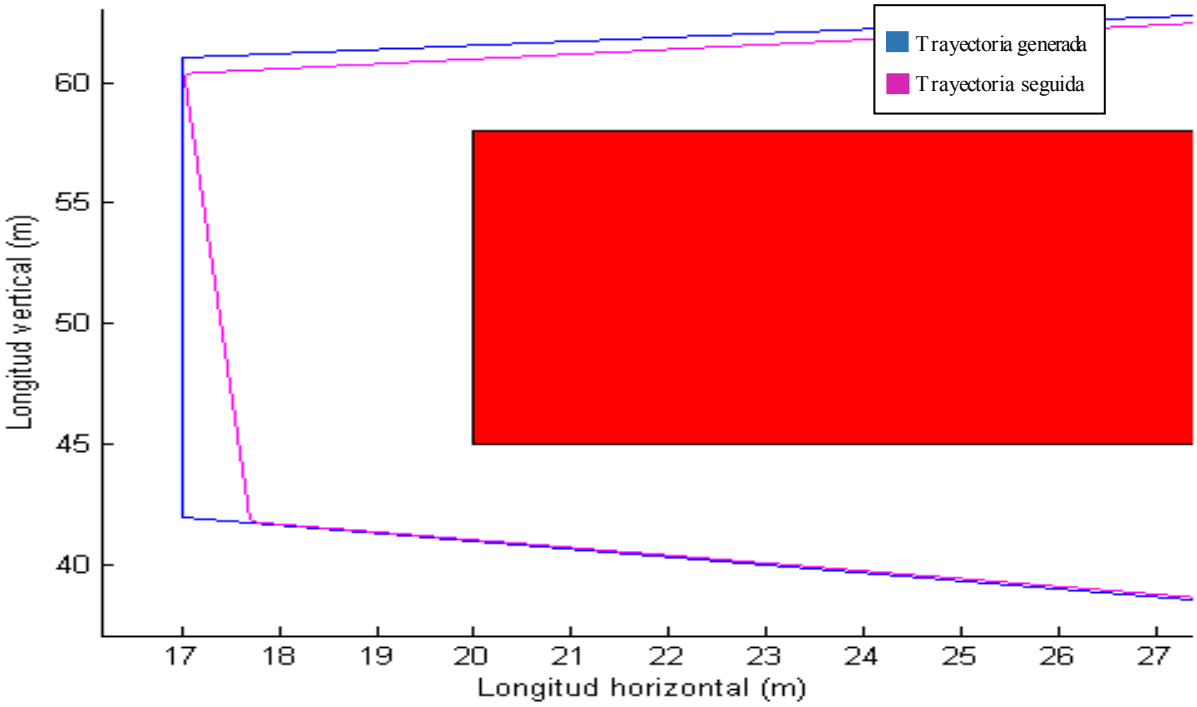


Figura 5-21. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 1 (GV).

De acuerdo con lo dicho antes de realizar la ampliación de la imagen, vemos como las constantes efectivamente son válidas, ya que a pesar de que la distancia que separa la trayectoria generada de la trayectoria seguida por el robot es de casi 0.8 metros, en los mapas que estamos usando no estamos barajando la posibilidad de tratar con caminos tan estrechos como para que esto fuese un problema, permitiendo así estas diferencias. Si tuviésemos en algún otro momento obstáculos que pudiesen estar más cerca, sería necesario cambiar dichos parámetros para así obtener un seguimiento más exacto.

Para asegurarnos de que las constantes que hemos decidido usar para el robot síncrono en el algoritmo que hemos empleado son constantes generales de dicho robot en los grafos de visibilidad, teniendo en cuenta el tipo de entornos en los que estamos trabajando, cambiaremos al mapa 2 donde podremos comprobar dicha premisa según los resultados obtenido.

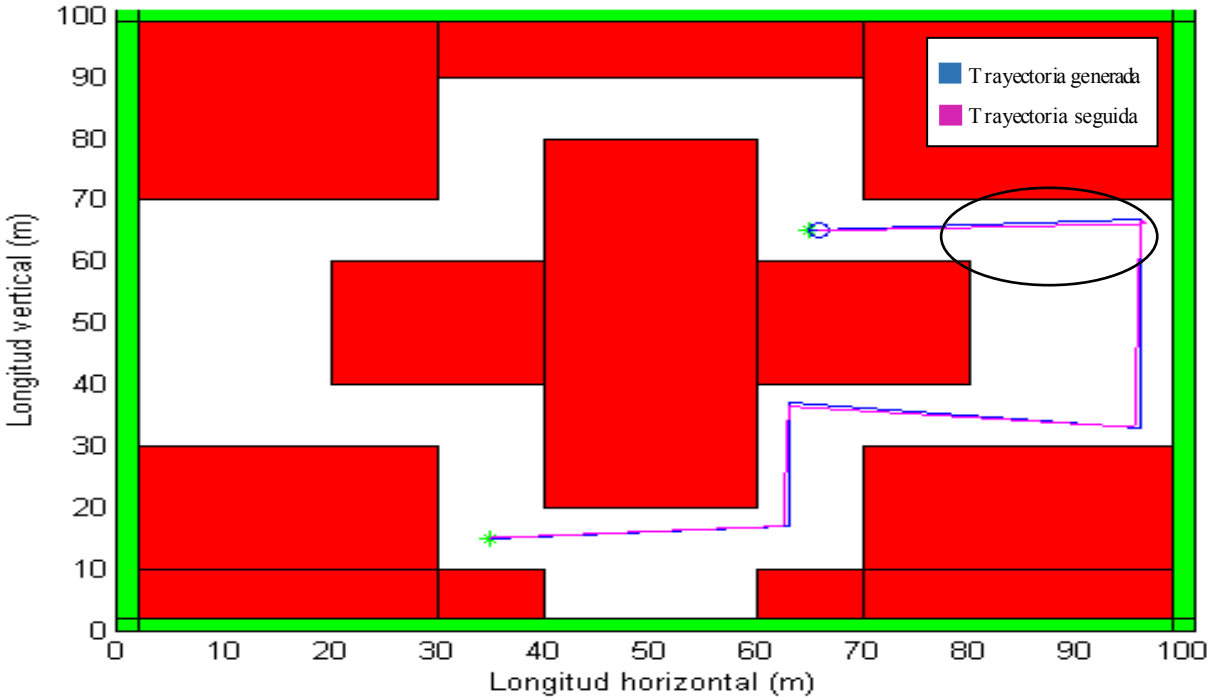


Figura 5-22. Seguimiento de trayectoria con robot síncrono en mapa 2 (GV).

En la imagen que vemos anteriormente observamos como efectivamente el robot sigue la trayectoria deseada con un margen de error, el cual era posible prever ya que en el mapa 1 vimos como con las constantes seleccionadas, ambas trayectorias no se seguían exactamente. Al igual que en las imágenes anteriores se realizará un zoom sobre la zona indicada, la cual es la considerada con mayores errores, para así poder ver con que errores estamos tratando.

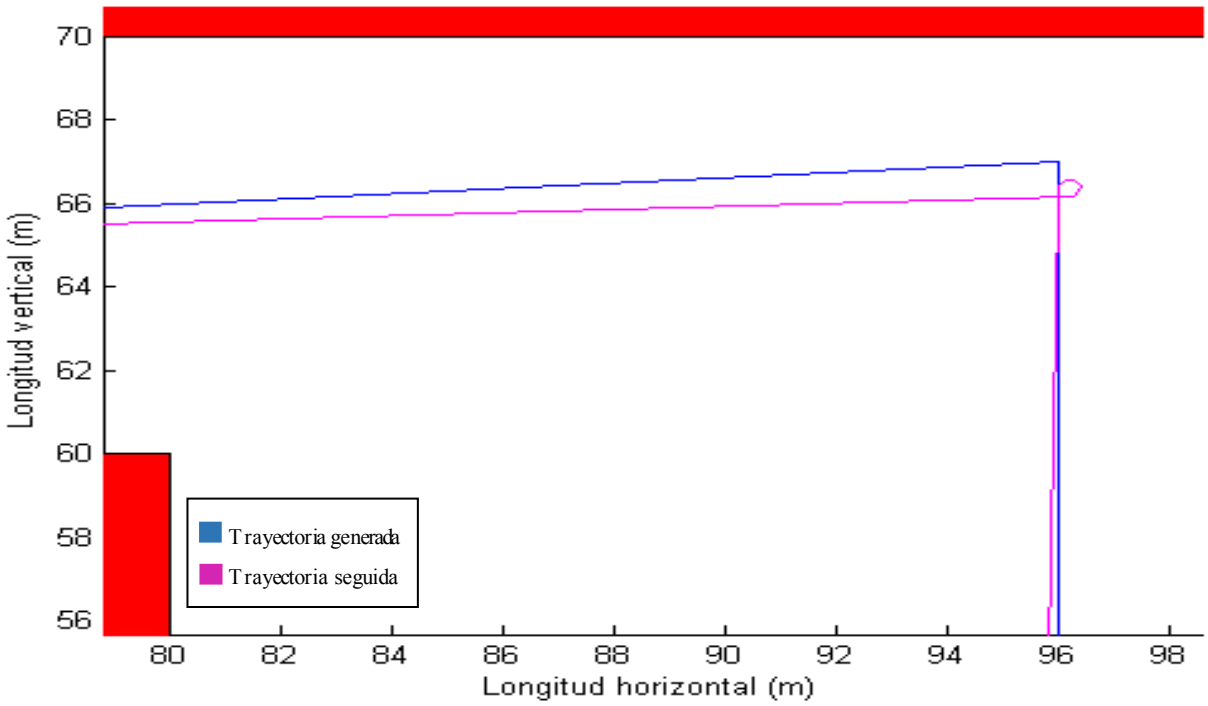


Figura 5-23. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 2 (GV).

En la imagen ampliada observamos como el robot al llegar a cierto punto gira en el sentido más largo, en lugar de realizar el giro más simple. Esto no presenta ningún inconveniente, ya que a pesar de realizar dichos giros la distancia máxima de separación entre ambas trayectorias es de aproximadamente 0.5 metros. Por lo tanto con esta ampliación confirmamos que las constantes asignadas, pueden suponerse como constantes globales del robot síncrono para los grafos de visibilidad.

- Robot triciclo

Para que este robot siga la trayectoria generada por el algoritmo grafos de visibilidad, es necesario establecer las constantes específicas de este robot para dicho algoritmo, las cuales son:

Tabla 5–4. Constantes del robot triciclo para GV

d	Kv	Kh	Ki
0	0.7	50	1

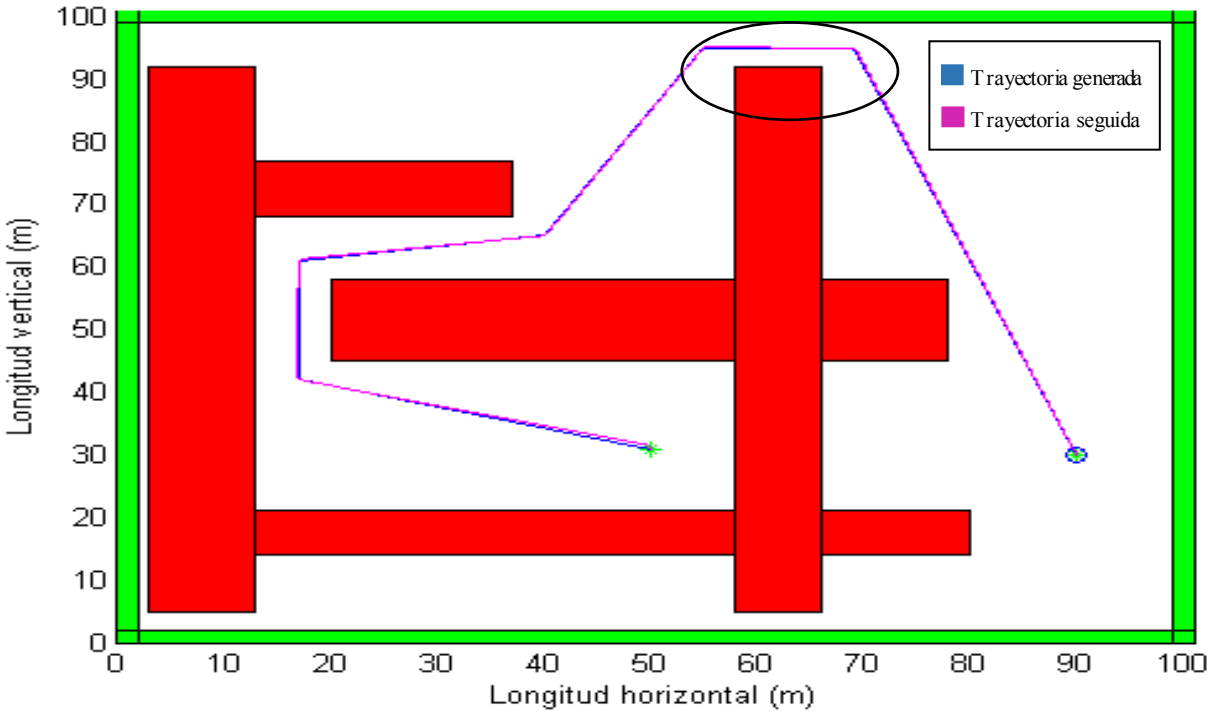


Figura 5-24. Seguimiento de trayectoria con robot triciclo en mapa 1 (GV).

La imagen anterior nos muestra como el robot sigue la trayectoria que hemos generado mediante los grafos de visibilidad con pequeñas variaciones casi inapreciables y una velocidad elevada, al igual que los robots biciclos. A pesar de que las variaciones parecen nulas, es preferible ampliar la imagen para así poder asegurarnos de que valores de errores estamos tratando, y así poder asegurarnos de si las constantes que hemos tomado son las adecuadas. La zona que hemos ampliado es la zona seleccionada al igual que en los robots anteriores.

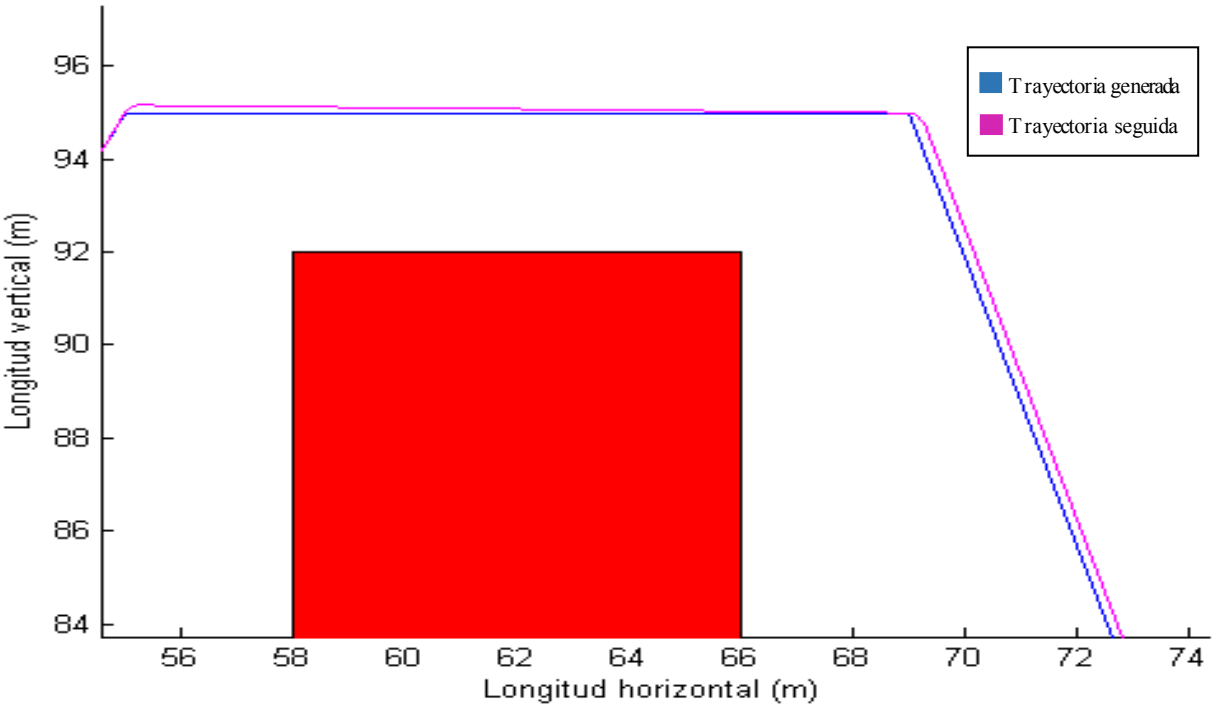


Figura 5-25. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 1 (GV).

Como ya preveíamos en la imagen sin ampliar, con esta imagen hemos podido confirmar como efectivamente las diferencias entre ambas trayectorias son de apenas 0.1 metros, de forma que podemos confirmar que las constantes usadas serán válidas a falta de probar el robot con las mismas constantes en otro mapa para confirmar nuestras predicciones.

A continuación realizaremos el seguimiento de trayectoria del robot triciclo con las mismas constantes empleadas en el mapa 1 pero en este caso en el mapa 2, para poder confirmar, como hemos mencionado anteriormente, que las constantes aplicadas son constantes generales del robot triciclo en los grafos de visibilidad.

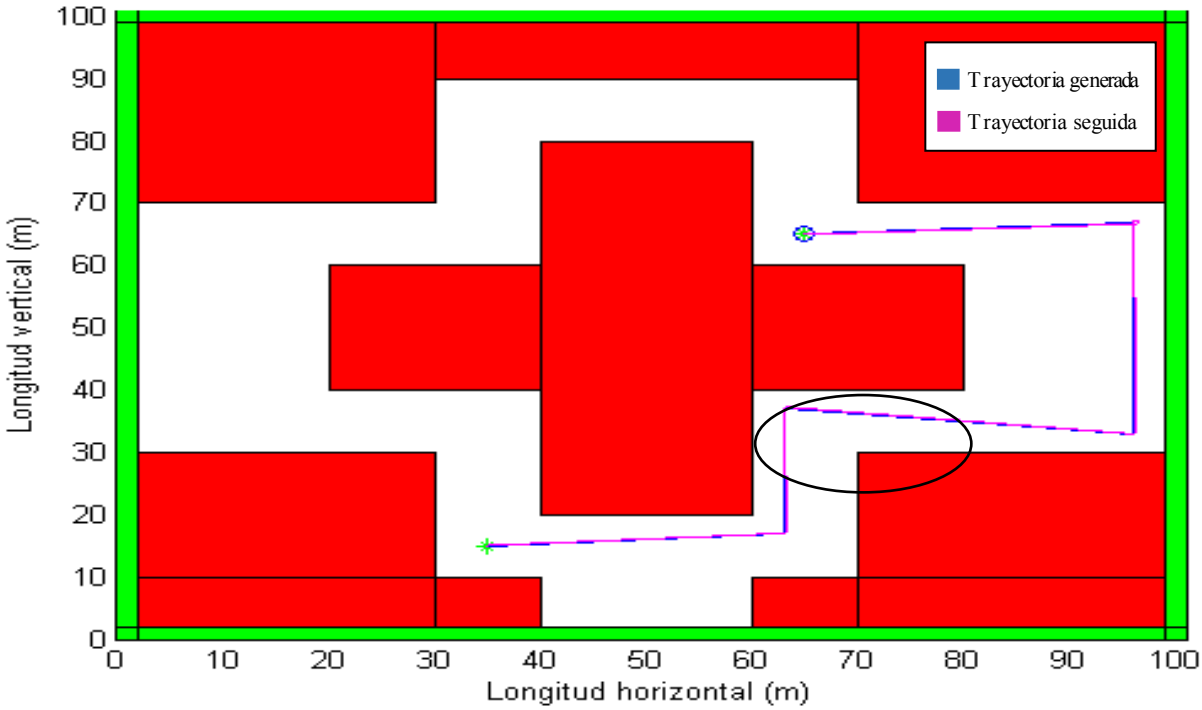


Figura 5-26. Seguimiento de trayectoria con robot triciclo en mapa 2 (GV).

Como ya imaginábamos, observamos en la imagen como el seguimiento de la trayectoria de robot es bastante fiel. A simple vista apenas podríamos diferenciar ambas trayectorias, por lo que esto confirmaría que las constantes usadas son válidas para este robot en los grafos de visibilidad. A pesar de ello vamos a realizar una ampliación en la zona seleccionada para ver cuánto difieren las trayectorias.

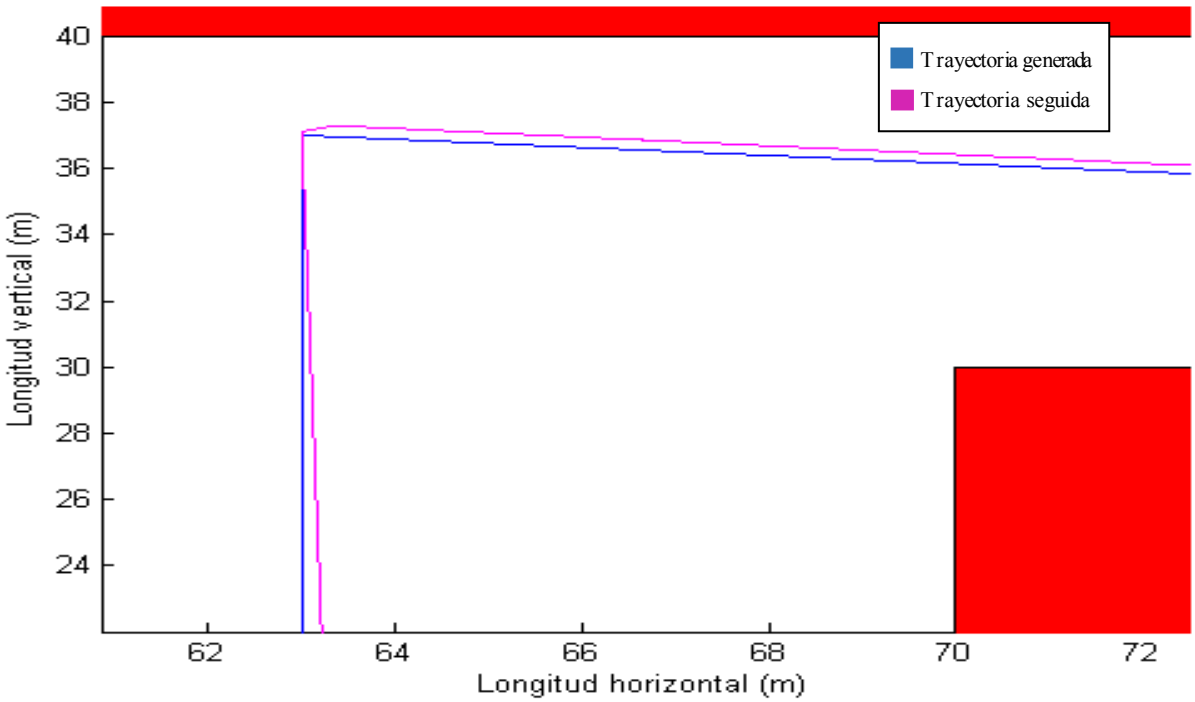


Figura 5-27. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 2 (GV).

Como era de esperar a pesar de haber ampliado la zona más conflictiva, las variaciones de ambas trayectorias no son mayores de 0.2 metros. Con esto confirmamos definitivamente nuestras sospechas con respecto a las constantes usadas y nos aseguramos de tener un seguimiento bastante próximo, de forma que no tendríamos problemas incluso en mapas más conflictivos.

5.2.2 Bug 2

- Robot biciclo

Al igual que en los grafos de visibilidad, para un seguimiento de trayectorias óptimo en el algoritmo Bug 2 con el robot biciclo, es necesario aplicar unas constantes específicas. Las cuales comprobaremos más adelante que son válidas para este robot y método. Dichas constantes son las siguientes:

Tabla 5–5. Constantes del robot biciclo para Bug 2

d	Kv	Kh	Ki
0	0.05	10	1000

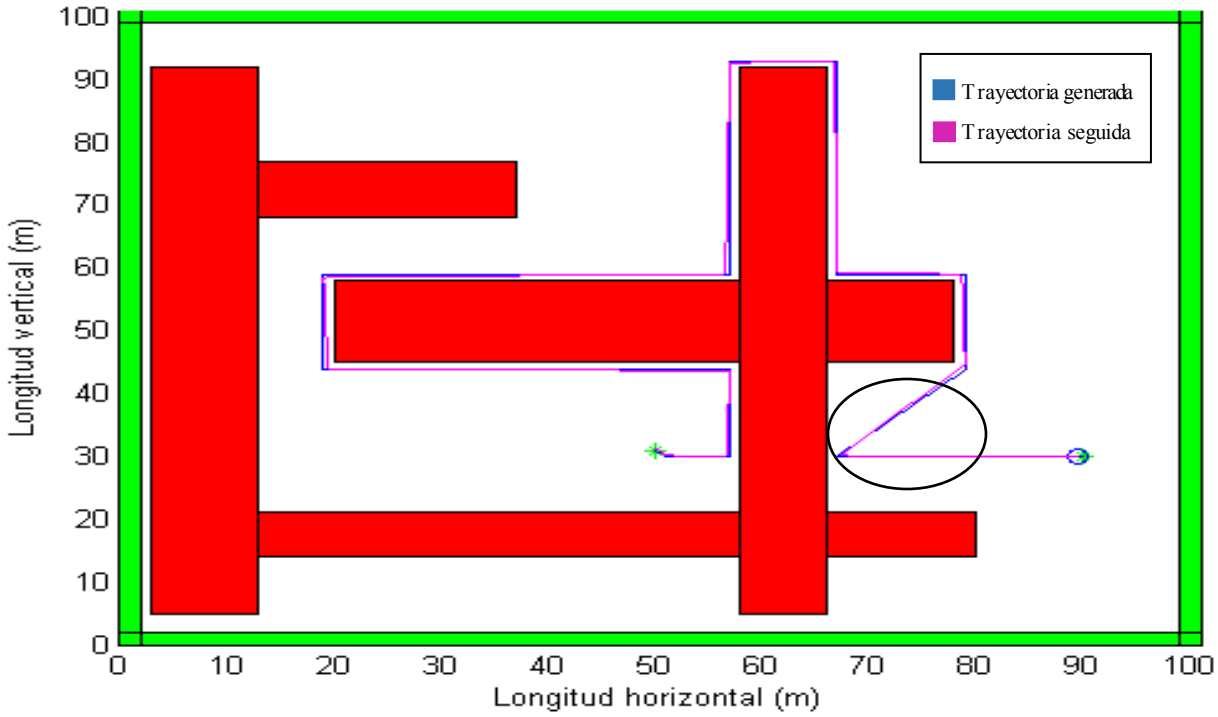


Figura 5-28. Seguimiento de trayectoria con robot biciclo en mapa 1 (Bug2).

Como podemos ver en la imagen, el robot sigue la trayectoria deseada, aunque observando la imagen podemos apreciar pequeñas variaciones entre ambas, a pesar de ellos, dichas variaciones no parece que provoquen colisiones con los obstáculos, aun así sería conveniente ampliar la imagen en la zona seleccionada ya que se considera la más perjudicada. Otra característica que hemos podido apreciar con este robot en el algoritmo Bug 2, es que la velocidad con la que el robot seguía la trayectoria generada era bastante más lenta que en los grafos de visibilidad. Esto era previsible ya que hemos disminuido considerablemente la constante de velocidad.

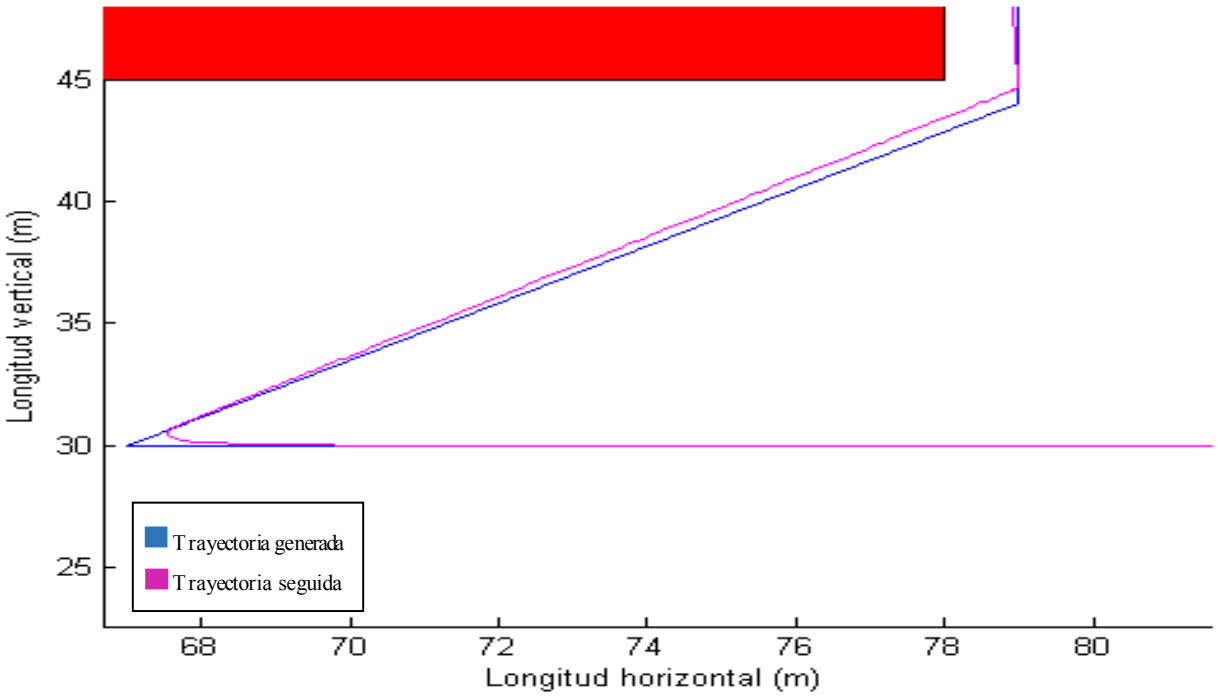


Figura 5-29. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 1 (Bug2).

Tras observar la ampliación de la imagen podemos ver como en una zona crítica la trayectoria realizada por el robot no se separa considerablemente de la trayectoria deseada, apenas se separa unos 0.3 metros, esta desviación es natural debido al tipo de giro que el robot debe realizar. A pesar de ello ya que nuestro robot mide 0.4 metros de diámetro, no tendríamos ningún problema referente a colisiones.

Ya que con este mapa no queda totalmente asegurada una circulación sin colisiones en todos los entornos que podamos generar, nos vemos en la necesidad de probar con otro mapa para asegurarnos de que las constantes escogidas son las adecuadas.

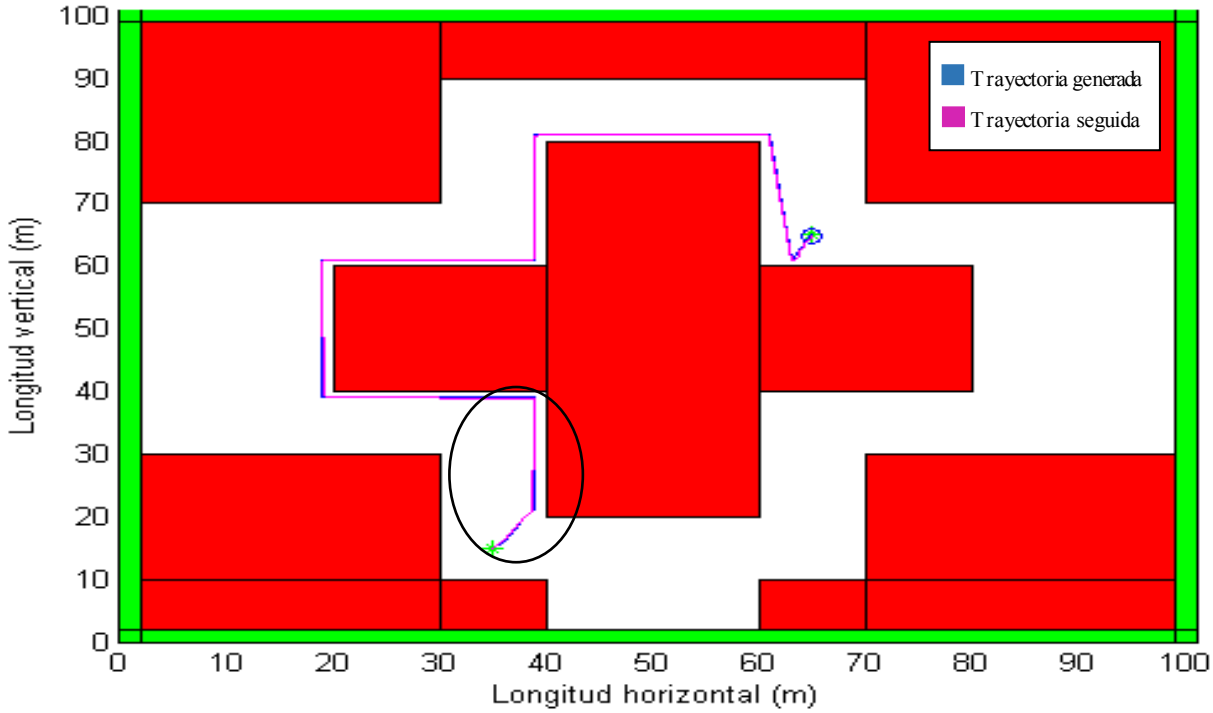


Figura 5-30. Seguimiento de trayectoria con robot biciclo en mapa 2 (Bug2).

Como vemos en este imagen en el mapa 2 podemos observar cómo se sigue la trayectoria de forma rigurosa en todo el tramo, excepto en la primera parte en la que podemos ver como a pesar de que el robot sigue la trayectoria deseada, encontramos una pequeña disparidad. Para poder asegurarnos de que dichas diferencias no causarán ningún problema al robot biciclo, vamos a realizar zoom sobre dicha zona.

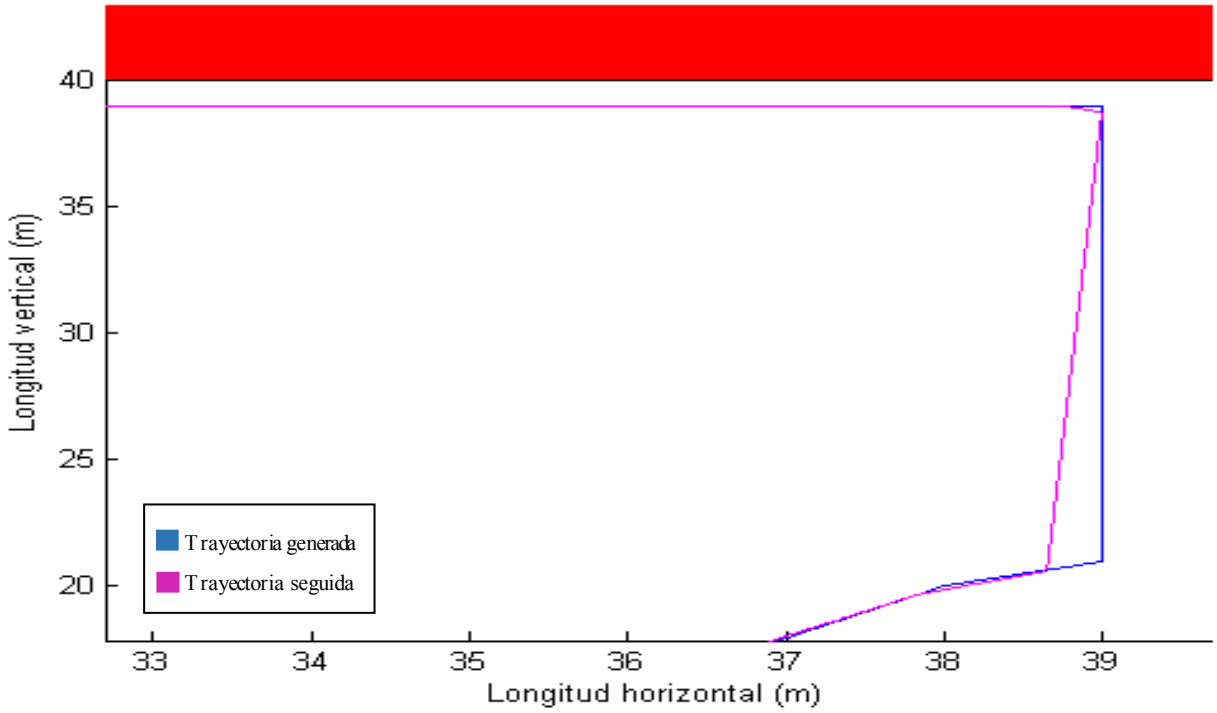


Figura 5-31. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 2 (Bug2).

Al ampliar sobre la zona marcada, vemos como las diferencias entre las trayectorias no son excesivamente grandes, aproximadamente de 0.5 metros. Dichas diferencias las encontramos a la hora de hacer uno de los giros, ya que en lugar de girar por la zona más corta, el robot gira por la zona de mayor ángulo. A pesar de ello no es algo que deba preocuparnos, ya que nuestro robot al ser de tamaño pequeño, no sufre ninguna colisión. Por estos motivos damos por válidas las constantes usadas para el robot biciclo en el algoritmo de Bug 2.

- Robot diferencial

  En esta ocasión el robot diferencial es el que para seguir la trayectoria generada por el algoritmo de Bug 2 necesita unas constantes, las cuales verificaremos analizando los resultados obtenidos. Dichas constantes son las siguientes:

Tabla 5–6. Constantes del robot diferencial para Bug 2

d	Kv	Kh	Ki
0	0.005	10	800

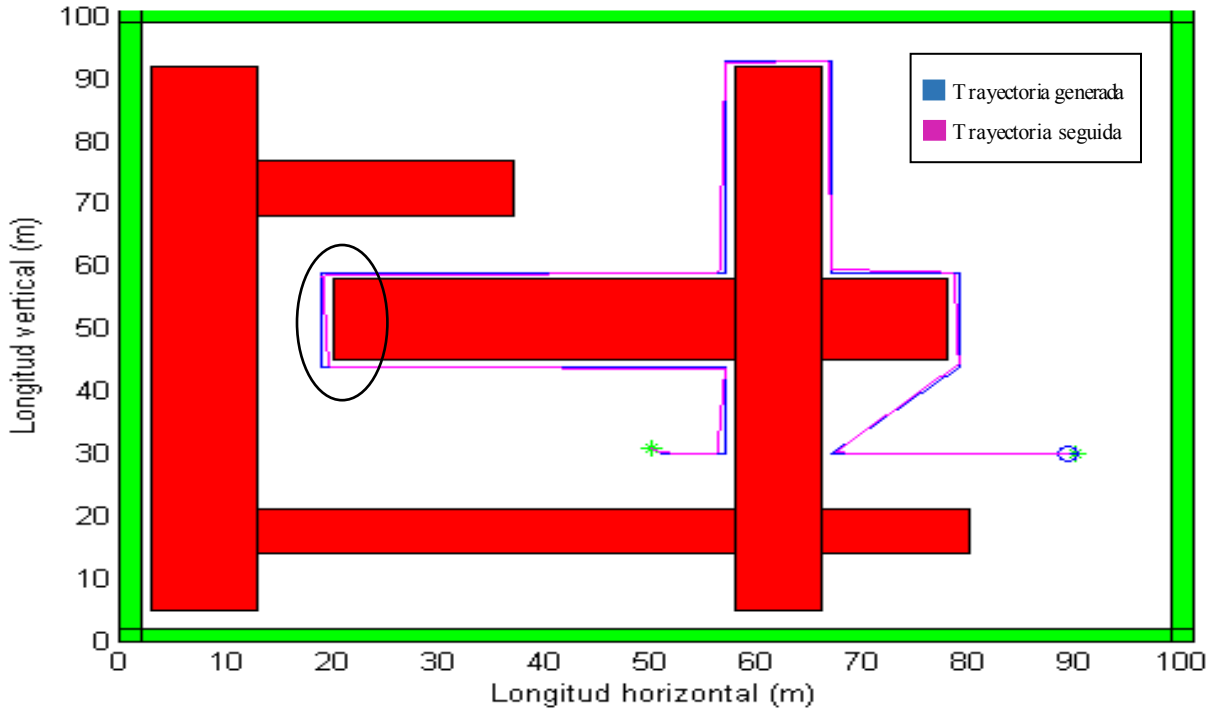


Figura 5-32. Seguimiento de trayectoria con robot diferencial en mapa 1 (Bug2).

En la imagen anterior vemos como la trayectoria es seguida de forma adecuada, presentando ciertas diferencias en algunos puntos localizados, a pesar de ellos estás diferencias parecen mínimas y en ningún momento se acerca a sufrir una colisión debido a ellas. Para asegurarnos de ello al igual que en los anteriores métodos realizaremos zoom sobre la zona seleccionada. Además de lo dicho anteriormente observamos una disminución de la velocidad en comparación con el robot biciclo, debida a que hemos disminuido la constante de velocidad.

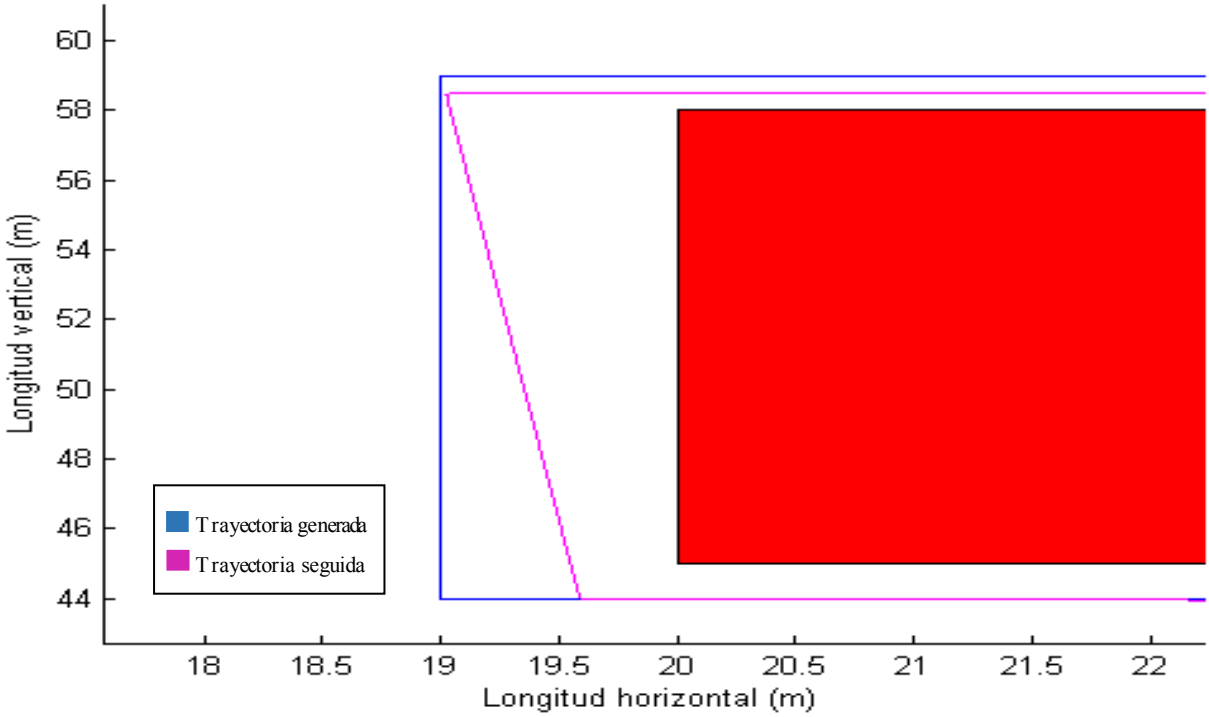


Figura 5-33. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 1 (Bug2).

Tras realizar la ampliación, podemos ver como efectivamente el robot sigue bastante de cerca a la trayectoria que hemos generado, excepto cuando tiene que realizar una maniobra algo más compleja, justo antes de iniciar dicha maniobra el robot se separa unos 0.6 metros y tras realizarla se distancia un máximo de 0.3 metros. Por lo que corroboramos que el robot es capaz de seguir la trayectoria de manera fiable.

Para asegurarnos de que variando el mapa el robot sigue realizando bien las trayectorias con las mismas constantes, hemos realizado el mismo experimento en el mapa 2.

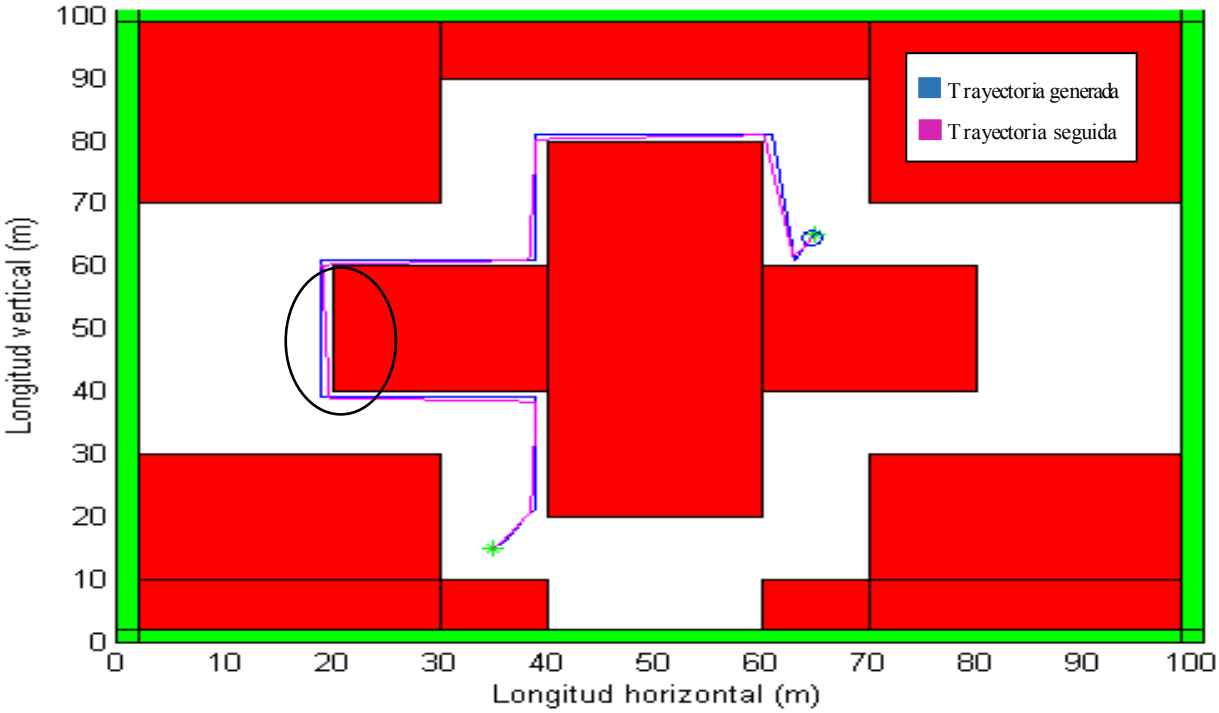


Figura 5-34. Seguimiento de trayectoria con robot diferencial en mapa 2 (Bug2).

Tras la ejecución del seguimiento del robot diferencial a la trayectoria generada por el algoritmo de Bug 2 en el mapa 2, vemos como ciertamente al igual que el otro mapa se sigue de manera fiable la trayectoria, excepto en ciertas zonas en la que el error que obtenemos no es excesivo. Tan solo encontramos una zona donde efectivamente podría haber una colisión, la cual es la zona señalada que ampliaremos para confirmar nuestras suposiciones.

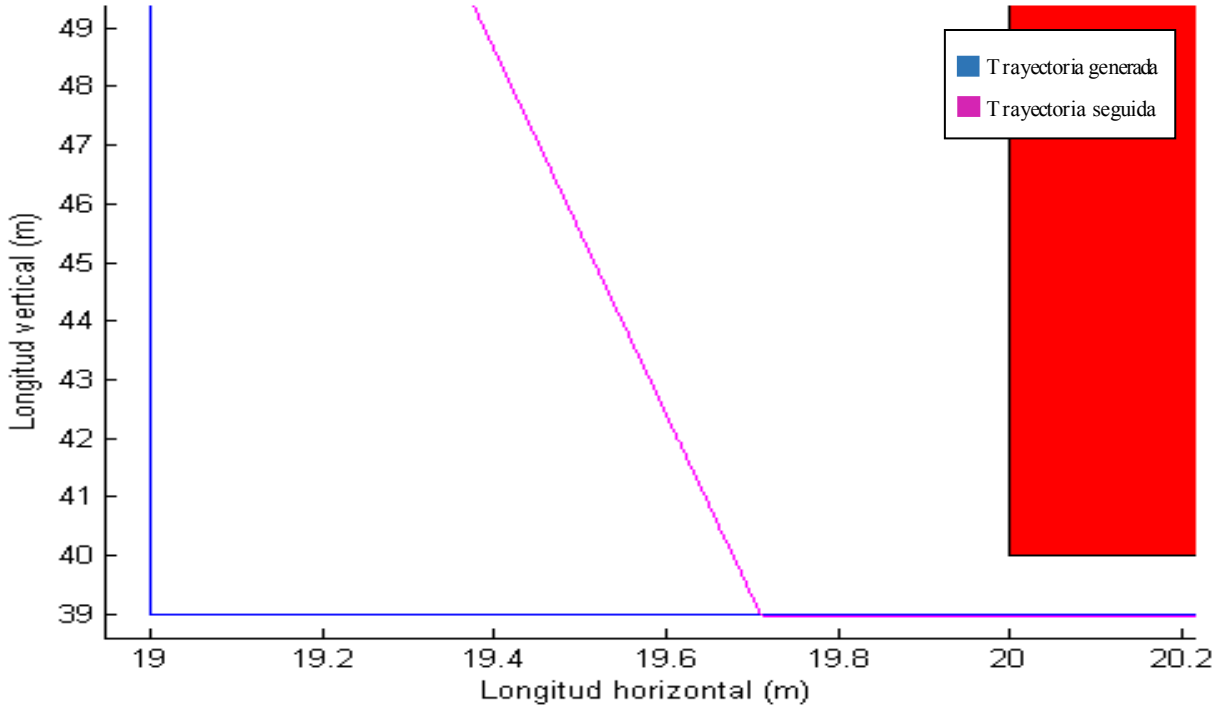


Figura 5-35. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 2 (Bug2).

A diferencia de lo que en la imagen anterior pudiese parecer, tras realizar la ampliación de la zona marcada vemos como se separa 0.7 metros de la trayectoria que debería de seguir, quedándose a una distancia de más de 0.4 metros de un obstáculo, más que suficiente para que no haya colisión. Como sabemos nuestro robot tiene 40 cm de diámetro, por lo que como la trayectoria del robot se traza desde el centro de él, para que se chocase la distancia que lo separase de la pared debería ser de menos de 0.2 metros.

- Robot síncrono

  En este punto es necesario conocer las constantes que le introduciremos al robot síncrono para seguir la trayectoria generada por el algoritmo de Bug 2. Dichas constantes serán las siguientes:

Tabla 5–7. Constantes del robot síncrono para Bug 2

d	Kv	Kh	Ki
0	0.005	10	800

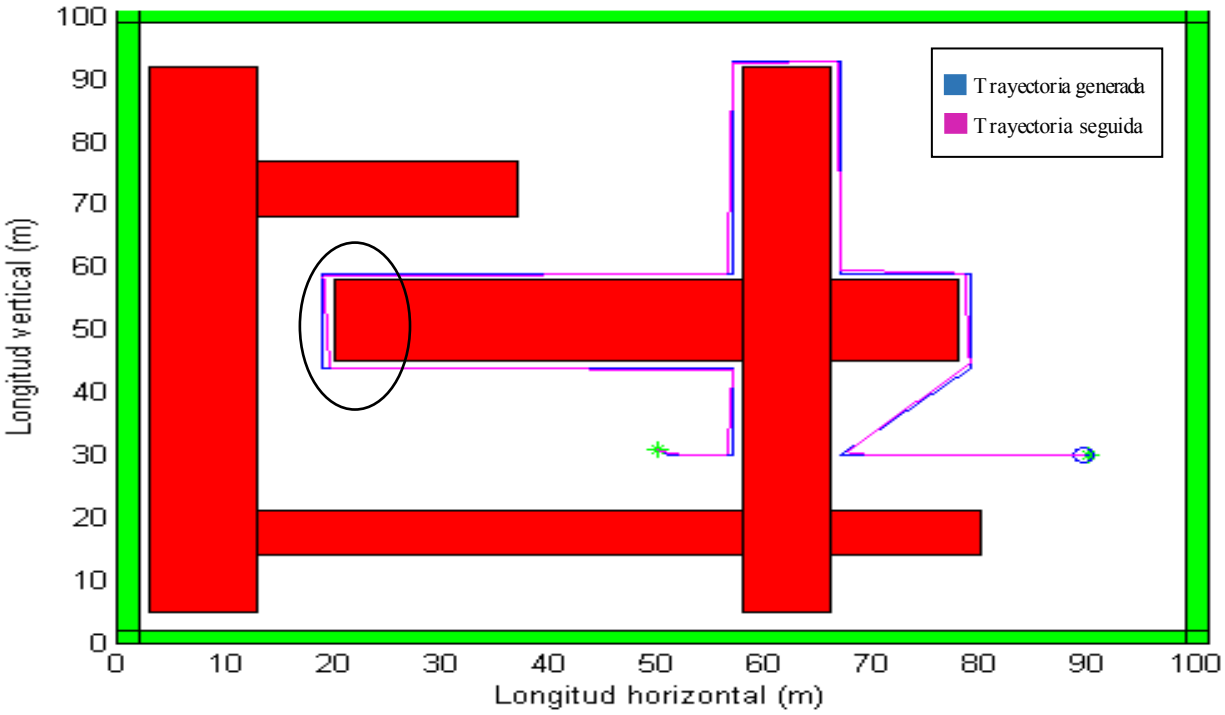


Figura 5-36. Seguimiento de trayectoria con robot síncrono en mapa 1 (Bug2).

Vemos en la imagen anterior cómo, efectivamente el robot sigue la trayectoria generada con unas desviaciones lo suficientemente pequeñas a simple vista como para que no se produzca una colisión. Para asegurarnos de ello ampliaremos la zona seleccionada, ya cual es la que nos ha parecido más apropiada. En cuanto a la velocidad con la que el robot recorre el mapa podríamos decir que es bastante similar a la del robot diferencial.

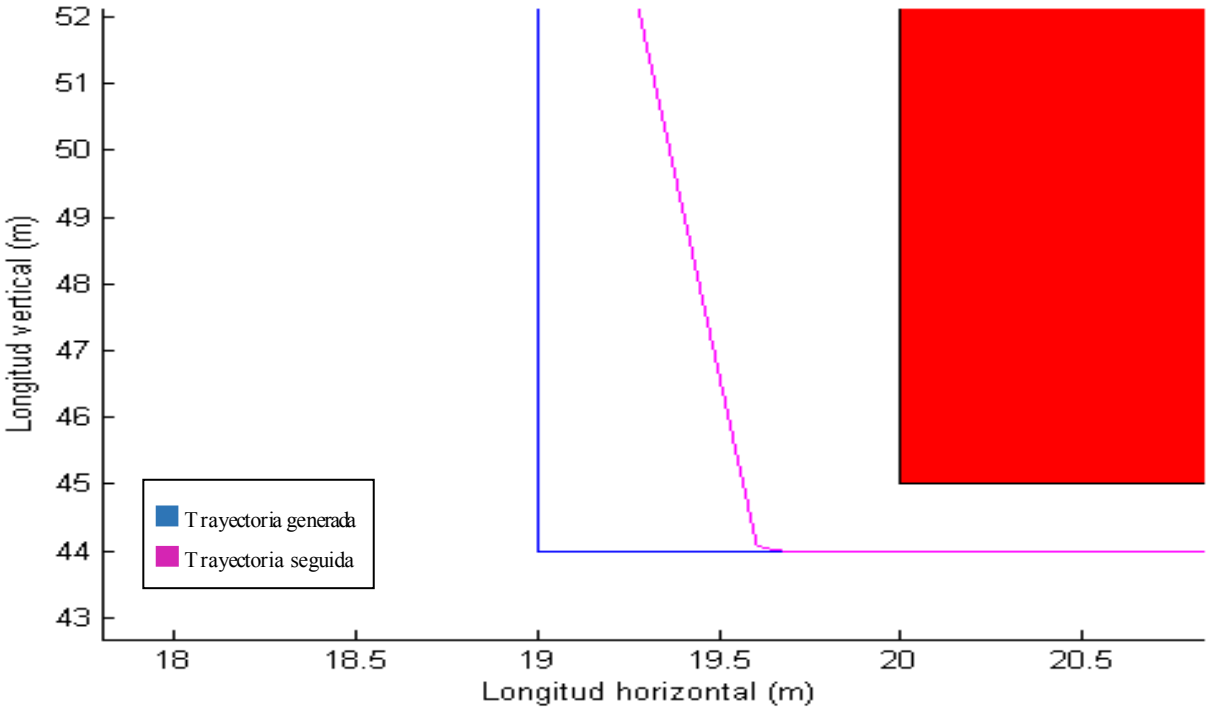


Figura 5-37. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 1 (Bug2).

Como podemos comprobar en la imagen, en la zona seleccionada el robot síncrono realiza un giro bastante

similar al realizado por el robot diferencial. Lo cual al igual que con el robot diferencial la trayectoria que obtenemos es una trayectoria válida.

Tras comprobar como las constantes asignadas han dado buen resultado en el mapa 1, realizaremos una simulación en el mapa 2 para así ser capaz de asegurarnos de que dichas constantes son válidas para cualquier mapa que podamos generar el cual posea una trayectoria generada por el algoritmo de Bug 2.

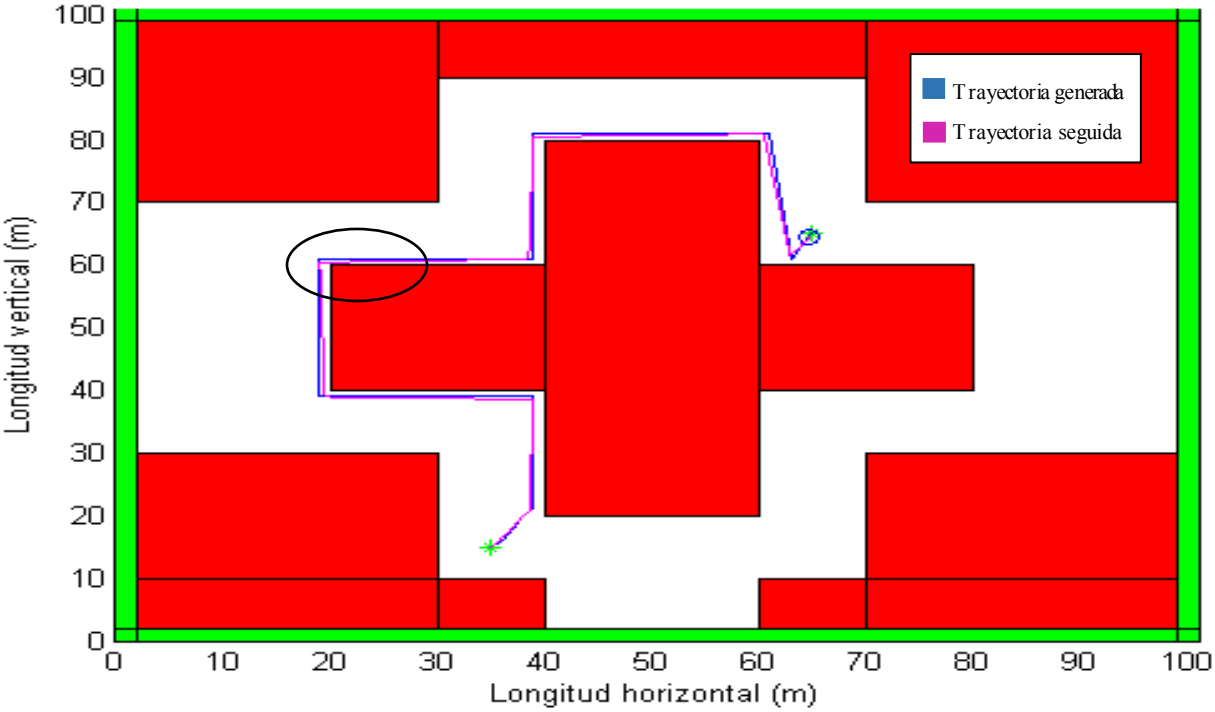


Figura 5-38. Seguimiento de trayectoria con robot síncrono en mapa 2 (Bug2).

Como comprobamos en la imagen obtenida, el seguimiento a la trayectoria del robot síncrono es bastante exacto, excepto en ciertas zonas, las cuales a pesar de tener un error, este no es lo suficientemente grande como para que nos cause algún problema, esto lo veremos más claramente en la imagen posterior que será un zoom de la zona seleccionada en la imagen actual. Por lo que aceptaríamos las constantes elegidas como constantes del robot síncrono para el algoritmo de Bug 2.

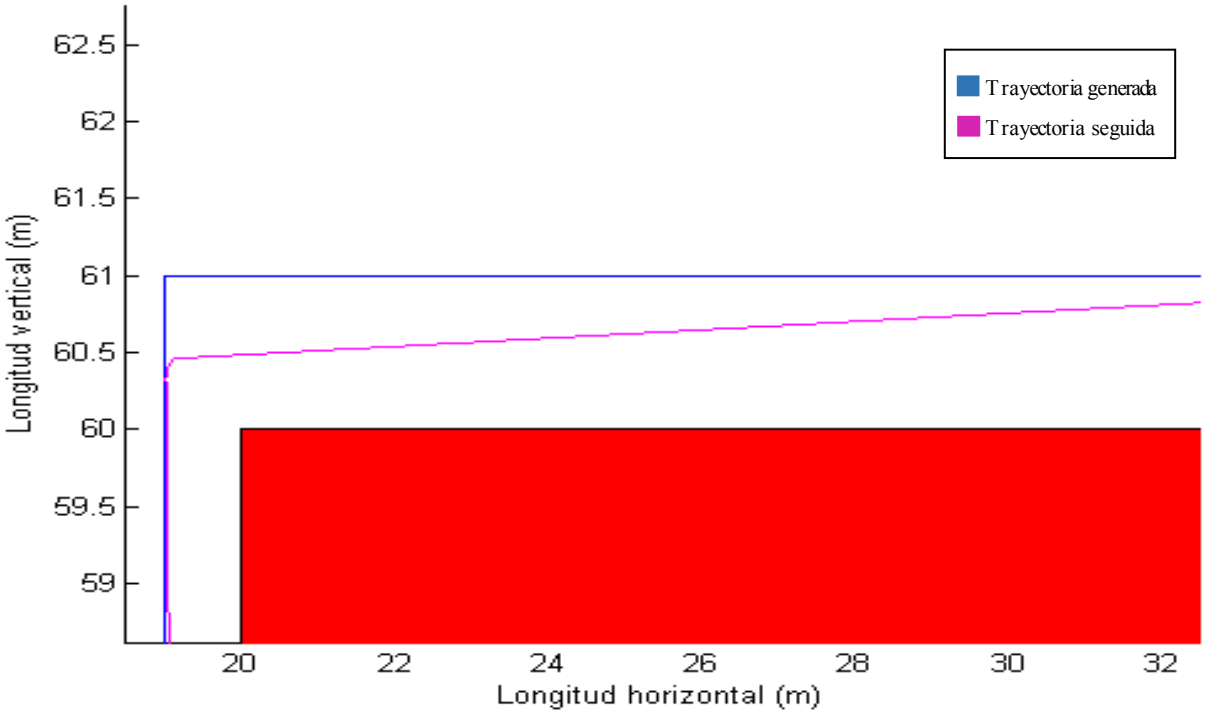


Figura 5-39. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 2 (Bug2).

Como ya esperábamos, en el punto seleccionado las trayectorias difieren un poco, pero no lo suficiente como para que esto sea un inconveniente como hemos dicho antes.

- Robot triciclo

  A continuación pasaremos a mostrar las contantes del robot triciclo para que este siga la trayectoria generada por Bug 2. Las cuáles serán las siguientes:

Tabla 5–8. Constantes del robot triciclo para Bug 2

d	Kv	Kh	Ki
0	0.05	10	1000

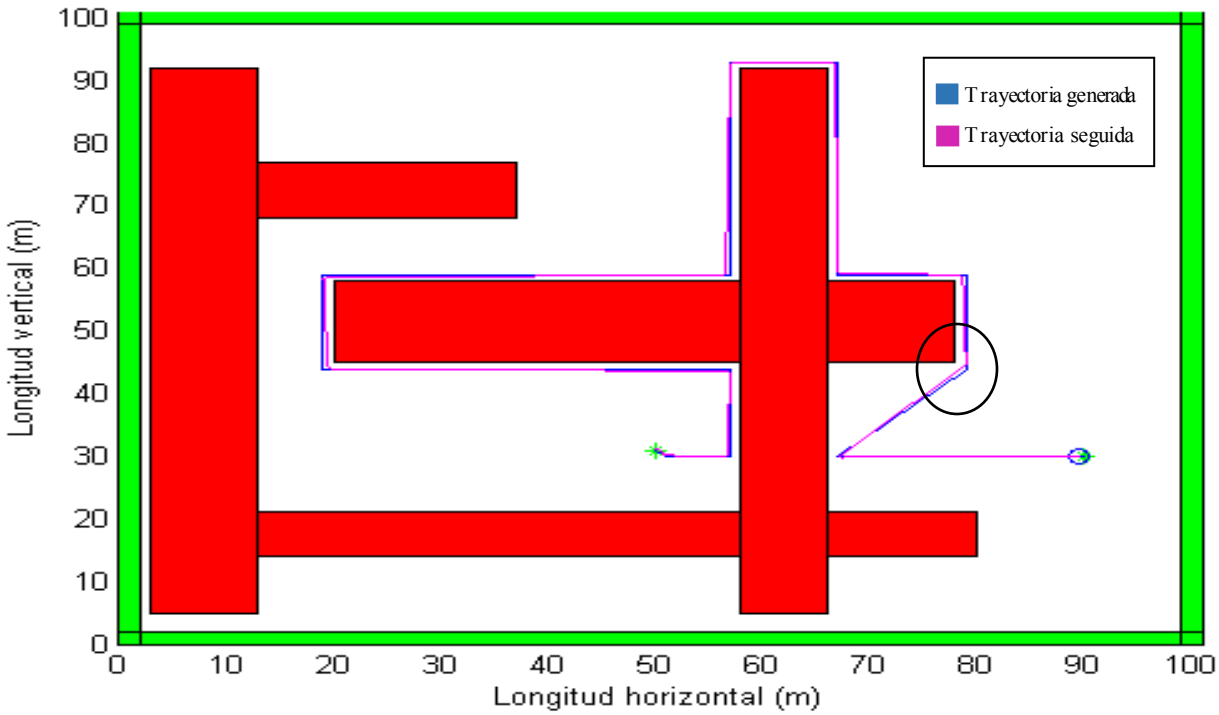


Figura 5-40. Seguimiento de trayectoria con robot triciclo en mapa 1 (Bug2).

En la imagen anterior podemos observar como nuestro robot triciclo sigue la trayectoria deseada con una desviación aparentemente aceptable. Al igual que hemos realizado en las imágenes anteriores, a continuación, se realizara un zoom sobre la zona remarcada para asegurarnos de que se cumple lo anteriormente predicho. Además de esto, hemos podido comprobar tras hacer la simulación como la velocidad con la que el robot recorre la trayectoria es mayor que en el caso del diferencial y del síncrono y similar al biciclo.

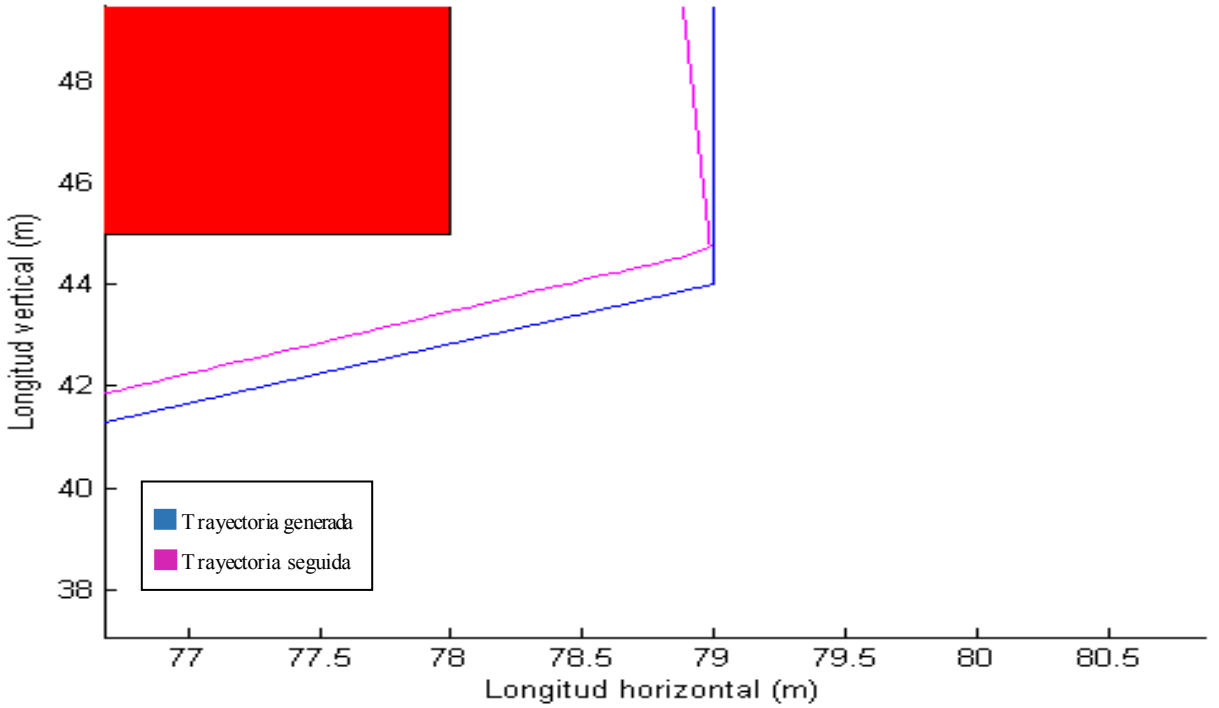


Figura 5-41. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 1 (Bug2).

Vemos como efectivamente la trayectoria es adecuada, ya que debido al reducido tamaño de nuestro robot no se produce colisión con las paredes.

Tras observar como con las constantes seleccionadas el robot recorre de forma correcta el mapa 1, se pasará a confirmar si con dichas constantes también se recorre la trayectoria adecuadamente en el mapa 2, y así poder afirmar que con dichas constantes el robot triciclo recorrerá adecuadamente cualquier trayectoria generada por el algoritmo de Bug 2.

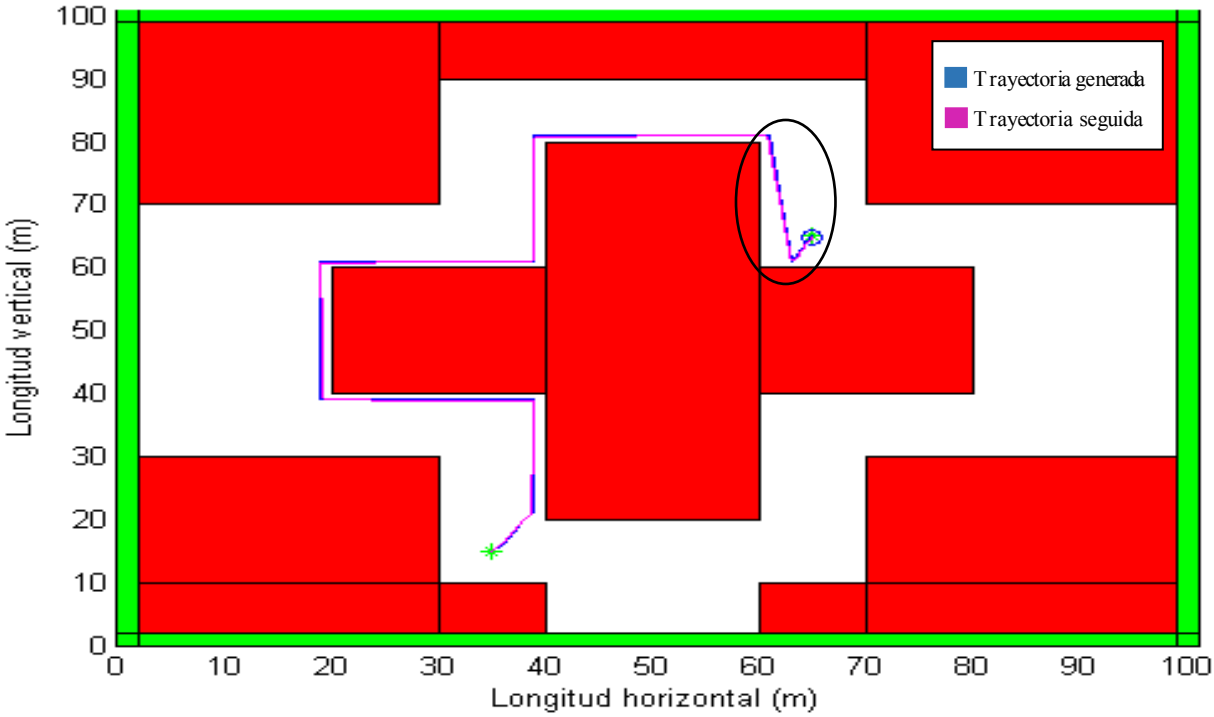


Figura 5-42. Seguimiento de trayectoria con robot triciclo en mapa 2 (Bug2).

Observamos en esta imagen como a simple vista la trayectoria que sigue el robot es la trayectoria adecuada. Para asegurarnos de ello ampliaremos la zona marcada y así comprobaremos que las constantes elegidas son las adecuadas.

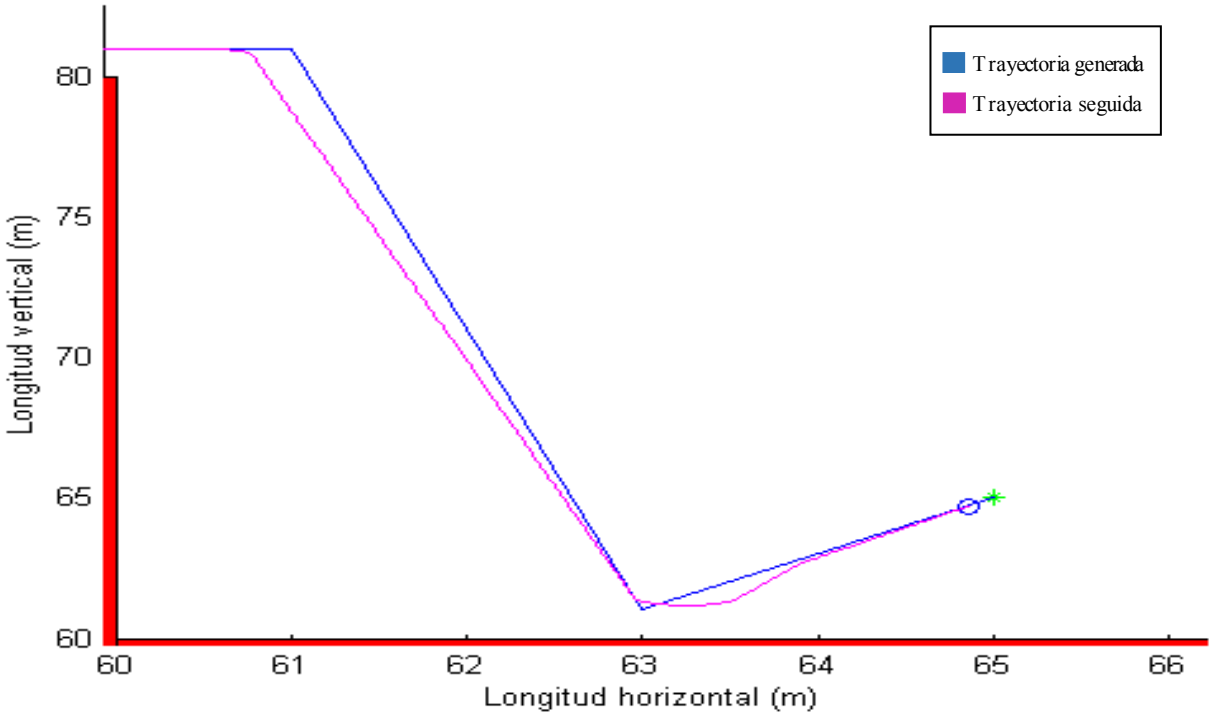


Figura 5-43. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 2 (Bug2).

Tras ampliar la imagen vemos como efectivamente la trayectoria es válida, ya que en el peor de los casos, el robot se queda a una distancia de aproximadamente 0.5 m. De esta forma corroboramos que las constantes elegidas son las adecuadas.

5.2.3 Diagramas de Voronoi

- Biciclo

  En la tabla siguiente mostramos las constantes elegidas para que el robot biciclo siga la trayectoria generada por los diagramas de Voronoi, las cuales comprobaremos más adelante que son las adecuadas.

Tabla 5–9. Constantes del robot biciclo para diagramas de Voronoi

d	Kv	Kh	Ki
0	0.1	10	300

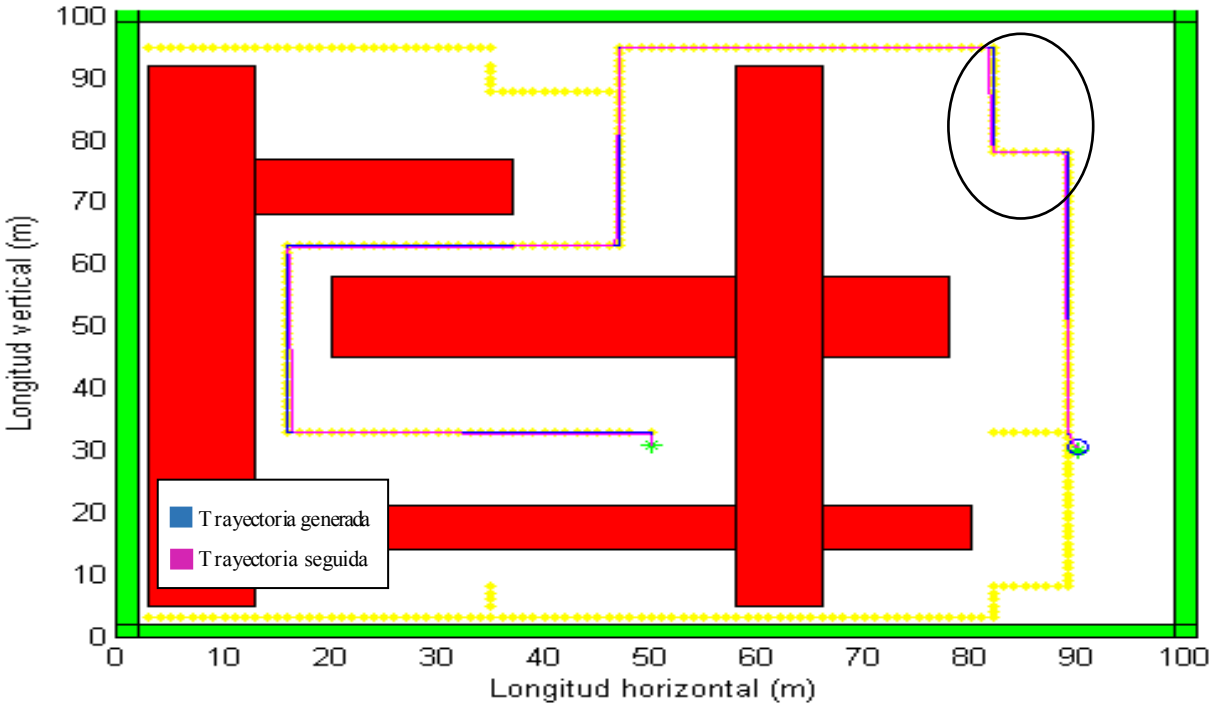


Figura 5-44. Seguimiento de trayectoria con robot biciclo en mapa 1 (D.Voronoi).

Observamos en la imagen anterior como sin ninguna duda la trayectoria que el robot sigue es válida, ya que no hay ningún punto de la misma está cerca a ningún obstáculo. A pesar de ello hemos hecho zoom sobre la zona seleccionada para ver cuánto difiere la trayectoria seguida de la deseada. Una característica que podemos apreciar es que el robot se desplaza más rápido en este método que en los diagramas de Bug2, aunque no más que en los algoritmos de visibilidad.

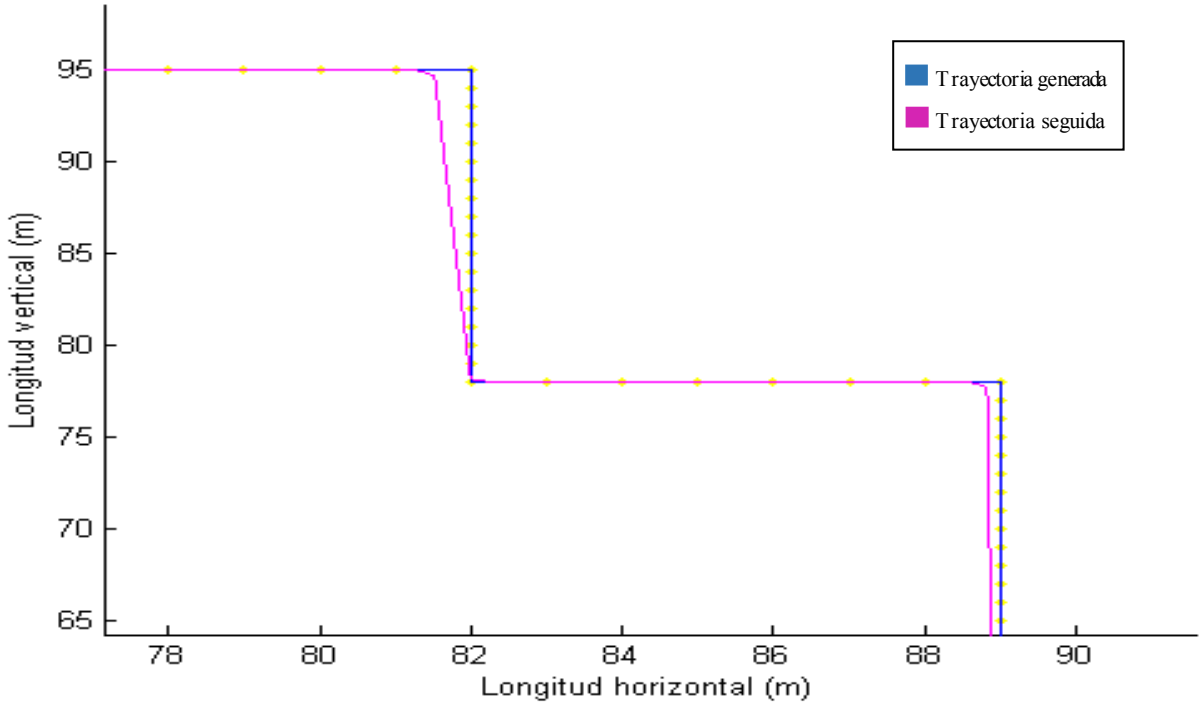


Figura 5-45. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 1 (D.Voronoi).

Tras ampliar la imagen podemos ver como la máxima distancia que el robot se separa de la trayectoria deseada es de aproximadamente 0.8 metros.

A pesar de que no cabe duda de que las constantes elegidas serán válidas para cualquier trayectoria generada usando el robot biciclo, mostraremos a continuación los resultados obtenidos tras seguir la trayectoria generada en el mapa 2.

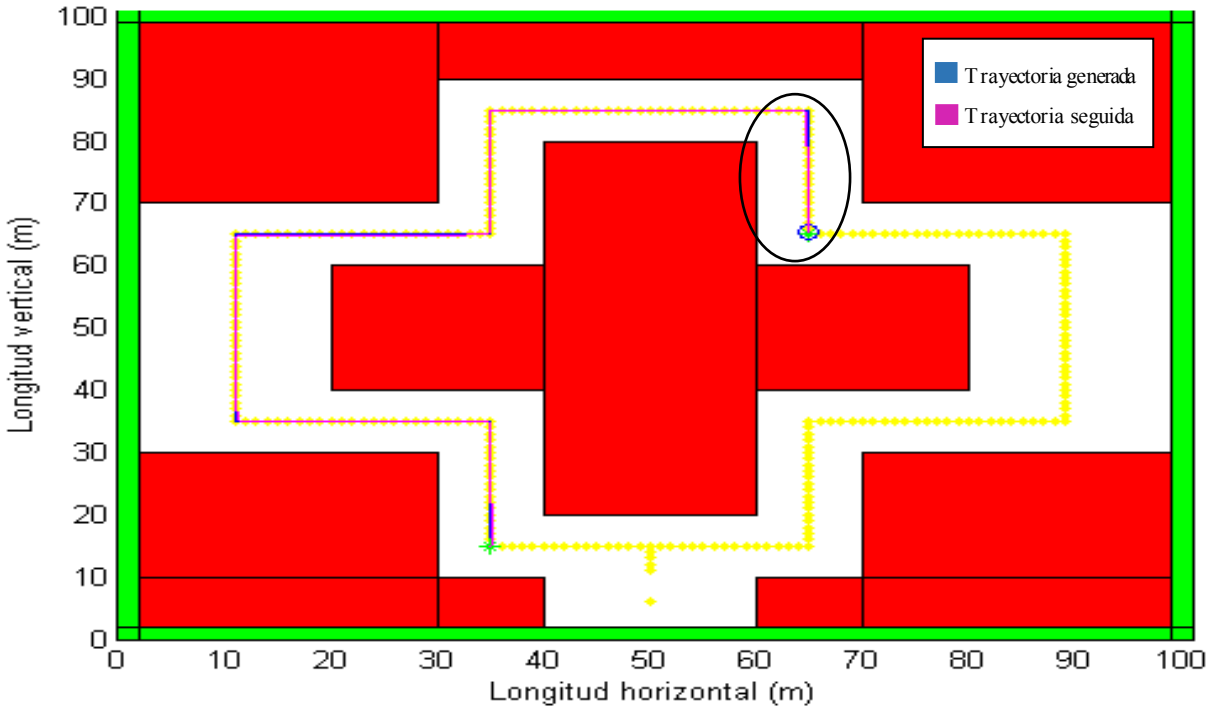


Figura 5-46. Seguimiento de trayectoria con robot biciclo en mapa 2 (D.Voronoi).

En esta imagen vemos como efectivamente la trayectoria es válida, tal y como preveíamos. A pesar de ello, al igual que en el mapa anterior ampliaremos la zona marcada para ver con cuanta distancia el robot recorre la trayectoria.

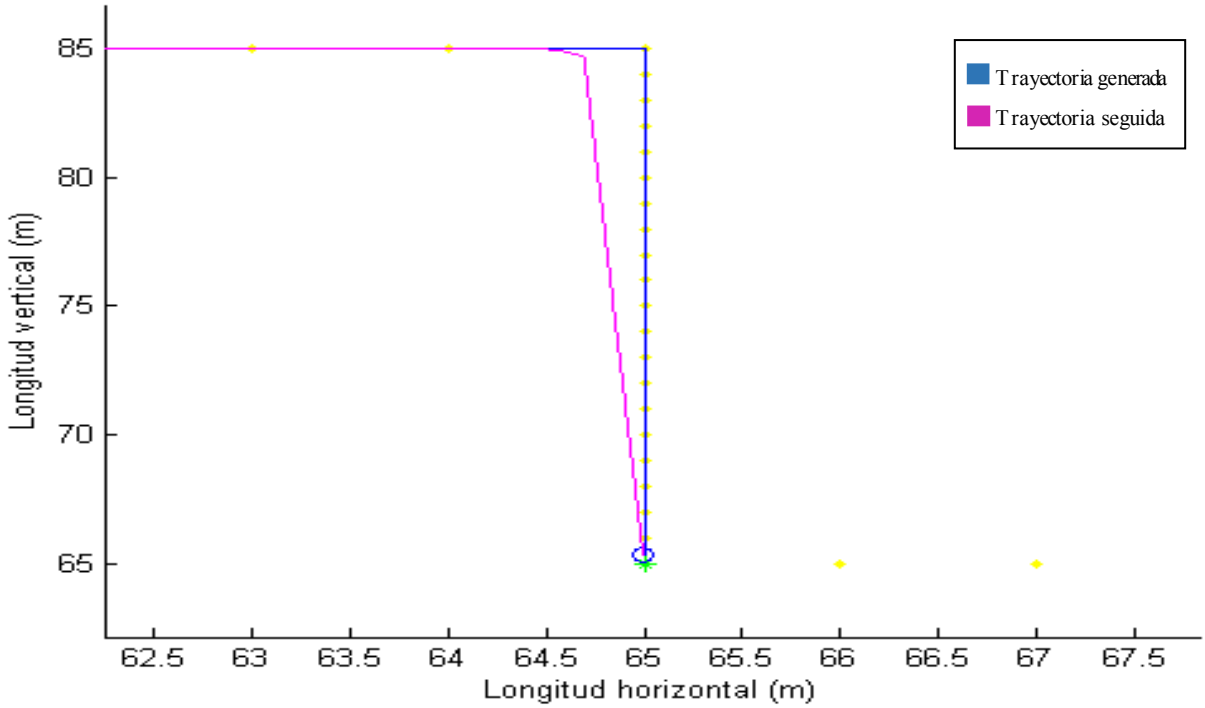


Figura 5-47. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 2 (D.Voronoi).

Al ampliar la imagen vemos que la distancia máxima que separa la trayectoria deseada y la trayectoria realizada por el robot es de 0.5 metros, más que suficiente para aceptar las constantes empleadas como válidas para el robot biciclo en cualquier tipo de trayectoria.

- Diferencial

Al igual que en los apartados anteriores mostraré lo primero las constantes empleadas para el seguimiento de las trayectorias generadas por los diagramas de Voronoi usando el robot diferencial.

Tabla 5–10. Constantes del robot diferencial para diagramas de Voronoi

d	Kv	Kh	Ki
0	0.001	10	300

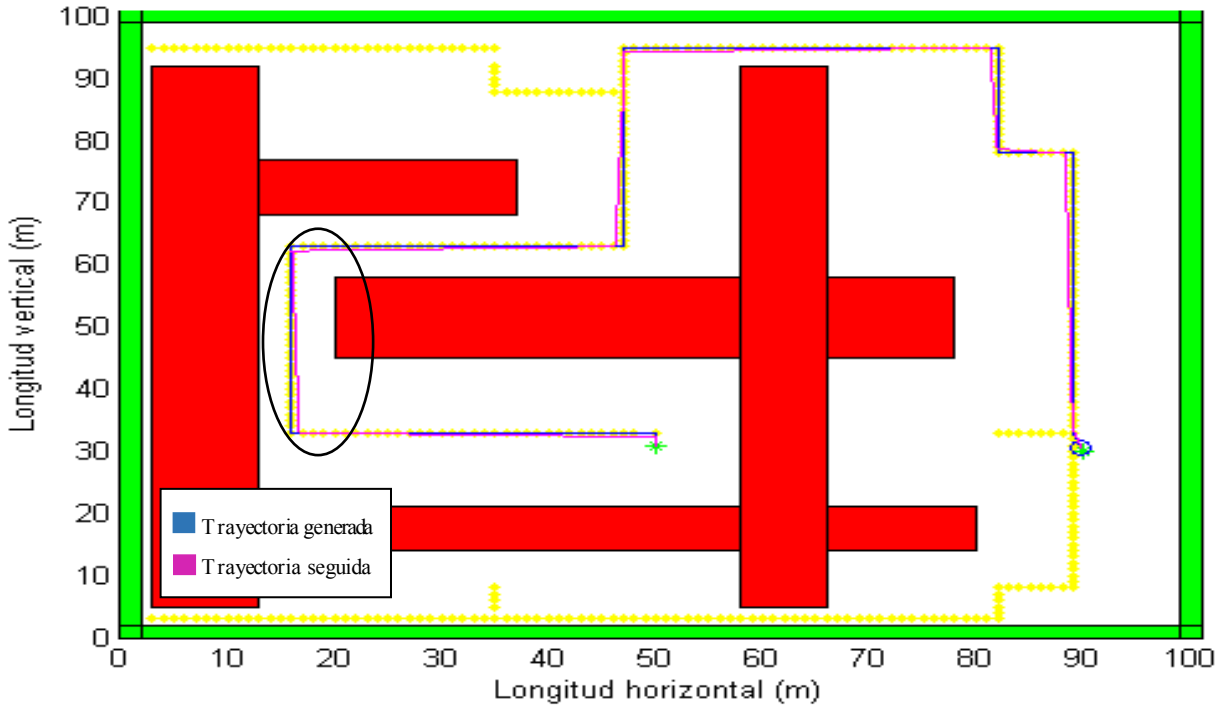


Figura 5-48. Seguimiento de trayectoria con robot diferencial en mapa 1 (D.Voronoi).

En este caso al igual que en el caso anterior podemos asegurar que la trayectoria seguida es aceptable, ya que no se produce ningún tipo de colisión. A pesar de ello en esta ocasión observamos como la trayectoria seguida y la deseada distan algo más que anteriormente, y es por ello que ampliaremos la zona seleccionada para ver de qué distancias estamos hablando. He de añadir que este robot se desplaza considerablemente lento si lo comparamos con el robot biciclo en las trayectorias generadas por los diagramas de Voronoi.

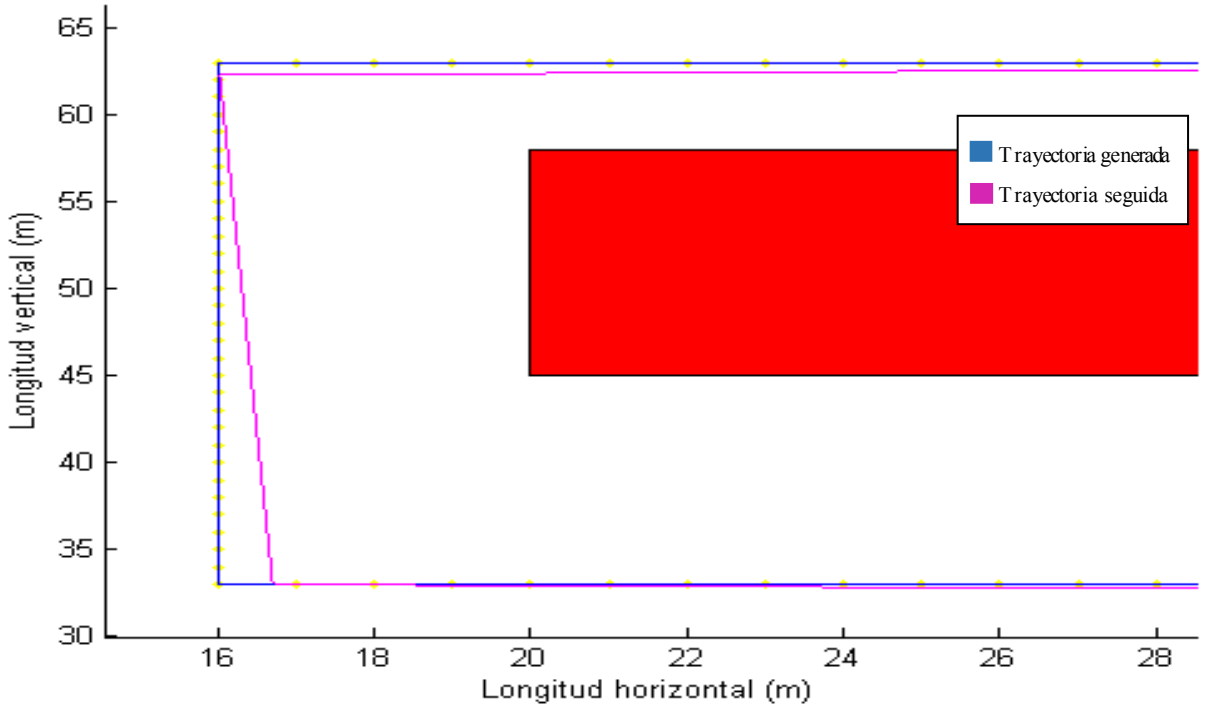


Figura 5-49. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 1 (D.Voronoi).

Tras ampliar la zona deseada, vemos como la distancia en el punto más crítico es de aproximadamente 0.8 metros, lo cual en principio no nos casaría ningún inconveniente como ya preveíamos.

A continuación, y a pesar de que con los resultados anteriores podríamos dar por válidas las constantes empleadas, hemos realizado el mismo experimento en el mapa 2 para así poder asegurarnos de que efectivamente son correctas.

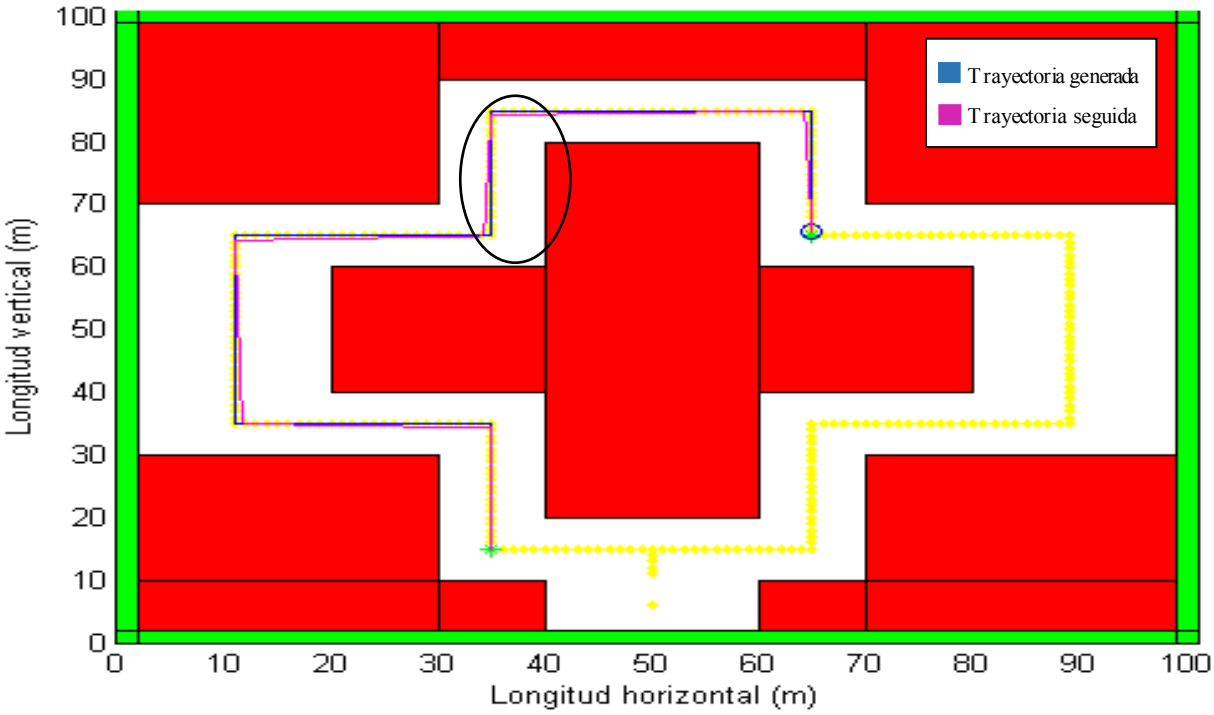


Figura 5-50. Seguimiento de trayectoria con robot diferencial en mapa 2 (D.Voronoi).

Como ya habíamos previsto, vemos como la nueva trayectoria seguida por el robot diferencial es correcta y no presenta ningún tipo de problema. Aun así ampliaremos la zona marcada y miraremos si las distancias que separan ambas trayectorias son lo suficientemente pequeñas como para aceptar dichas variables como buenas.

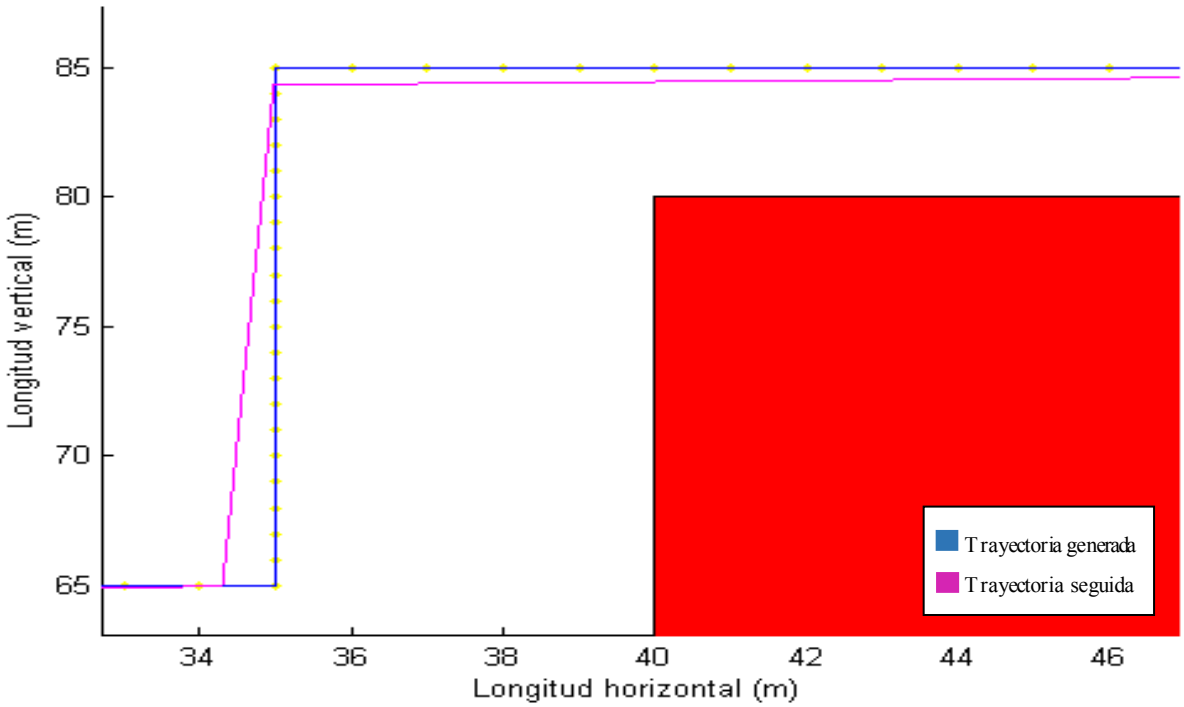


Figura 5-51. Ampliación de seguimiento de trayectoria con robot sincrono en mapa 2 (D.Voronoi).

En esta ocasión las diferencias entre ambas trayectorias son menores de 0.5 metros, por lo cual consideramos que las variables valdrían para las trayectorias generadas por los diagramas de Voronoi y seguidas por el robot diferencial.

- Síncrono

Mostraremos a continuación las constantes usadas para el robot síncrono en el seguimiento de las trayectorias generadas por los diagramas de Voronoi. Tras ello, veremos los resultados de los experimentos realizados con dichas constantes, con los que comprobaremos como estas son válidas.

Tabla 5–11. Constantes del robot síncrono para diagramas de Voronoi

d	Kv	Kh	Ki
0	0.005	10	300

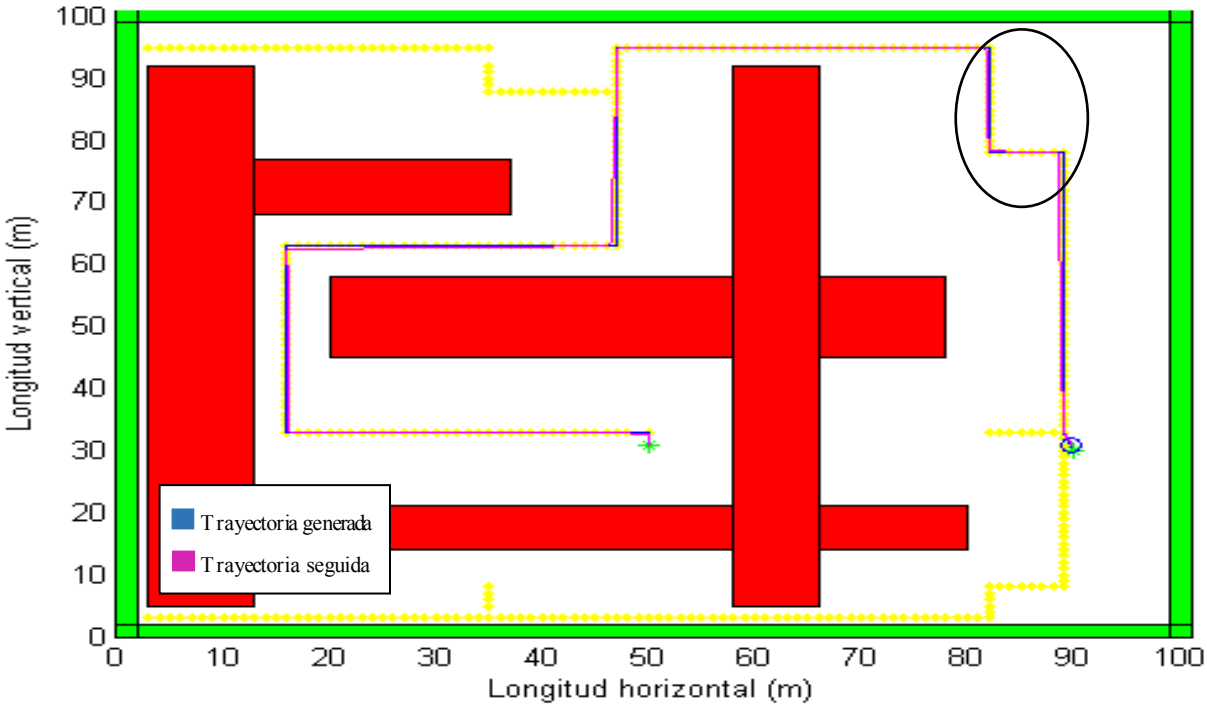


Figura 5-52. Seguimiento de trayectoria con robot síncrono en mapa 1 (D.Voronoi).

Observamos en la imagen como claramente la trayectoria seguida por el robot, a pesar de no coincidir exactamente con la calculada por los diagramas de Voronoi, es una trayectoria correcta, ya que no presenta ningún tipo de duda acerca de si colisiona con los obstáculos o no. A pesar de ello vamos a ampliar la zona marcada para ver de qué distancias estamos hablando. En cuanto a la velocidad con la que este robot recorre la trayectoria, es similar a la del robot diferencial para este mismo algoritmo.

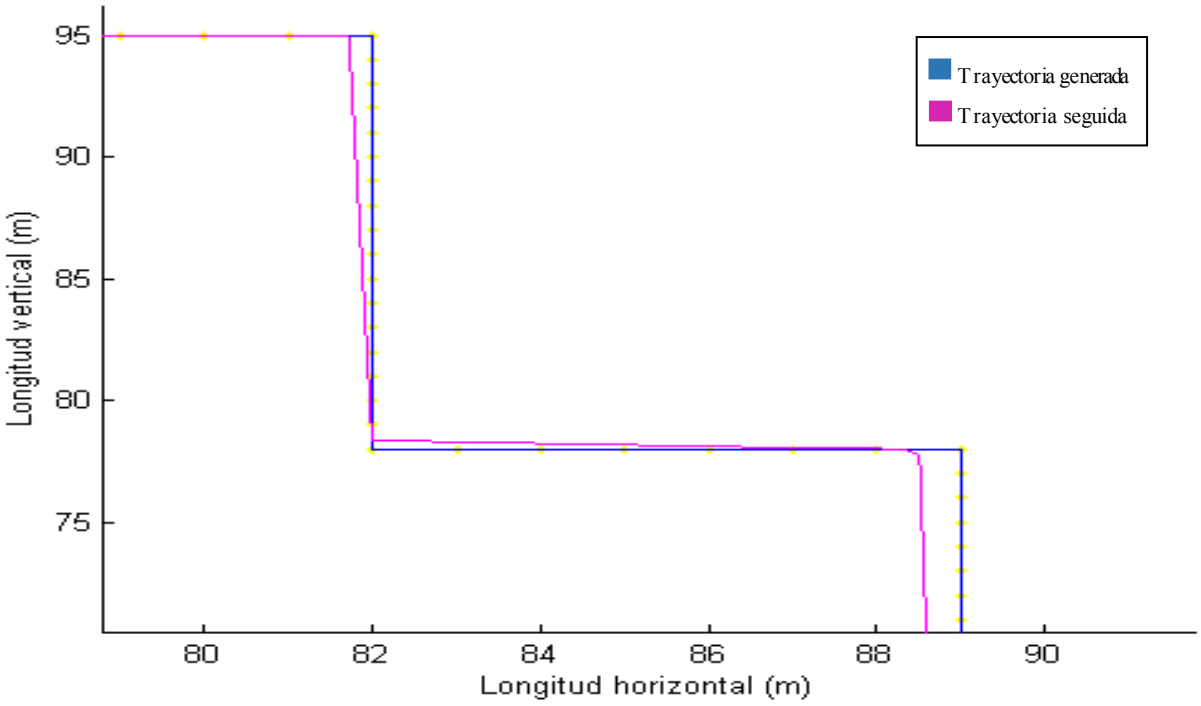


Figura 5-53. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 1 (D.Voronoi).

Tras realizar la ampliación vemos como la separación entre ambas trayectorias son aproximadamente 0.5 metros, lo cual para este tipo de trayectorias es bastante aceptable.

Aun sabiendo gracias a los resultados obtenidos con el experimento anterior que nuestras constantes son válidas, hemos realizado un segundo experimento para confirmarlo.

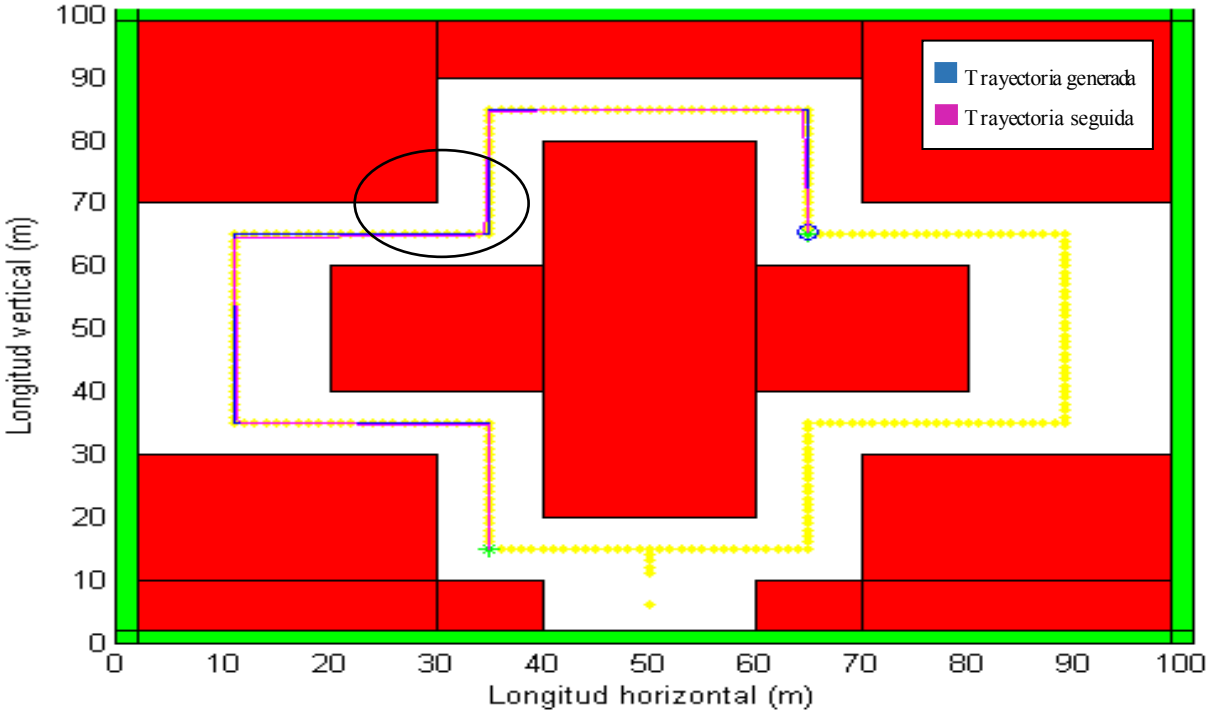


Figura 5-54. Seguimiento de trayectoria con robot síncrono en mapa 2 (D.Voronoi).

De nuevo vemos como los resultados son bastante buenos y que a pesar de no tener ningún tipo de duda acerca que la trayectoria es válida, ampliaremos a continuación la zona seleccionada, para ver las diferencias entre ambas. Aunque sin necesidad de ampliarlas podemos afirmar que las constantes usadas son las adecuadas.

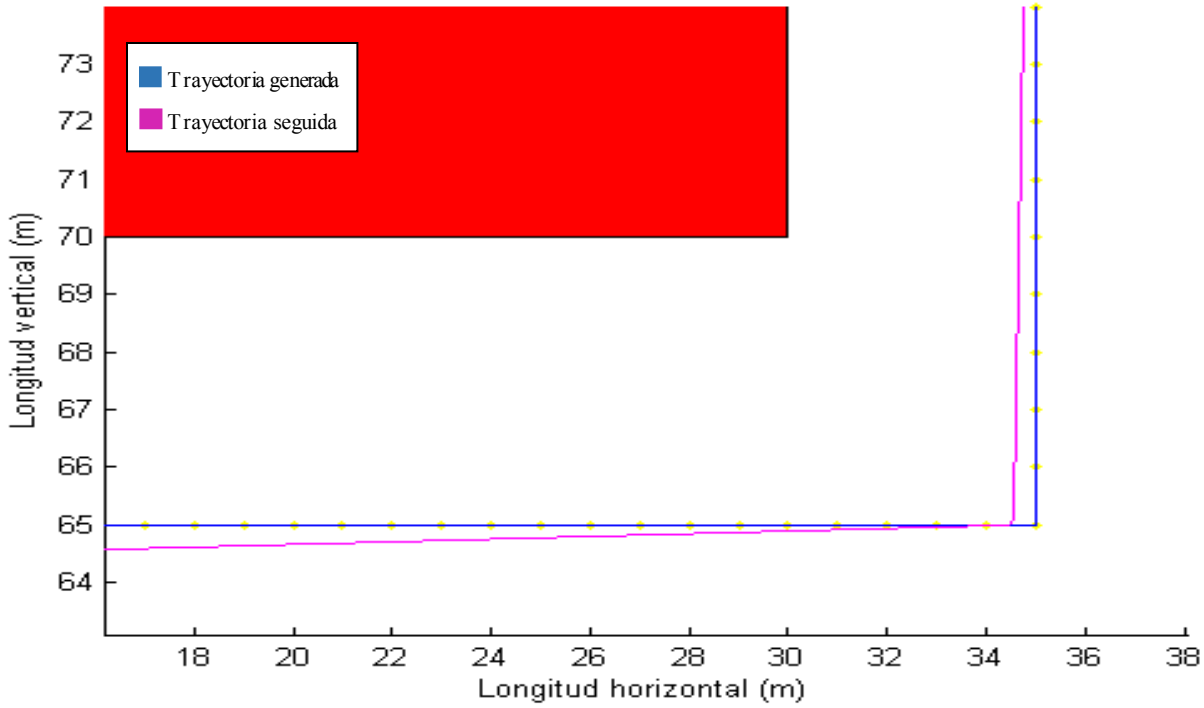


Figura 5-55. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 2 (D.Voronoi).

- Triciclo

  Las constantes que en esta ocasión hemos empleado para el robot triciclo siguiendo a una trayectoria generada por los diagramas de Voronoi son las siguientes:

Tabla 5–12. Constantes del robot triciclo para diagramas de Voronoi

d	Kv	Kh	Ki
0	0.05	10	30

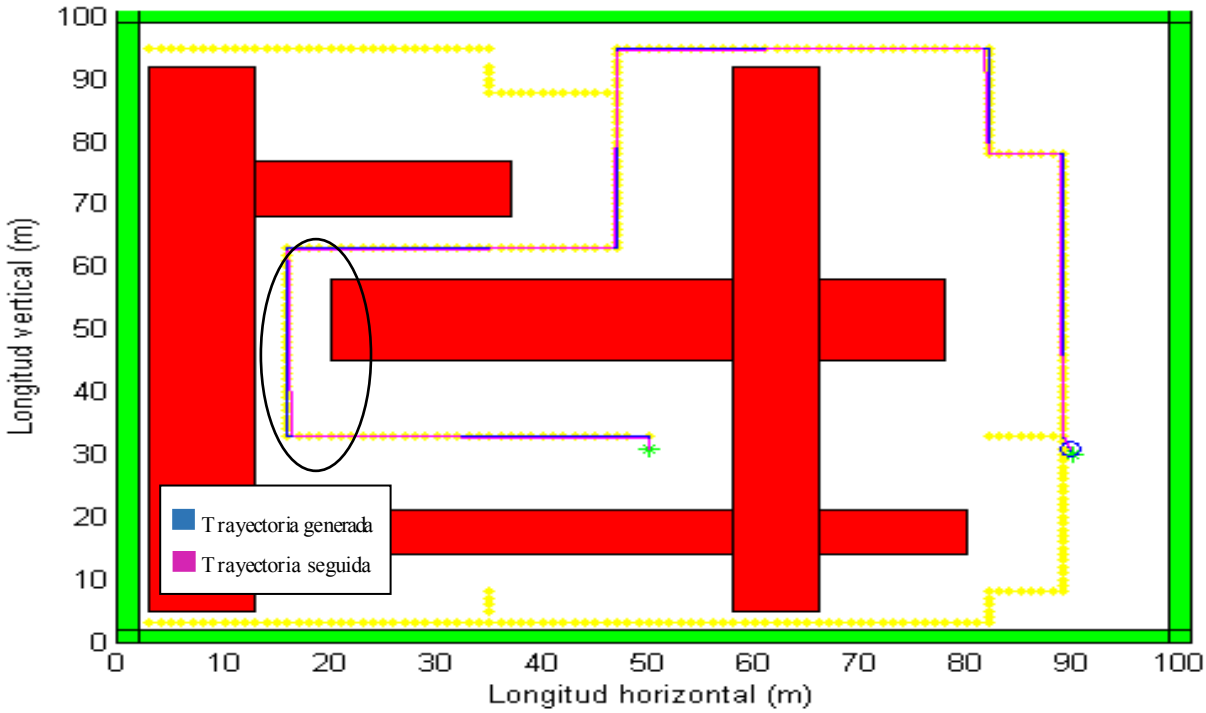


Figura 5-56. Seguimiento de trayectoria con robot triciclo en mapa 1 (D.Voronoi).

En esta ocasión y al igual que con todos los robots anteriores, este método proporciona unas trayectorias que permiten saber a simple vista que el robot no colisiona con ningún obstáculo presente. Aun así y por conocer la distancia que separa ambas, ampliamos al igual que anteriormente la zona redondeada. Para este caso la velocidad con la que el robot sigue el camino deseado es considerablemente superior que cuando el robot síncrono o diferencial seguían la misma trayectoria, y bastante similar a cuando la seguía el biciclo.

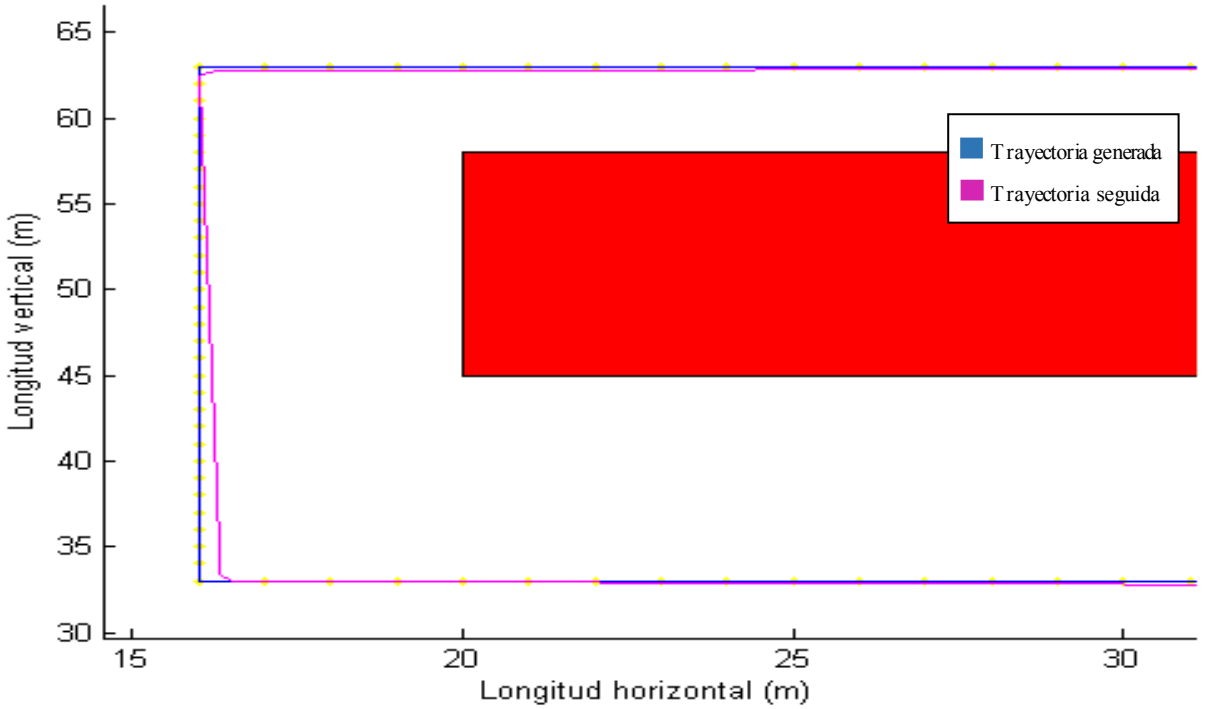


Figura 5-57. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 1 (D.Voronoi).

Con esta ampliación vemos como las diferencias entre ambas trayectorias no son excesivamente grandes, aproximadamente 0.8 metros, pudiendo ser asumibles medidas bastante mayores.

A continuación mostraremos los resultados obtenidos tras la realización de un segundo experimento, a pesar de que observando los resultados obtenidos en el experimento anterior pudiésemos prácticamente asegurar que las constantes seleccionadas son las correctas.

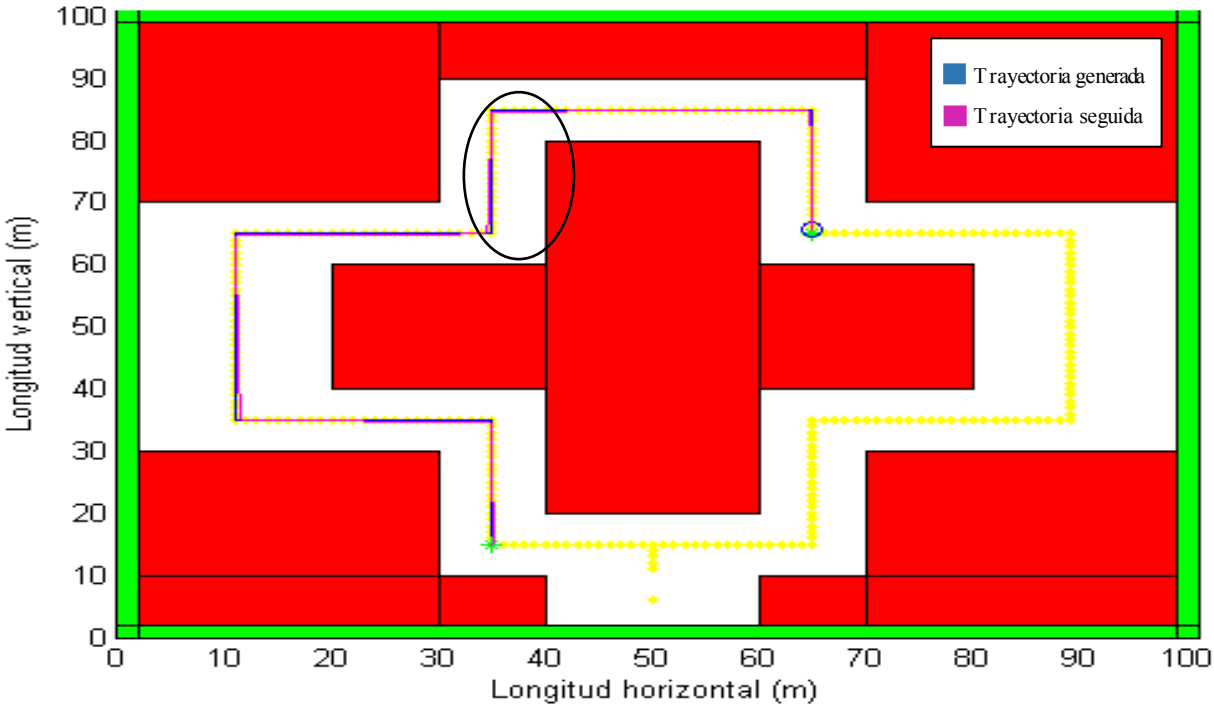


Figura 5-58. Seguimiento de trayectoria con robot triciclo en mapa 2 (D.Voronoi).

Como ya preveíamos los resultados de este segundo experimento son los esperados, siendo las diferencias entre ambas trayectorias mínimas y por consiguiente válidas.

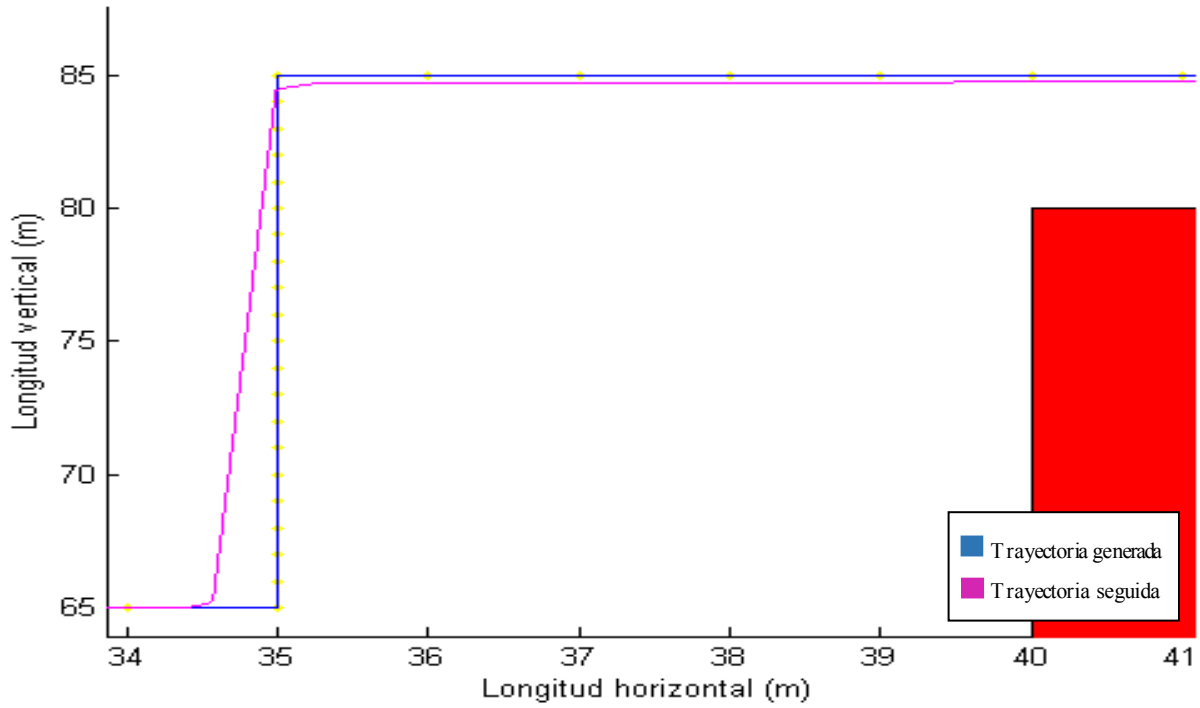


Figura 5-59. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 2 (D.Voronoi).

5.2.4 Descomposición en celdas verticales

- Biciclo

  En primer lugar mostraremos las constantes elegidas para el robot biciclo de forma que este siga correctamente la trayectoria generada por el algoritmo de descomposición en celdas verticales. Tras ello podremos ver diferentes experimentos en los que podremos afirmar que dichas constantes elegidas son las correctas.

Tabla 5–13. Constantes del robot biciclo para desc. en celdas verticales

d	Kv	Kh	Ki
0	0.05	10	30

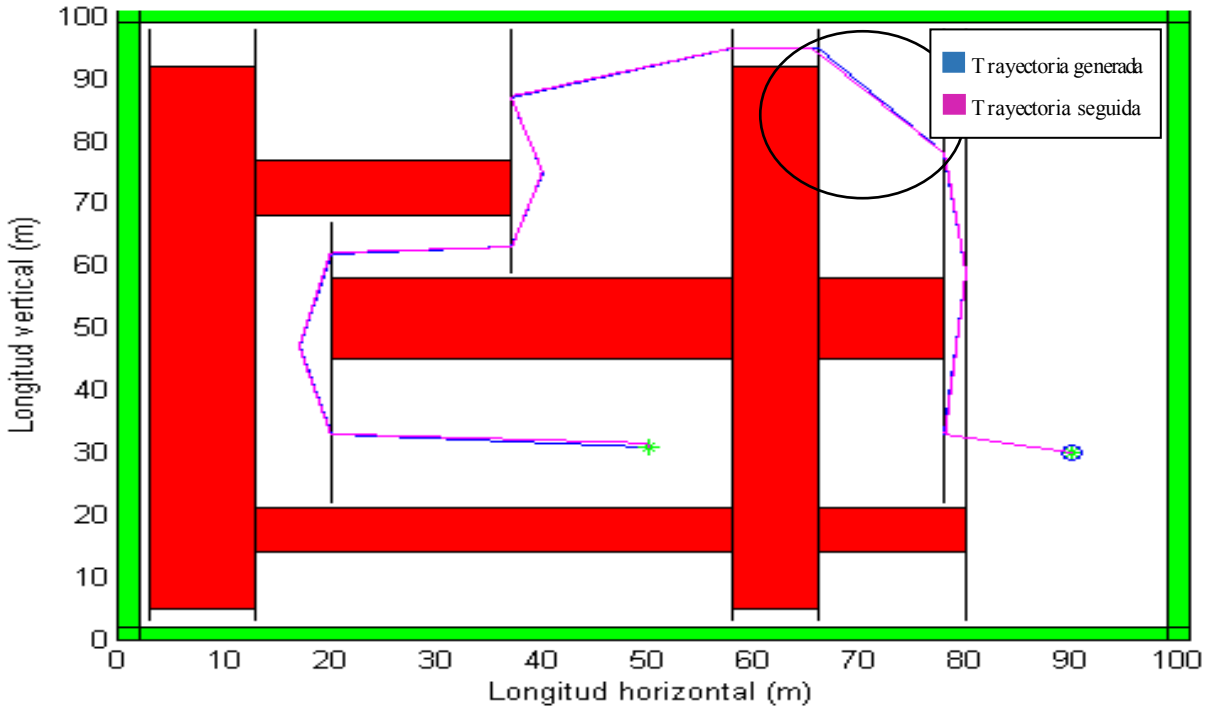


Figura 5-60. Seguimiento de trayectoria con robot biciclo en mapa 1 (Desc.Celdas Verticales).

Tras observar el resultado del primer experimento realizado, vemos como el robot sigue de forma bastante fiel a la trayectoria que hemos generado mediante el algoritmo de descomposición en celdas verticales. Podríamos incluso afirmar a simple vista que la trayectoria seguida es una trayectoria libre de colisiones y por tanto adecuada. Aun así, ampliaremos la zona marcada para ver cuál es la diferencia entre ambas. Además podemos ver que en cuanto a velocidad para el mismo tipo de robot las velocidades de este método y de los diagramas de Voronoi son similares.

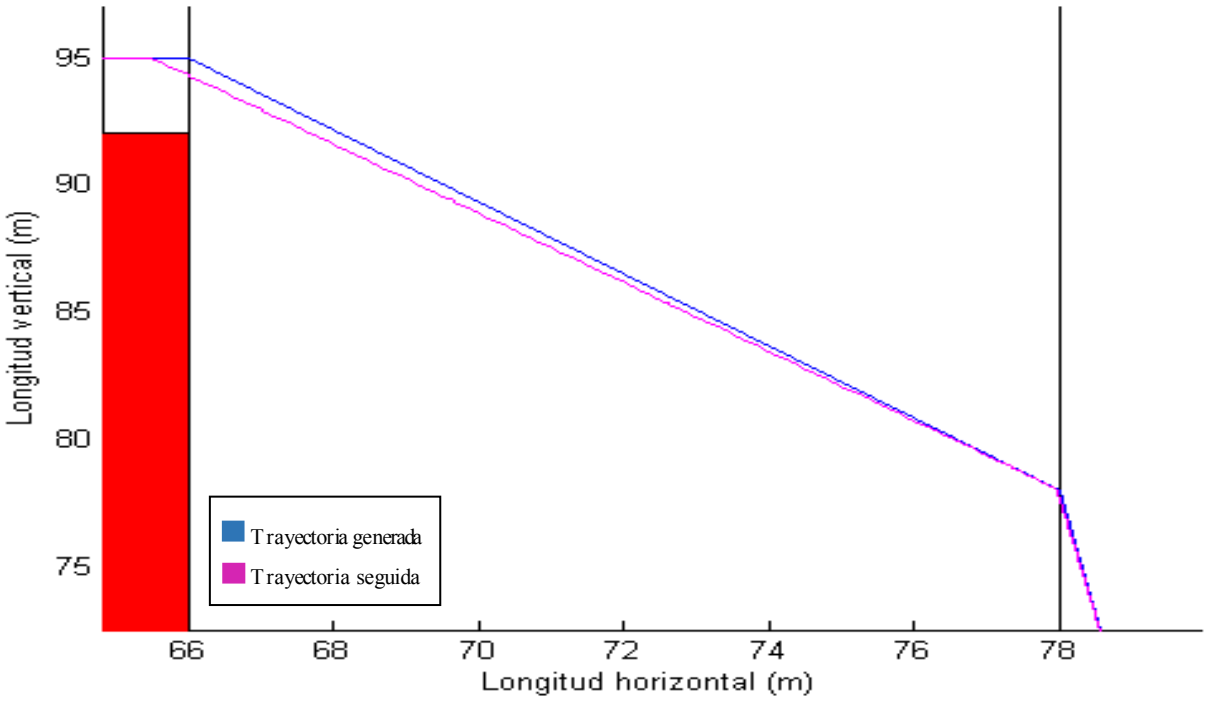


Figura 5-61. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 1 (Desc.Celdas Verticales).

En la ampliación del primer experimento vemos como en el punto donde más se distancian ambas trayectorias, esta diferencia es de apenas 0.5 metros. Por tanto es una trayectoria correcta.

A continuación y para asegurarnos de que efectivamente las constantes asignadas son las adecuadas, realizaremos un segundo experimento.

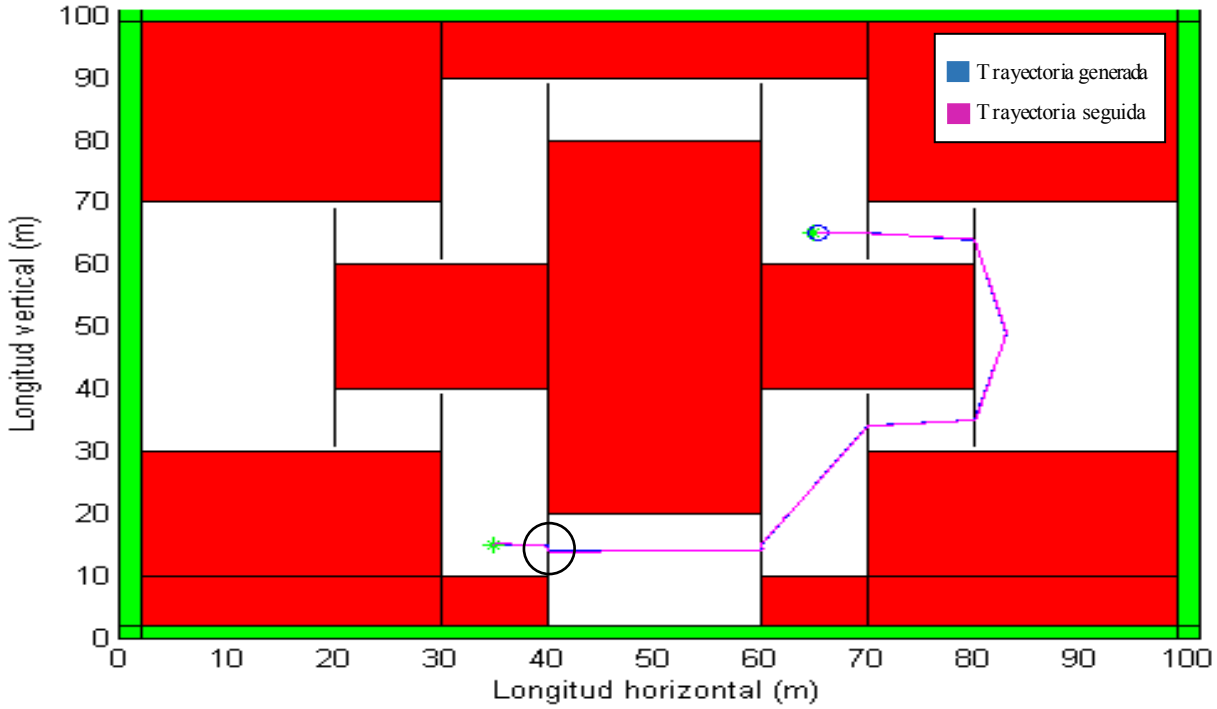


Figura 5-62. Seguimiento de trayectoria con robot biciclo en mapa 2 (Desc.Celdas Verticales).

Como ya habíamos previsto tras el experimento anterior, la trayectoria realizada por el robot es bastante similar a la generada por el algoritmo de descomposición en celdas verticales. De hecho podemos afirmar que las constantes escogidas son las adecuadas sin necesidad de ampliar la imagen, a pesar de ello realizaremos un zoom para ver de qué diferencias estamos hablando.

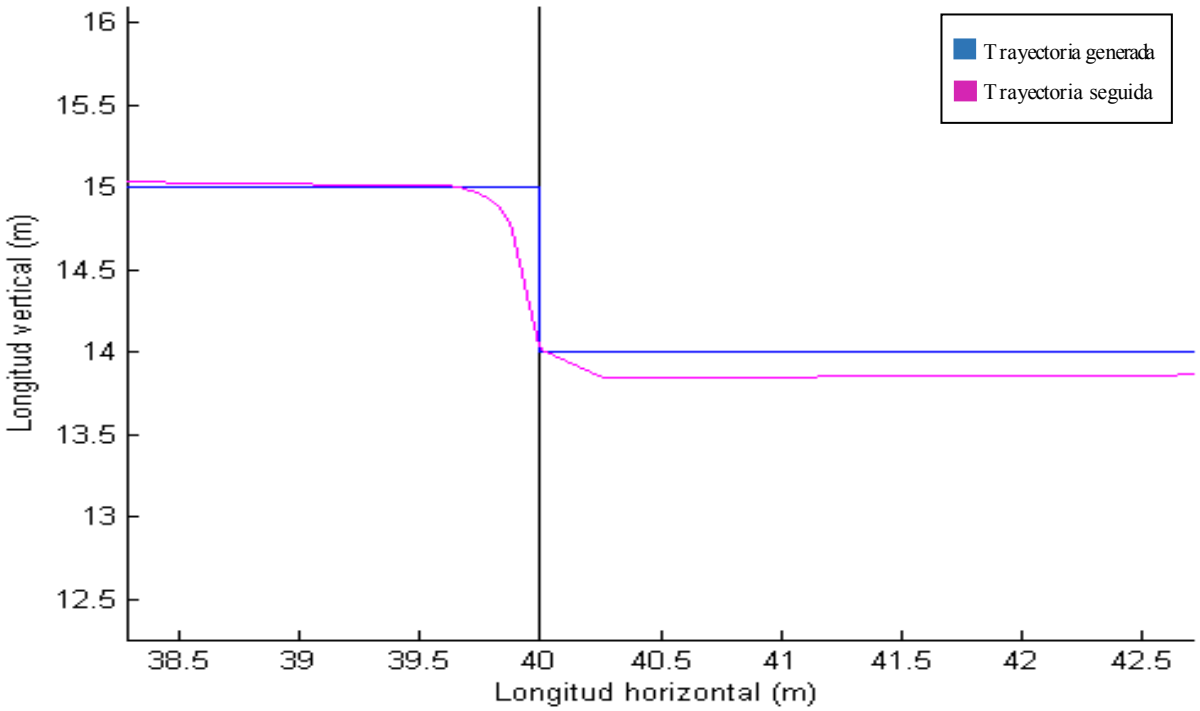


Figura 5-63. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 2 (Desc.Celdas Verticales).

- Diferencial

  Al igual que para el resto de métodos y robots, en la siguiente tabla encontramos las constantes usadas para el robot diferencial siguiendo la trayectoria generada por el algoritmo de descomposición en celdas verticales. Más adelante se confirmará que dichas constantes son las apropiadas.

Tabla 5–14. Constantes del robot diferencial para desc. en celdas verticales

d	Kv	Kh	Ki
0	0.01	10	300

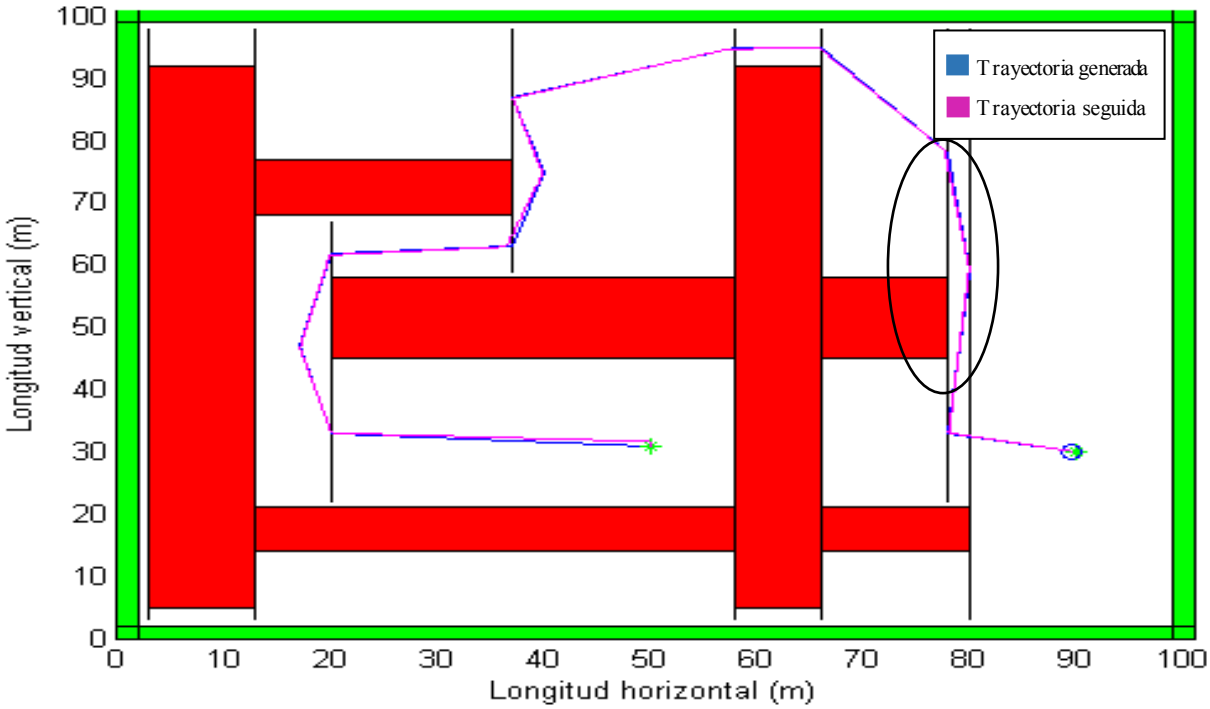


Figura 5-64. Seguimiento de trayectoria con robot diferencial en mapa 1 (Desc.Celdas Verticales).

Al igual que en el robot biciclo siguiendo la misma trayectoria, vemos como claramente la trayectoria seguida es la apropiada, ya que es una trayectoria libre de colisiones. No obstante ampliaremos la zona que parece más conflictiva para asegurarnos que no se produzca ningún choque. Una cosa que podemos apreciar es que la velocidad con la que se desplaza el robot diferencial siguiendo esta trayectoria específica es ligeramente inferior que cuando la recorre el robot biciclo.

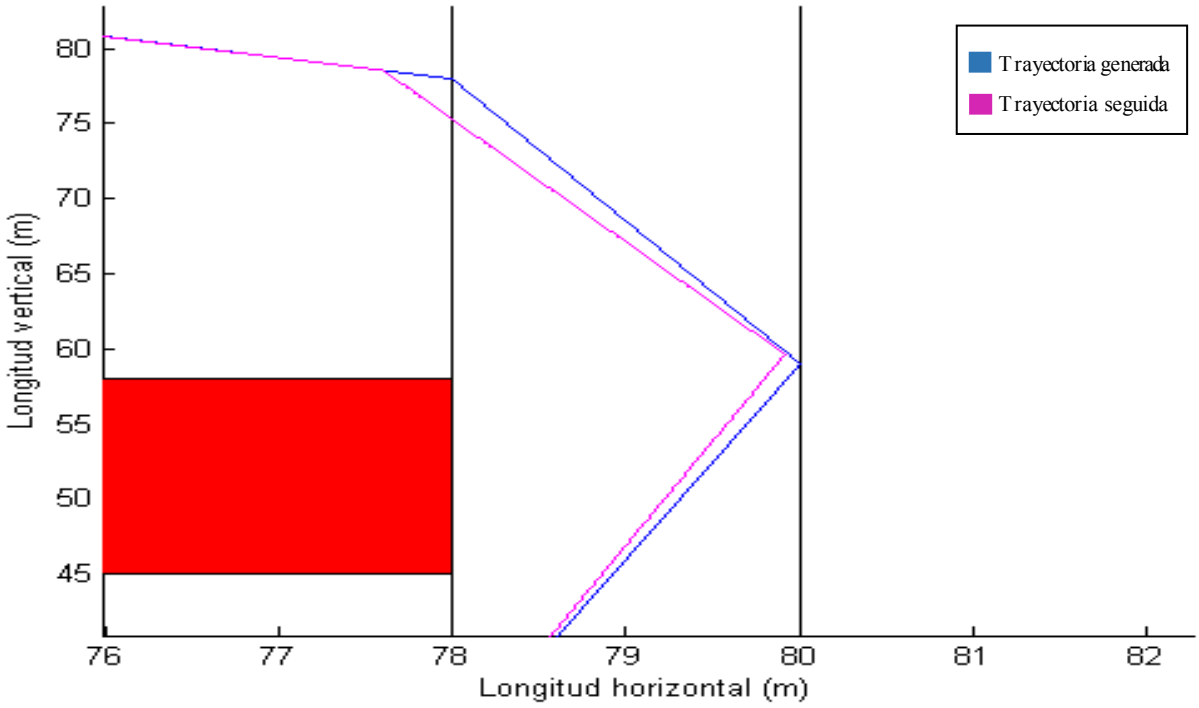


Figura 5-65. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 2 (Desc.Celdas Verticales).

Tras observar la zona ampliada vemos que lo que se diferencian ambas trayectorias es lo suficientemente pequeño como para no invalidar las constantes. Además en la zona más conflictiva podemos apreciar como el robot se distancia del objeto más de 0.5 metros, distancia suficiente como para que el robot pase sin ningún problema.

Para dar por válidas las constantes empleadas realizaremos un segundo experimento.

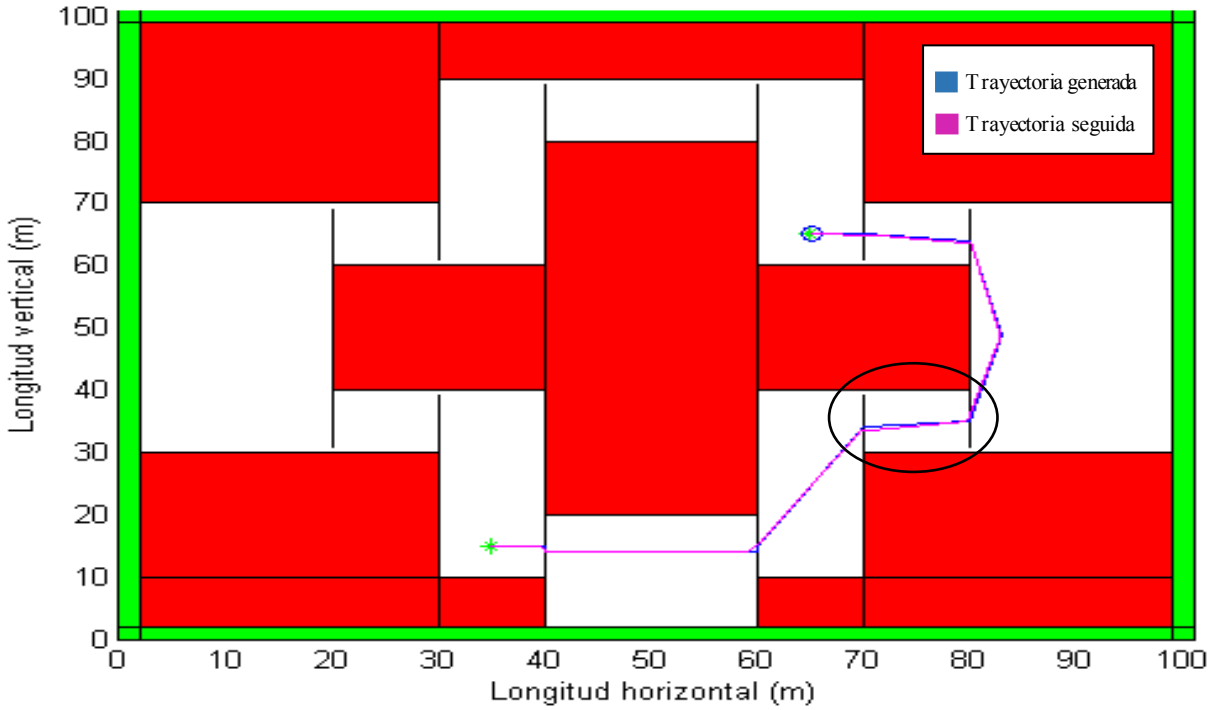


Figura 5-66. Seguimiento de trayectoria con robot diferencial en mapa 2 (Desc.Celdas Verticales).

En este segundo recorrido podemos comprobar como efectivamente las constantes son las adecuadas, ya que la trayectoria que hemos generado no presenta ningún inconveniente. Vamos a ampliar la zona seleccionada para poder apreciar con más claridad cómo estamos en lo cierto.

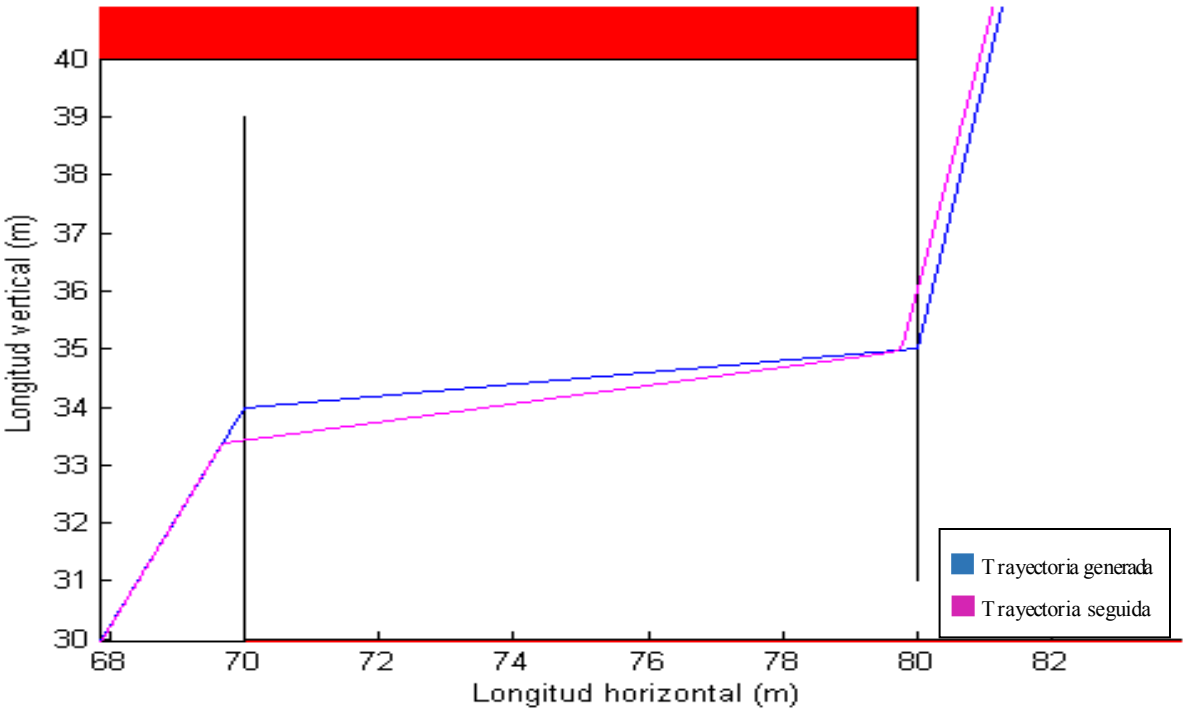


Figura 5-67. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 2 (Desc.Celdas Verticales).

- Síncrono

  En la siguiente tabla mostraremos las constantes empleadas para que el robot síncrono siga la trayectoria generada por el algoritmo de descomposición en celdas verticales adecuadamente.

Tabla 5–15. Constantes del robot síncrono para desc. en celdas verticales

d	Kv	Kh	Ki
0	0.01	10	300

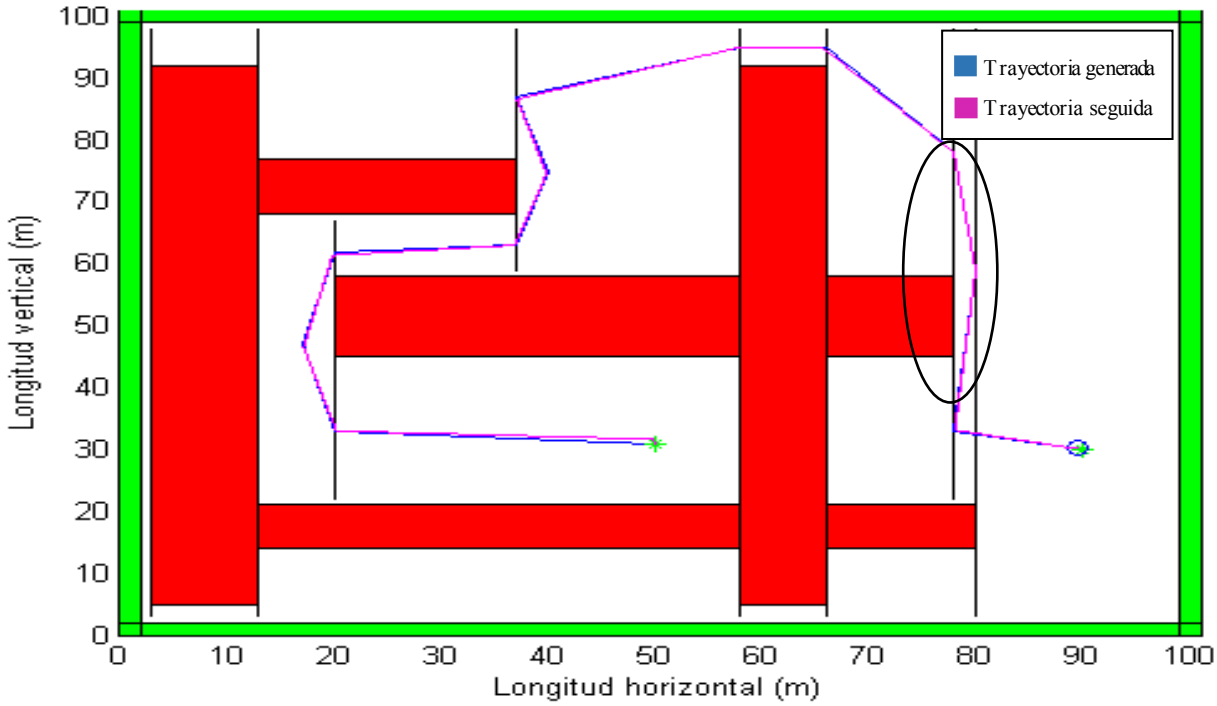


Figura 5-68. Seguimiento de trayectoria con robot síncrono en mapa 1 (Desc.Celdas Verticales).

Como vemos en la imagen anterior la trayectoria que realiza el robot síncrono es una trayectoria que aparentemente parece válida, para confirmarlo ampliaremos y estudiaremos la zona seleccionada, ya que es la zona donde encontraríamos algún problema de haberlo. Con respecto a la velocidad con la que el robot recorre la trayectoria, esta es muy similar a con la que el robot diferencial la recorre y por consiguiente algo menor que cuando la recorre el robot biciclo.

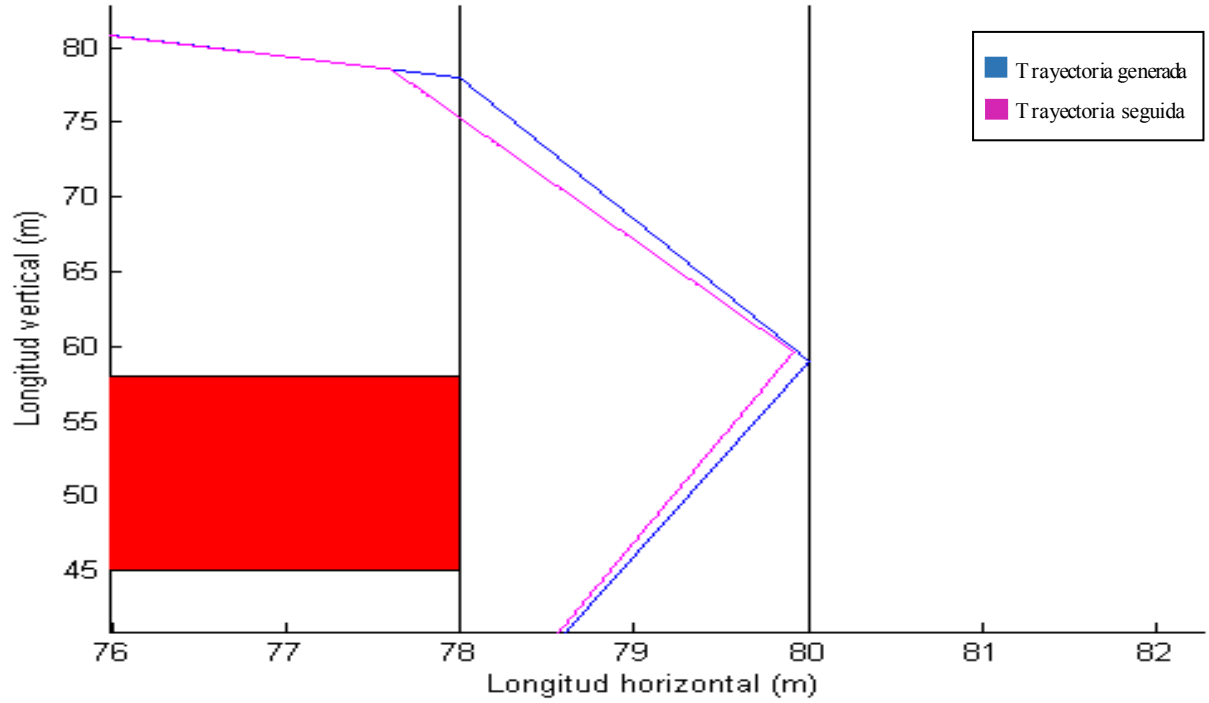


Figura 5-69. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 1 (Desc.Celdas Verticales).

Al ampliar la zona vemos como la trayectoria que el robot sigue es válida, ya que en el punto que más cerca se encuentra del obstáculo la distancia entre ellos es de más de 0.5 metros. Además observamos como el comportamiento del robot ante la trayectoria es bastante similar al del robot diferencial, por tanto esta es válida también.

Para asegurarnos de que con dichas constantes las trayectorias se seguirán de forma correcta, se realizará otra prueba con la que podremos ver como las constantes elegidas son las adecuadas.

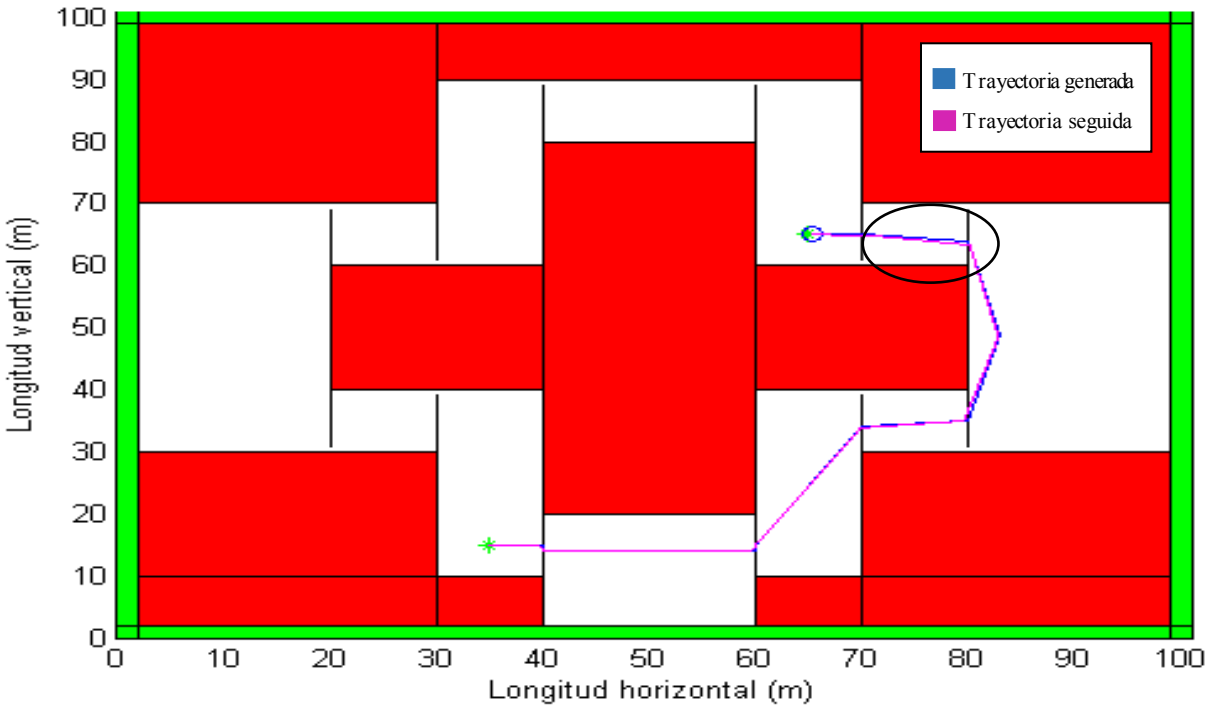


Figura 5-70. Seguimiento de trayectoria con robot síncrono en mapa 2 (Desc.Celdas Verticales).

Como podemos comprobar, este segundo experimento nos corrobora que las constantes son las adecuadas ya que la trayectoria del robot síncrono es totalmente correcta. A pesar de ello ampliaremos la zona redondeada para comprobar las diferencias existentes entre ambas trayectorias.

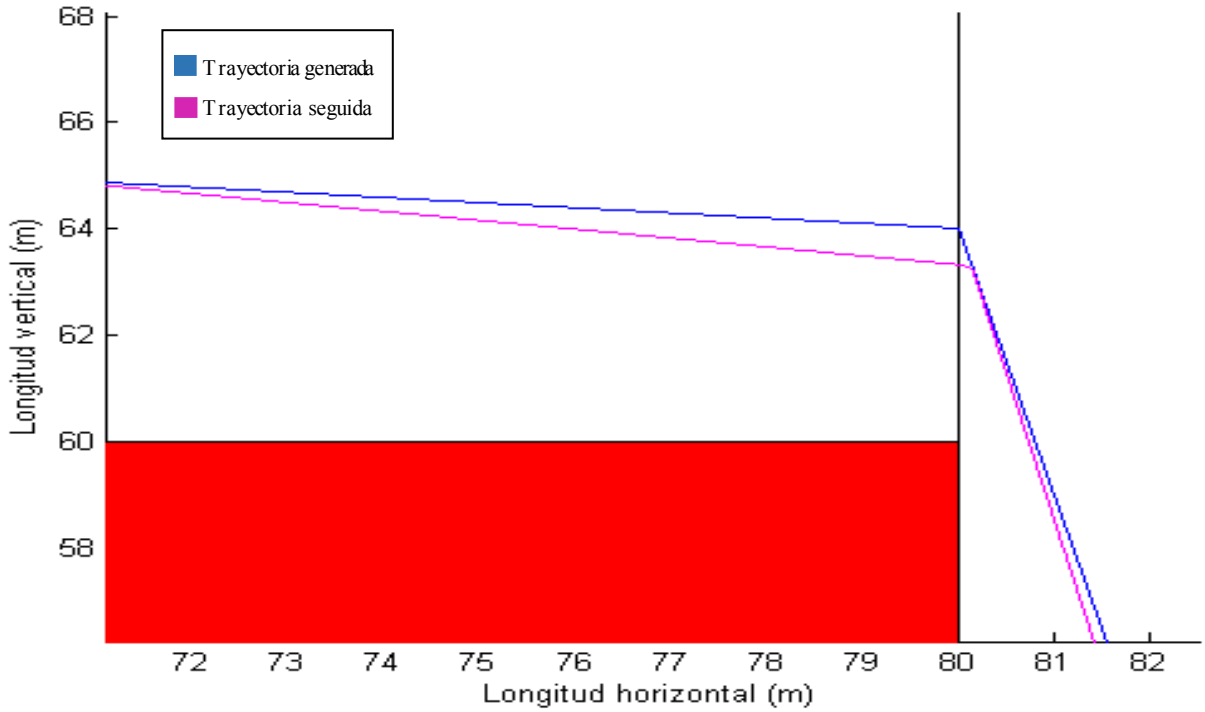


Figura 5-71. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 2 (Desc.Celdas Verticales).

- Triciclo

  Las constantes empleadas para el robot triciclo, cuando queremos que este siga la trayectoria generada por el algoritmo de descomposición en celdas verticales son las siguientes:

Tabla 5–16. Constantes del robot triciclo para desc. en celdas verticales

d	Kv	Kh	Ki
0	0.05	10	30

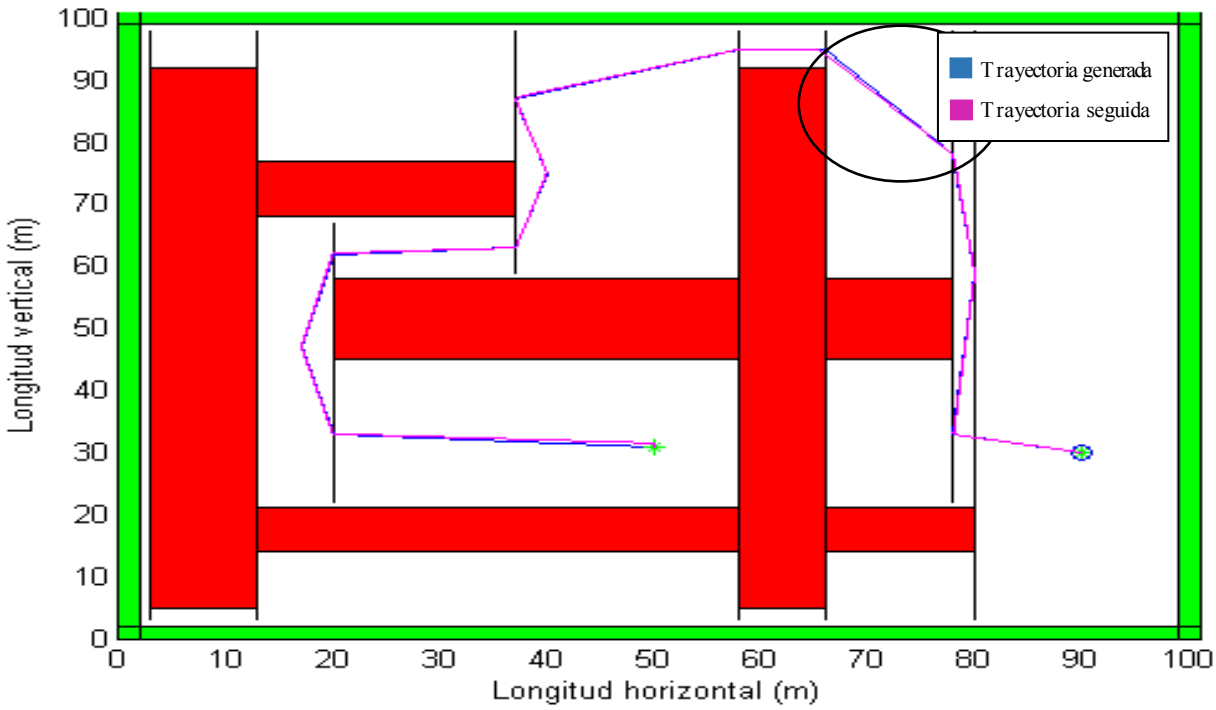


Figura 5-72. Seguimiento de trayectoria con robot triciclo en mapa 1 (Desc.Celdas Verticales).

Como todos los demás robots usados para seguir la trayectoria generada por el algoritmo de descomposición en celdas verticales, la trayectoria de este robot es válida, al menos aparentemente. Para confirmarlo haremos zoom sobre la zona marcada. Por otro lado la velocidad con la que este robot la recorre es algo superior que cuando la recorre el robot diferencial y el robot síncrono y similar a cuando la recorre el robot biciclo.

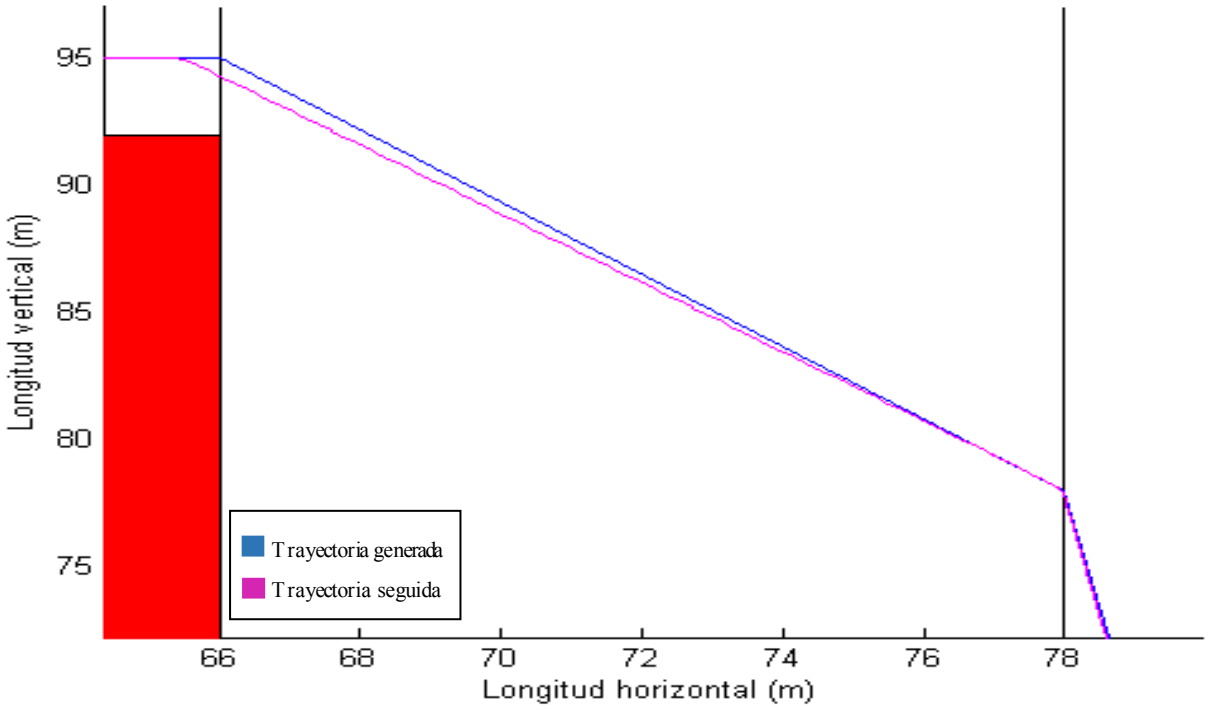


Figura 5-73. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 1 (Desc.Celdas Verticales).

Como comprobamos en la imagen ampliada y como preveíamos, los resultados obtenidos son los correctos,

y por tanto las constantes elegidas son las adecuadas.

Aun así, realzaremos otro experimento donde poder aseguramos de que efectivamente son válidas las constantes empleadas.

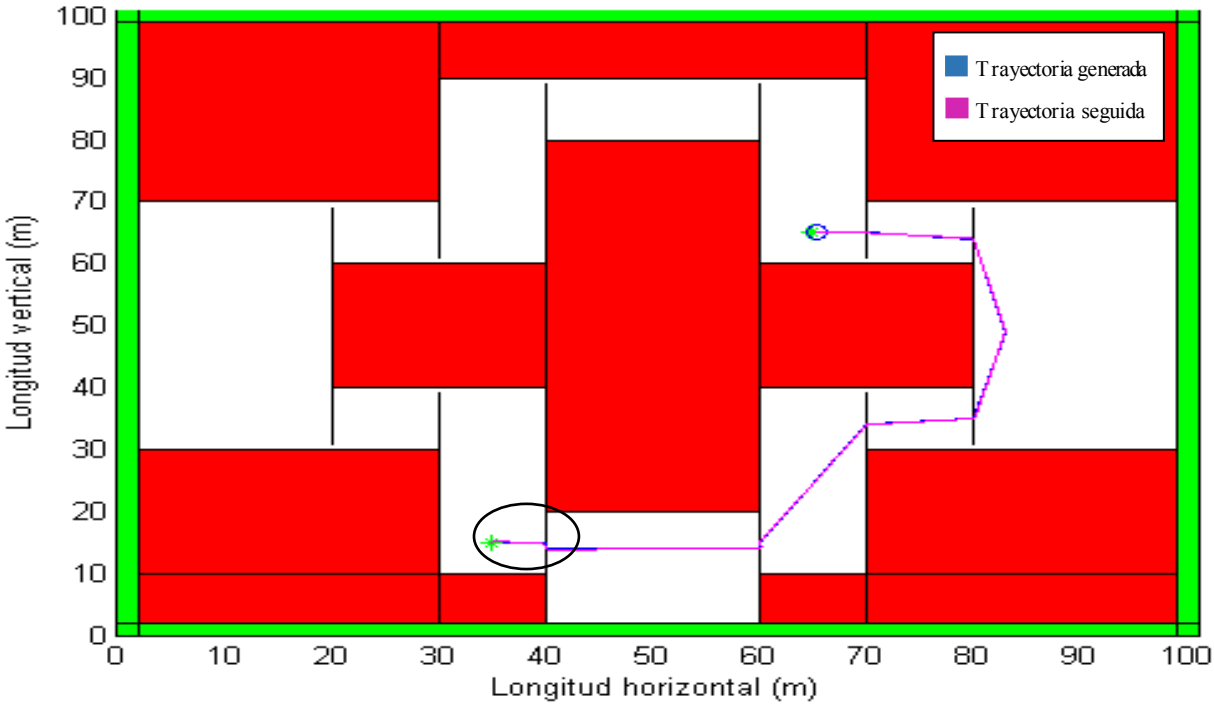


Figura 5-74. Seguimiento de trayectoria con robot triciclo en mapa 2 (Desc.Celdas Verticales).

Efectivamente con este segundo experimento confirmamos que las constantes son las adecuadas, ya que sin necesidad de ampliar ninguna zona vemos como la trayectoria que realiza el robot es la adecuada, a pesar de ello vamos a ampliar la zona marcada para ver las discrepancias entre la trayectoria seguida por el robot y la generada por el algoritmo.

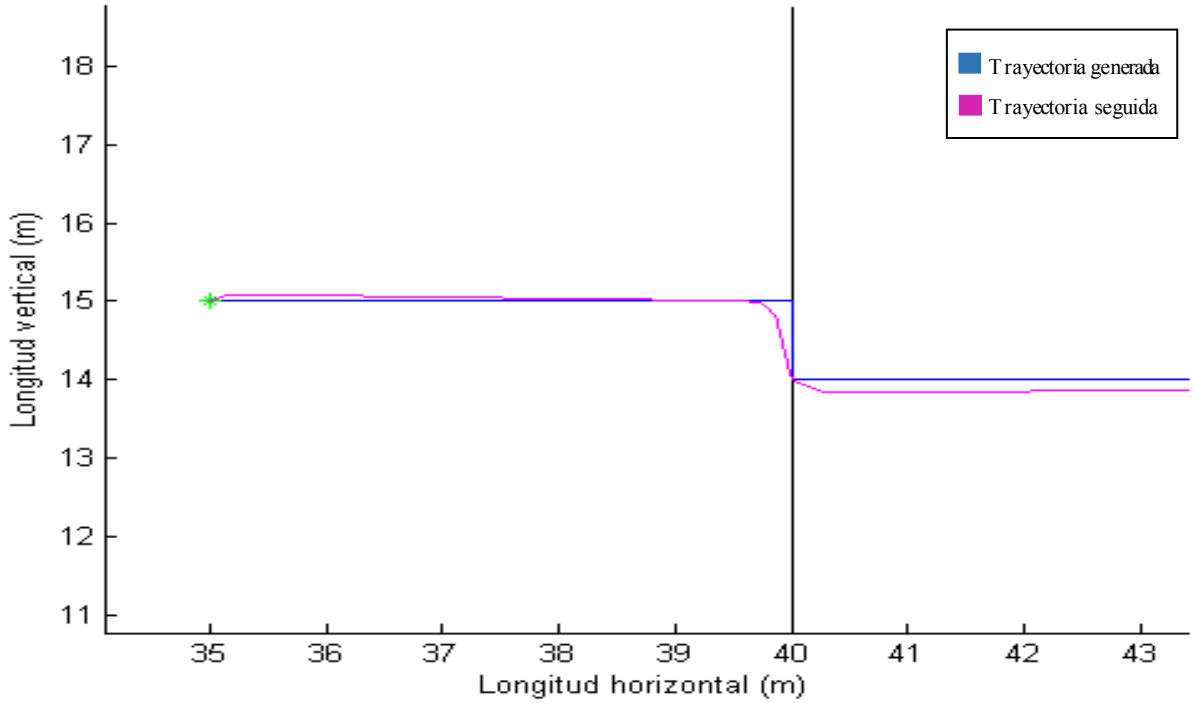


Figura 5-75. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 2 (Desc.Celdas Verticales).

5.2.5 Planificador de frente de ondas

- Biciclo

  En primer lugar pasaremos a mostrar la tabla en la que encontramos las constantes usadas para el robot biciclo en el seguimiento de una trayectoria generada por el algoritmo de frente de ondas.

Tabla 5–17. Constantes del robot biciclo para frente de ondas

d	Kv	Kh	Ki
0	0.05	10	10

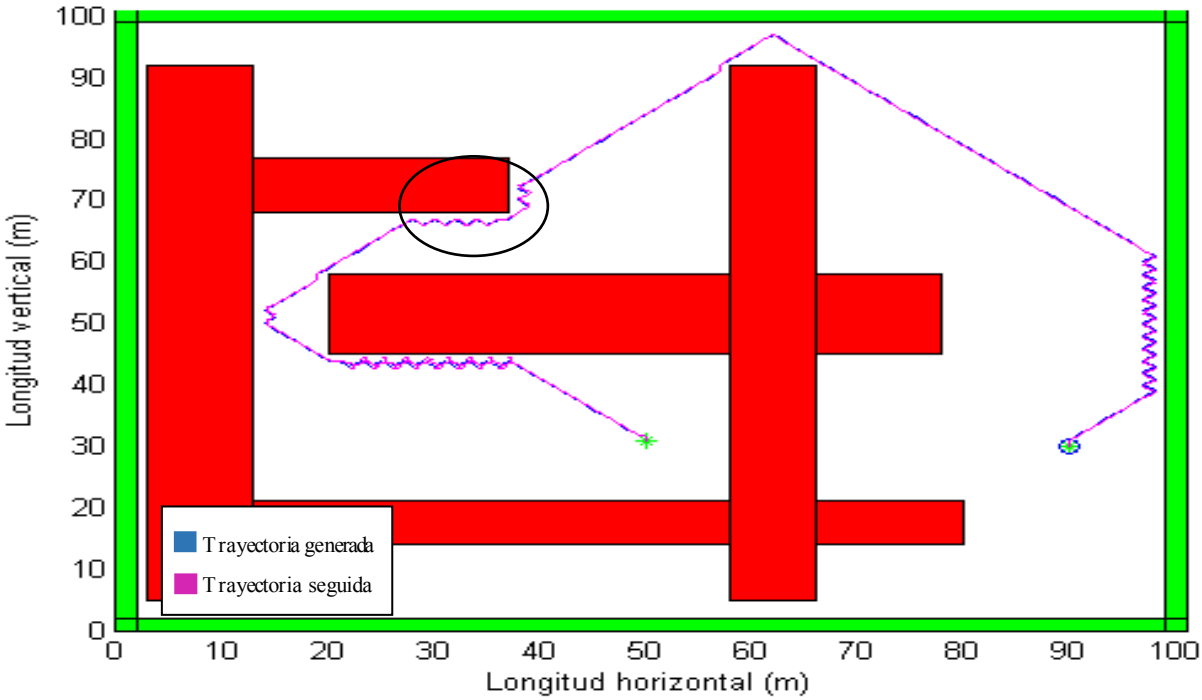


Figura 5-76. Seguimiento de trayectoria con robot biciclo en mapa 1 (Frente de ondas).

En la imagen anterior vemos como no podemos saber si la trayectoria es válida a simple vista, ya que en la zona seleccionada hay un punto que podría ser conflictivo. Para aclarar dicha duda ampliaremos esa zona y comprobaremos si se produce colisión en la misma o no. Por otro lado podemos ver como el robot recorre la trayectoria a una velocidad similar a con la que recorre las trayectorias generadas por los demás algoritmos exceptuando los grafos de visibilidad y los diagramas de Voronoi, en los cuales el robot va considerablemente más rápido.

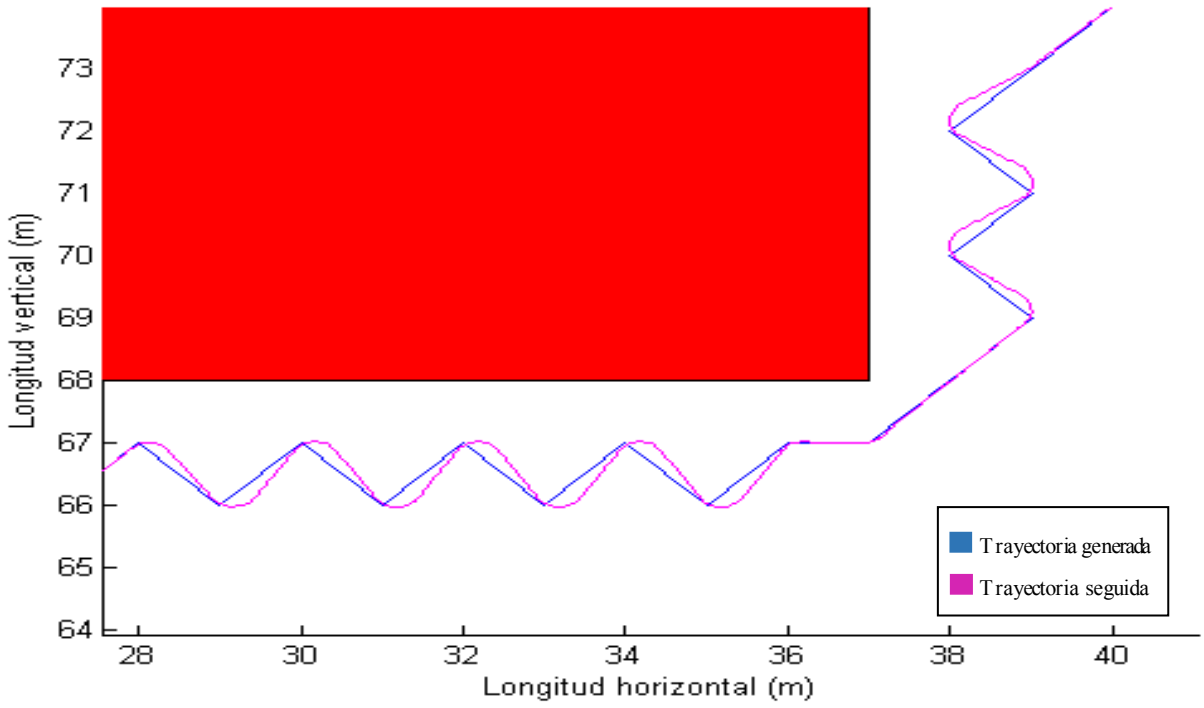


Figura 5-77. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 1 (Frente de ondas).

En la imagen anterior vemos como efectivamente tras ampliar la imagen se ve claramente que la trayectoria es válida ya que en el punto que encontramos la mínima distancia entre un obstáculo y el robot, esta es de más de 0.5 metros.

Para asegurarnos de que las constantes son válidas en distintos mapas, hemos realizado un segundo experimento.

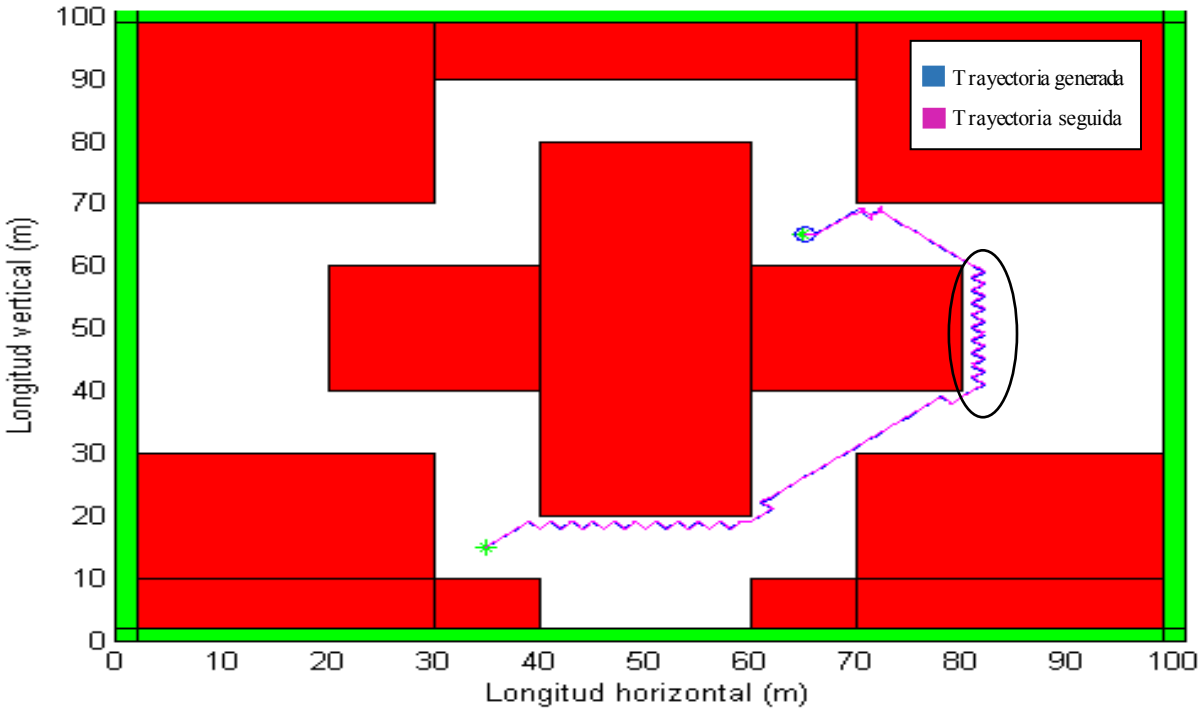


Figura 5-78. Seguimiento de trayectoria con robot biciclo en mapa 2 (Frente de ondas).

En este experimento al igual que en el anterior encontramos un punto similar al que nos encontramos en la trayectoria del mapa 1, y debido a que en el mapa 1 no existía colisión en este tampoco. Además de esto en el mapa nos encontramos una zona con gran cantidad de zig-zags, los cuales no dan problemas en el hecho de que se produzca un choque, pero sería interesante ver cómo se comporta el robot. Por estos motivos las constantes elegidas serán aceptadas.

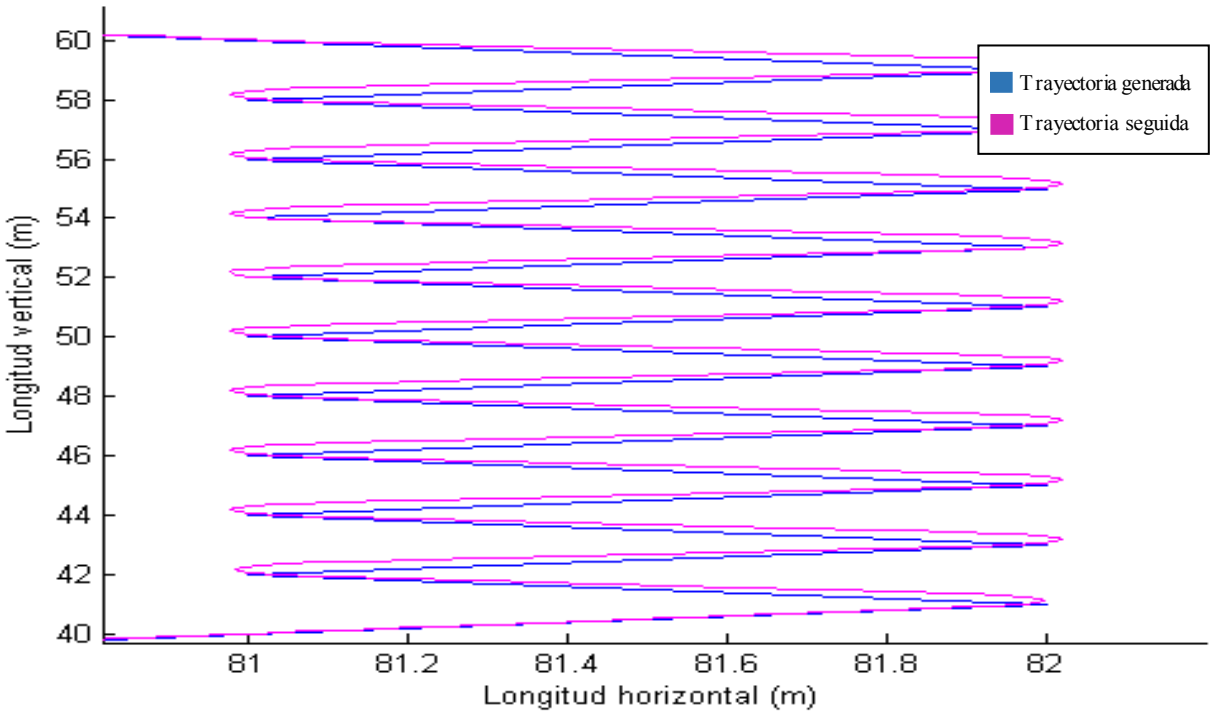


Figura 5-79. Ampliación de seguimiento de trayectoria con robot biciclo en mapa 2 (Frente de ondas).

En efecto vemos como no hay problemas de colisión con el zig-zag pero podemos ver como el hecho de tener que girar tantas veces el robot hace que ambas trayectorias difieran aunque no demasiado.

- Diferencial

A continuación mostramos la tabla con las constantes del robot diferencial cuando sigue la trayectoria generada por el algoritmo de frente de ondas.

Tabla 5–18. Constantes del robot diferencial para frente de ondas

d	Kv	Kh	Ki
0	0.02	10	300

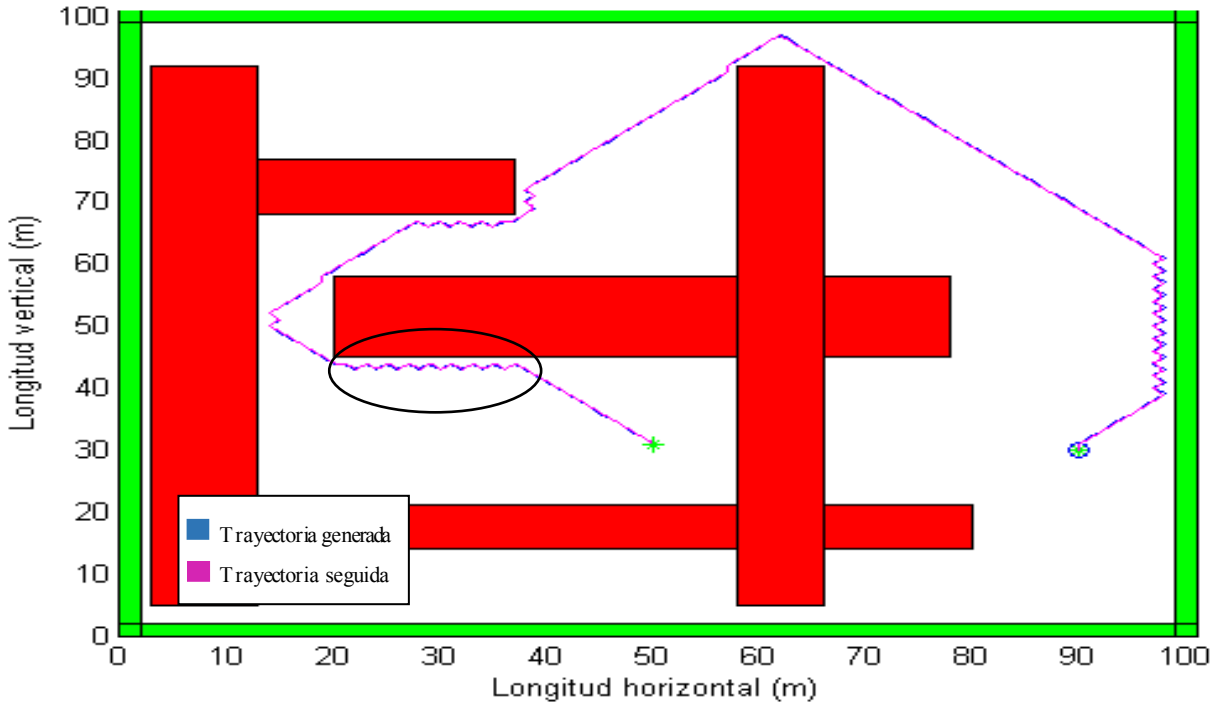


Figura 5-80. Seguimiento de trayectoria con robot diferencial en mapa 1 (Frente de ondas).

En la imagen vemos como no existe ningún punto en el que podamos dudar acerca de si existiese colisión, por este motivo la trayectoria seguida es una trayectoria válida, no obstante, al igual que en el robot anterior ampliaremos una zona la cual consideramos interesante.

La velocidad con la que este robot recorre la trayectoria es ligeramente inferior a la del robot biciclo recorriendo la misma.

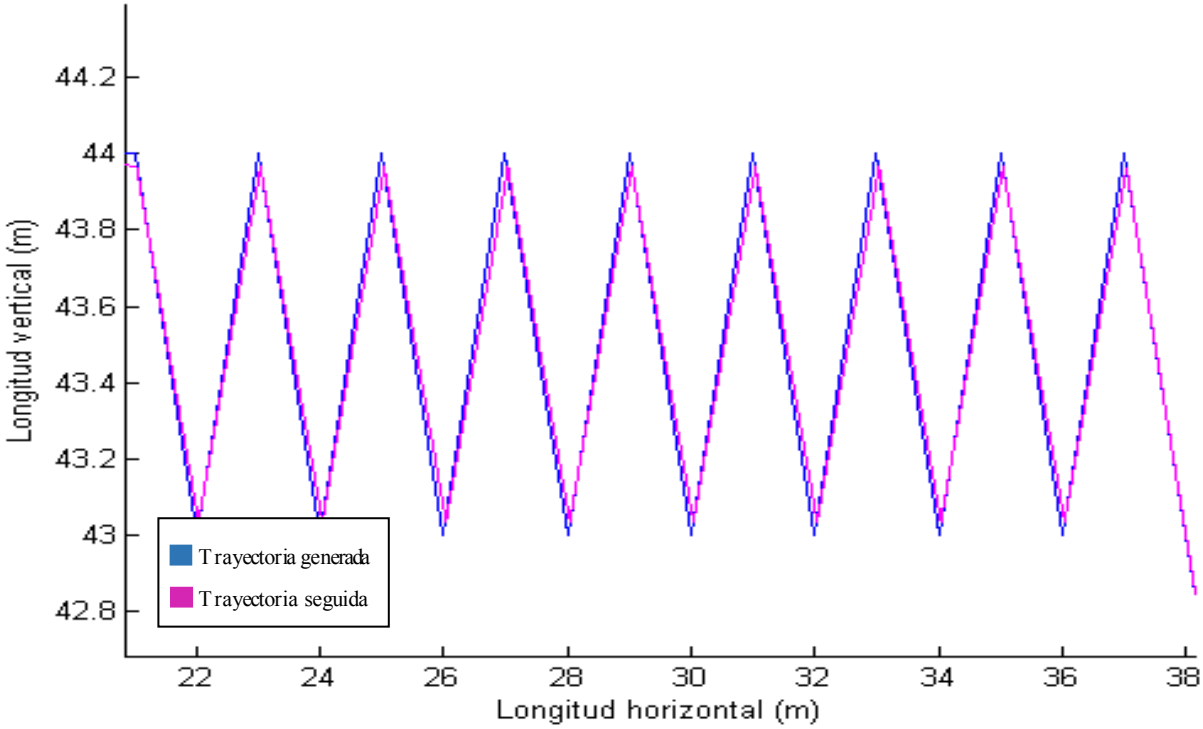


Figura 5-81. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 1 (Frente de ondas).

Tras ampliar la imagen vemos como el robot sigue la trayectoria de manera bastante fiel.

A pesar de que las constantes hayan sido adecuadas para la trayectoria anterior, para asegurarnos de que son correctas realizaremos otro experimento que lo corrobore.

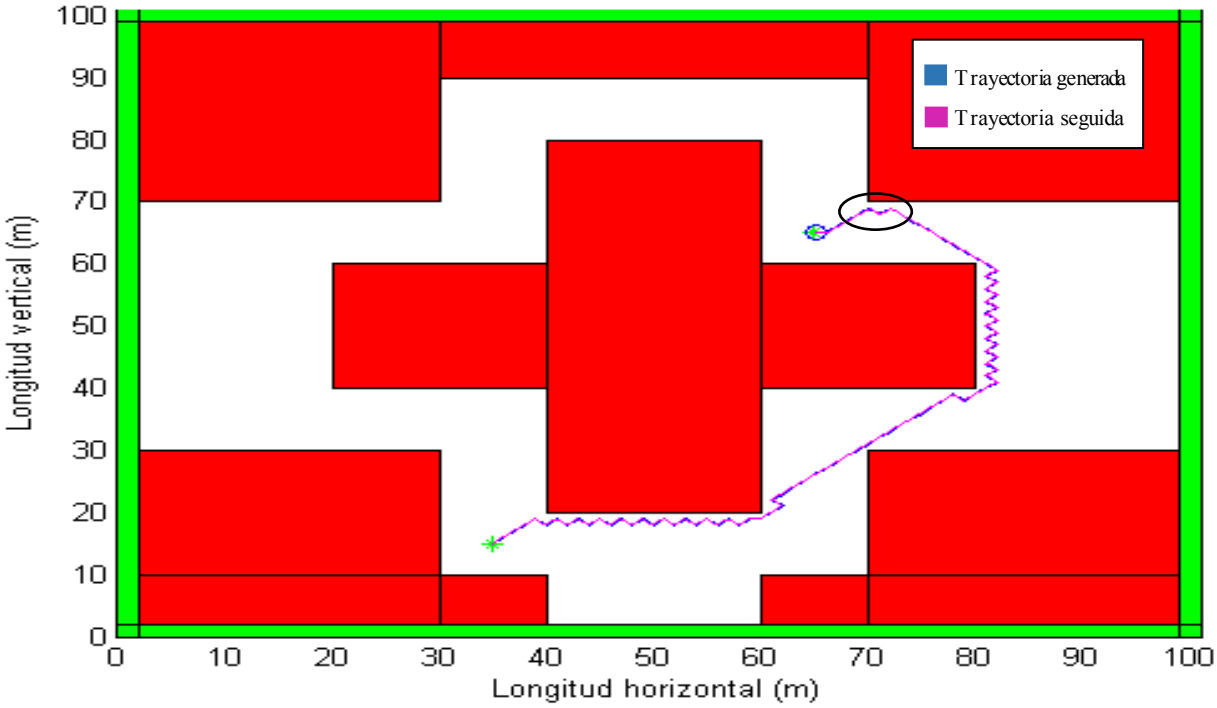


Figura 5-82. Seguimiento de trayectoria con robot diferencial en mapa 2 (Frente de ondas).

Efectivamente la trayectoria seguida es adecuada tal y como preveíamos, por consiguiente las constante usadas son las adecuadas.

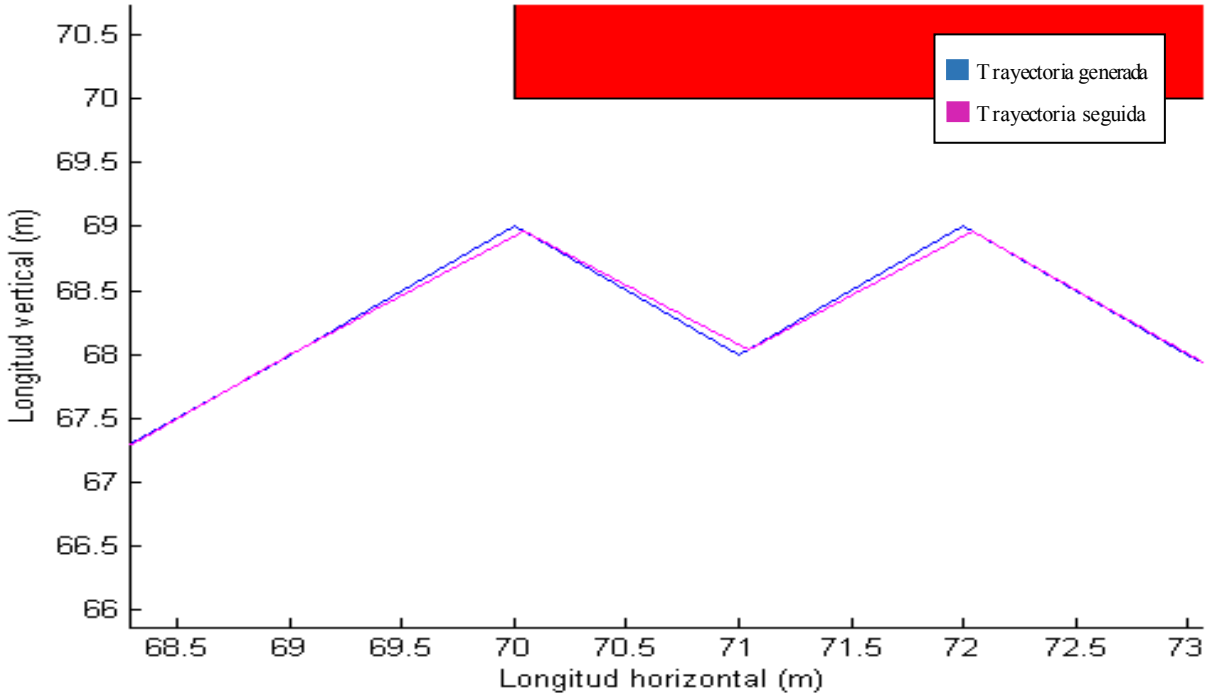


Figura 5-83. Ampliación de seguimiento de trayectoria con robot diferencial en mapa 2 (Frente de ondas).

- Síncrono

  En primer lugar mostramos las constantes con las que el robot síncrono sigue la trayectoria generada por el algoritmo de frente de ondas.

Tabla 5–19. Constantes del robot síncrono para frente de ondas

d	Kv	Kh	Ki
0	0.05	50	200

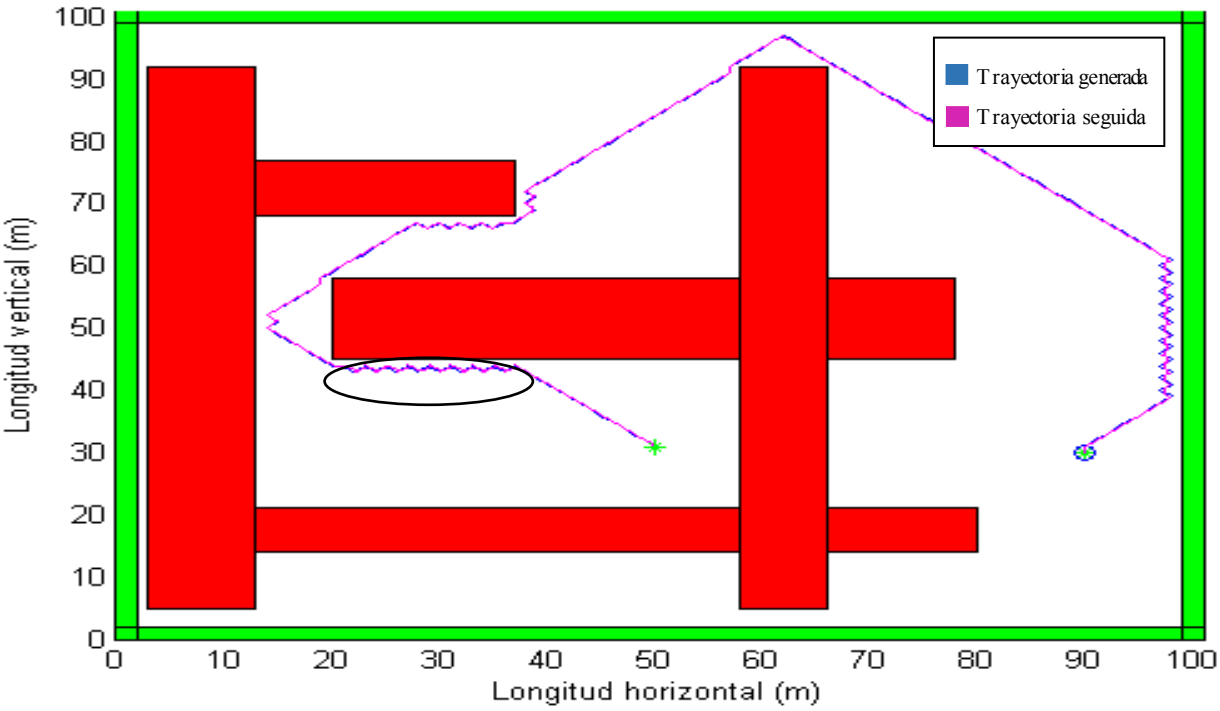


Figura 5-84. Seguimiento de trayectoria con robot síncrono en mapa 1 (Frente de ondas).

La trayectoria que el robot sigue es la adecuada, ya que en ningún momento existe riesgo de colisión, no obstante hemos seleccionado una zona en la que de nuevo volvemos a encontrarnos esos giros en los que en lugar de girar hacia el ángulo menor, prefiere girar hacia el mayor. La velocidad en este caso es algo mayor que en el robot diferencial y bastante similar a la del robot biciclo cuando ambos siguen esta trayectoria.

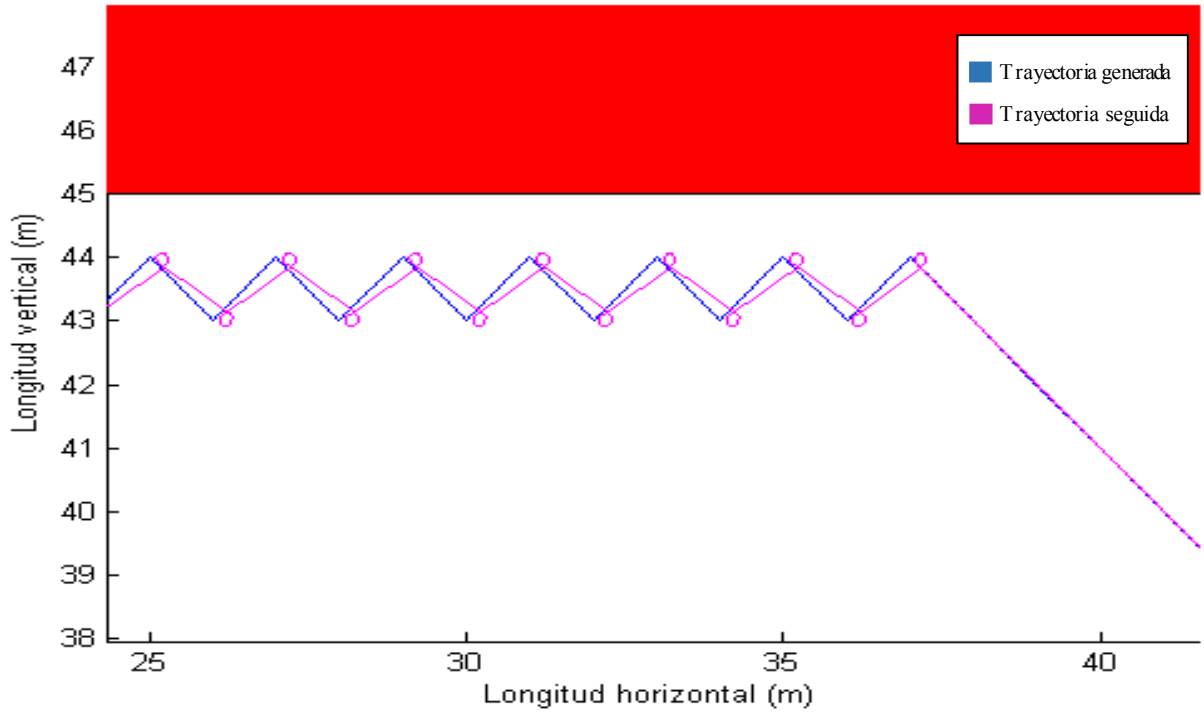


Figura 5-85. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 1 (Frente de ondas).

Como comentábamos anteriormente en la ampliación podemos ver claramente los giros mencionados.

Para confirmar que las constantes elegidas son las adecuadas realizaremos una segunda prueba, esta vez en el mapa 2, donde veremos si la trayectoria seguida es la adecuada.

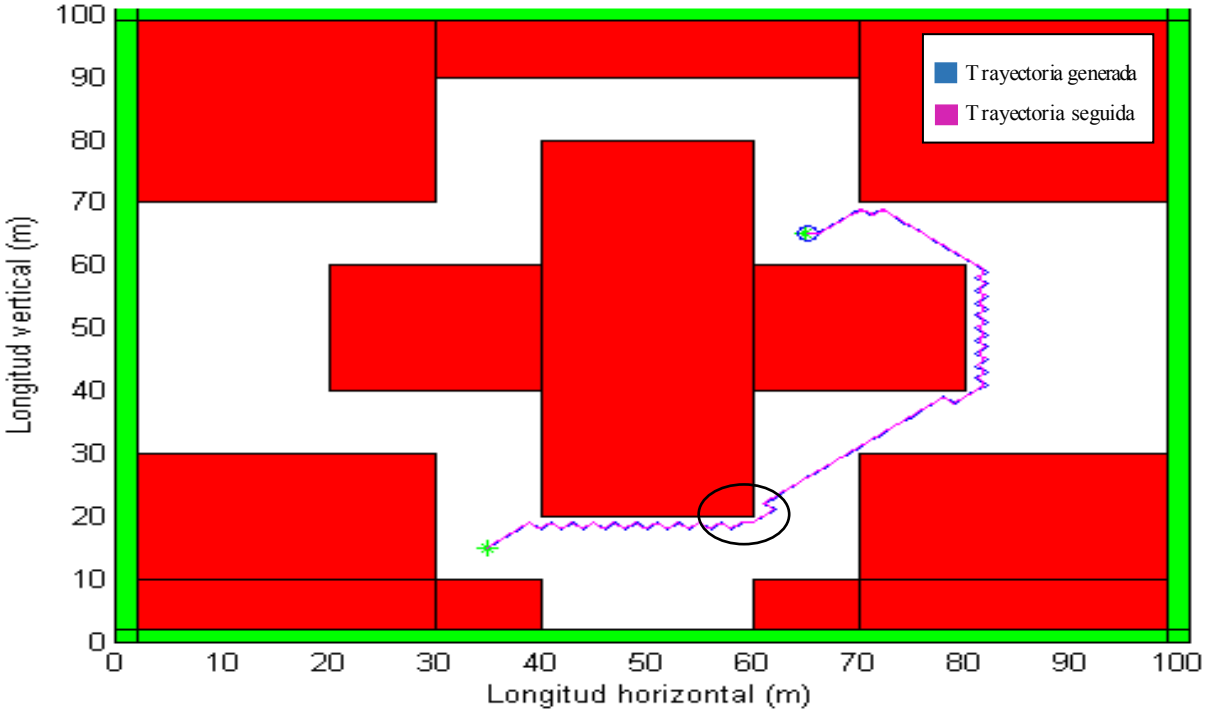


Figura 5-86. Seguimiento de trayectoria con robot síncrono en mapa 2 (Frente de ondas).

De acuerdo con lo que nos imaginábamos, la trayectoria seguida por el robot es válida, por consiguiente las constantes asignadas para este robot cuando sigue una trayectoria generada por el algoritmo de frente de ondas

son correctas. Hemos ampliado una zona para corroborar que la trayectoria que el robot realiza no colisionará con los obstáculos existentes.

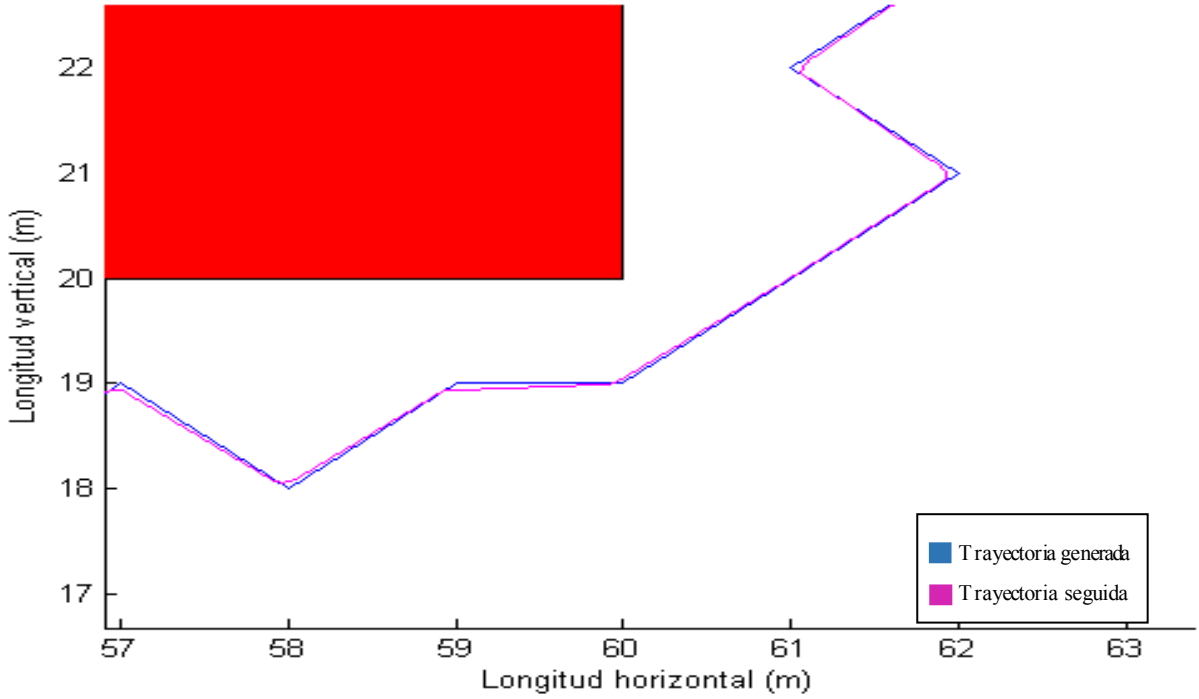


Figura 5-87. Ampliación de seguimiento de trayectoria con robot síncrono en mapa 2 (Frente de ondas).

- Triciclo

  Las constantes elegidas para que el robot triciclo siga la trayectoria generada por el algoritmo de frente de ondas son las siguientes:

Tabla 5–20. Constantes del robot biciclo para frente de ondas

d	Kv	Kh	Ki
0	0.05	10	10

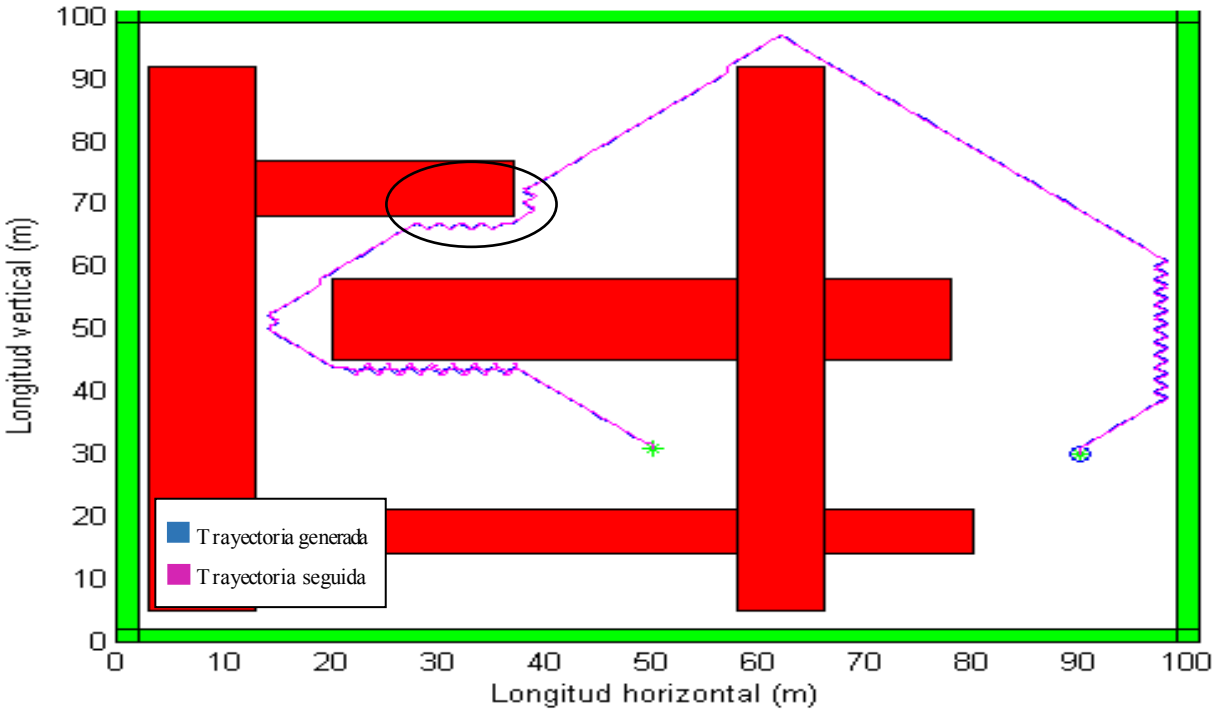


Figura 5-88. Seguimiento de trayectoria con robot triciclo en mapa 1 (Frente de ondas).

Podemos ver en la imagen como la trayectoria aparentemente es válida, a pesar de que podemos encontrar un punto que nos cause ciertas dudas, para despejarlas ampliaremos la zona y veremos si los resultados son adecuados. La velocidad de este robot siguiendo la trayectoria generada por el algoritmo de frente de ondas es similar a la del resto de robots para la misma trayectoria a excepción del diferencial que es una poco menor.

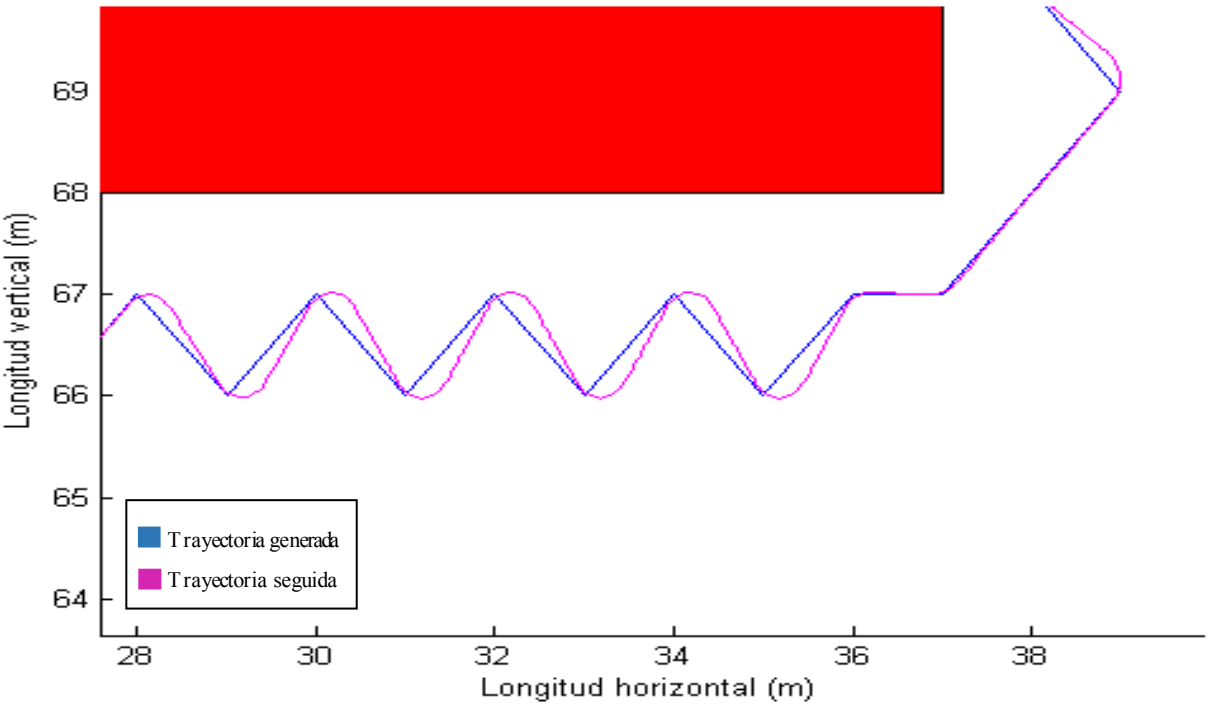


Figura 5-89. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 1 (Frente de ondas).

Tras ver más de cerca el punto conflictivo, vemos como efectivamente la trayectoria es correcta, ya que la mínima distancia entre el robot y los obstáculos es de más de 0.5 metros, por lo que no existiría colisión.

Para poder asegurarnos de que las constantes elegidas son las adecuadas, realizaremos otro experimento y así comprobaremos si la trayectoria realizada por el robot está libre de colisiones para las mismas.

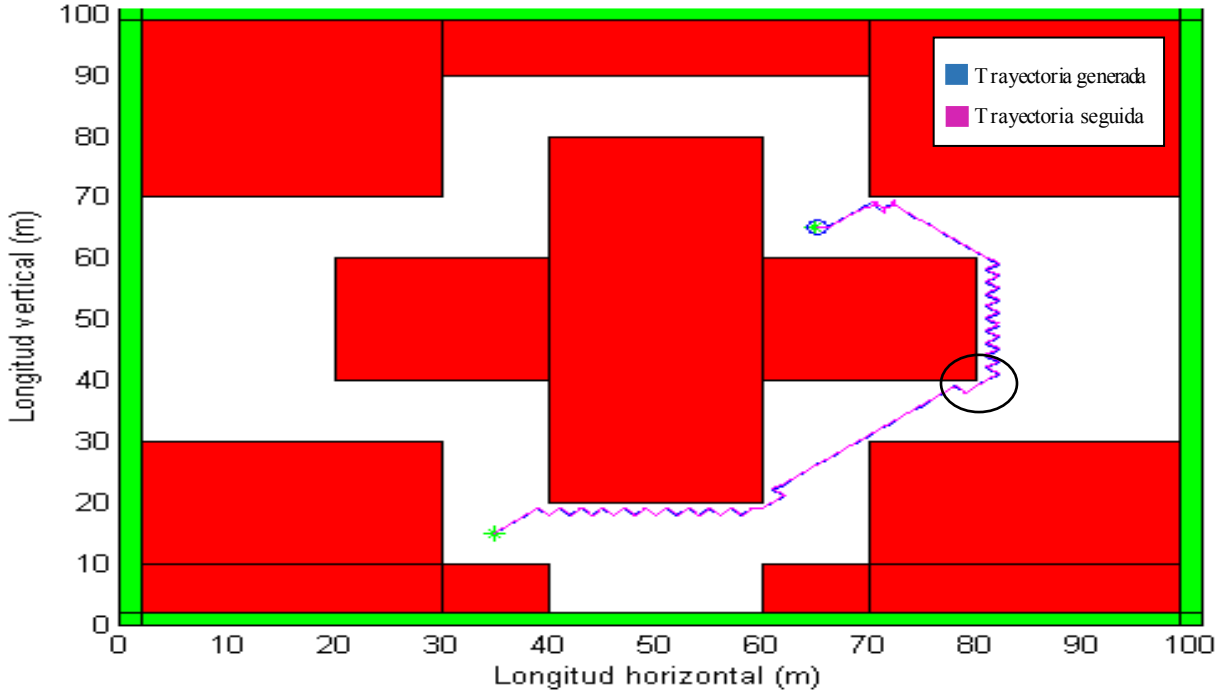


Figura 5-90. Seguimiento de trayectoria con robot triciclo en mapa 2 (Frente de ondas).

Efectivamente vemos como la trayectoria realizada por el robot triciclo es la adecuada, y por consiguiente las constantes elegidas también lo son. Hemos ampliado la zona marcada para mostrar como efectivamente no hay riesgo de colisión.

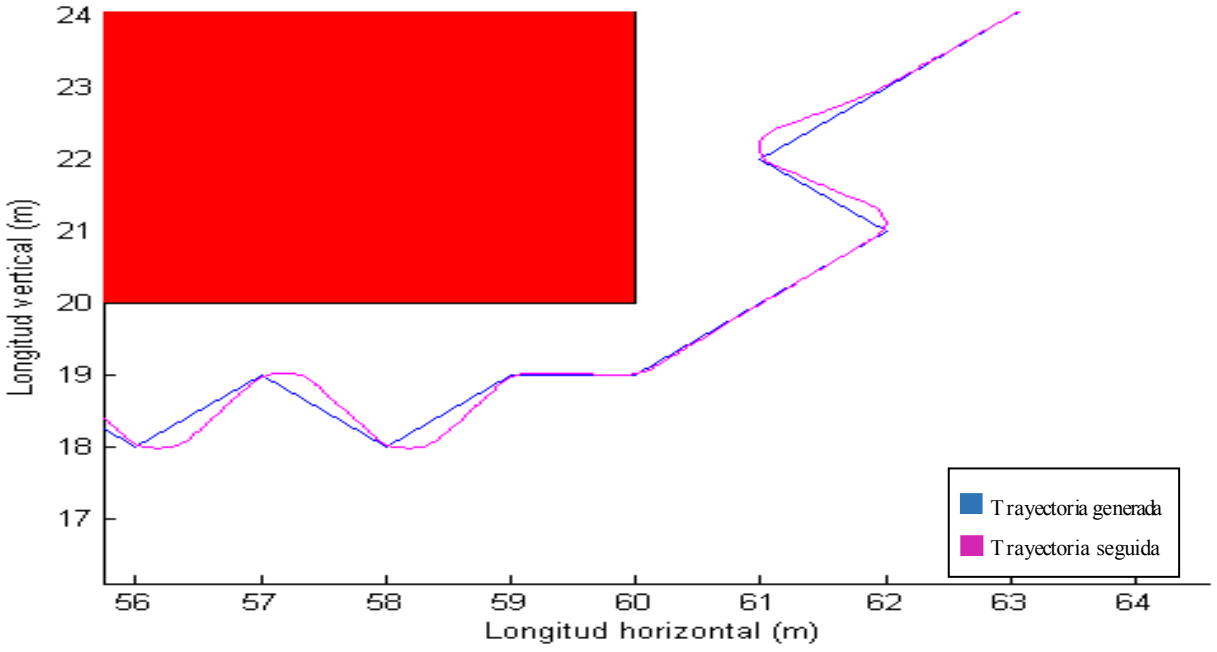


Figura 5-91. Ampliación de seguimiento de trayectoria con robot triciclo en mapa 2 (Frente de ondas).

6 CONCLUSIONES Y TRABAJOS FUTUROS

En este último apartado mostraremos las conclusiones obtenidas tras la realización del trabajo fin de grado en cuanto a los métodos de generación de trayectorias y al seguimiento de las mismas por los distintos robots móviles. Además tras mostrar lo anteriormente citado, expondré los caminos por los que yo pienso que se podría seguir con este trabajo.

En primer lugar comentaré las diferencias que tras la realización del trabajo hemos podido observar en los diferentes métodos de generación de trayectorias y por consiguiente, cuándo unos métodos son mejores que otros.

Una de las diferencias que nos encontramos antes de iniciar el algoritmo y que hace que separemos los algoritmos en dos bloques claros, es la de la necesidad de un tiempo y una capacidad de cómputo previa el funcionamiento del algoritmo de generación de trayectorias. Esto hace que tengamos el bloque formado por los algoritmos que sí necesitan ese cómputo previo, el cual está formado por los grafos de visibilidad, el algoritmo de descomposición en celdas verticales, los diagramas de Voronoi y el planificador de frente de ondas. Dichos algoritmos necesitan conocer el mapa en el que se generará la trayectoria junto con algunos datos más, como los vértices de los obstáculos o donde estos estás situados, previamente a la ejecución de los mismos. Y otro bloque en el que solamente encontramos al algoritmo de Bug 2, que no necesita cómputo previo, ya que solamente necesita conocer el punto inicial y el final y el resto de decisiones las toma con los datos que recibe de los sensores.

Otra de las diferencias que encontramos entre estos algoritmos es la de si los caminos generados son los más apropiados o no. Al igual que en la desigualdad anterior en este caso también podríamos hacer dos bloques, aunque ahora los elementos de cada bloque poseen matices. En un primer bloque podemos encontrarnos con los grafos de visibilidad, el algoritmo de descomposición en celdas verticales y los diagramas de Voronoi, todos estos métodos generan trayectorias óptimas de acuerdo con la forma de calcularla de cada uno. Mientras que en un segundo bloque encontramos al algoritmo de Bug 2 y al de planificador de frente de ondas, el de Bug 2 está claro que no va a generar una trayectoria óptima ya que cuando encuentra un obstáculo lo bordea de forma aleatoria, pero el planificador de frente de ondas no decimos que genera una trayectoria óptima ya que hay momentos en los que realiza zig-zags o giros que no son necesarios, y que podría perjudicar al movimiento del robot. Este tipo recorridos los realiza por el mero hecho de que hay varias celdas subyacentes válidas a las que se podría acceder y el acceso a cada una de ellas es aleatorio.

Partiendo de las dos grandes diferencias anteriores, comentaremos a continuación que método es el más apropiado bajo que circunstancia. Empezaremos comentando los que generan un camino más acertado y posteriormente pasaremos a los que lo generan menos acertado.

En cuanto a los grafos de visibilidad, la trayectoria que estos generan tras separar los puntos de los obstáculos son unas trayectorias bastante apropiadas, en la que la regla que hemos usado para determinar cuál de todas las trayectorias posibles es la que usaremos es la del menor número de puntos. Este tipo de algoritmo es muy útil siempre que el número de obstáculos presentes en el mapa sea reducido, esto es debido a que el coste computacional de este algoritmo de eleva de forma importante con el número de obstáculos presentes, llegando un punto en el que no merece la pena optar por él.

Por otro lado tenemos los algoritmos de descomposición en celdas verticales, en los cuales su mayor virtud es su simpleza. Como ya hemos visto, este algoritmo nos da unos caminos apropiados, bastante buenos en los tramos en los que el desplazamiento es horizontal, ya que este se encuentra en los puntos medios de los obstáculos pero no tanto en los desplazamientos verticales, ya que al igual que en los grafos de visibilidad es necesario separarlo de la pared ya que sino estos chocarían, en este caso bastaría con el punto central. Por lo tanto es un algoritmo útil siempre y cuando en sus trayectorias no existan muchas zonas de desplazamientos

verticales, en cuyo caso sería más útil usar otro algoritmo.

El último algoritmo de los que hemos visto que genera trayectorias óptimas son los diagramas de Voronoi. En los cuales las trayectorias que este genera normalmente son muy útiles ya que es el algoritmo que más se aleja de los obstáculos, posicionándose en todo momento en los puntos centrales de los mismos. Esto normalmente es útil cuando existen muchos obstáculos, pero si los obstáculos son pocos, este algoritmo hace que se den rodeos muy grandes y es preferible en este caso el uso de los grafos de visibilidad.

Uno de los algoritmos que hemos visto que no genera una trayectoria apropiada es el de los planificadores de frente de ondas, en los que como anteriormente hemos dicho el hecho de que en cada movimiento existan varias posiciones válidas hace que en muchas ocasiones las trayectorias generadas tengan tramos difíciles de seguir para los robots. Aun así este es un algoritmo muy fácil de implementar, lo que en según qué situaciones merece la pena.

Finalmente, el último algoritmo que nos queda por comentar es uno que no generan la trayectoria más óptima, aunque en este caso teniendo en cuenta que el motivo es por el ahorro de coste computacional y tiempo de computo previo, en ciertas circunstancias en las que no podamos tener dichas ventajas este algoritmo es muy útil debido a su facilidad de implementar y su bajo requerimiento de capacidad computacional.

En cuanto a los distintos robots móviles, hemos visto como todos ellos son capaces de seguir las trayectorias que se les presenten, aunque con mayor o menor dificultad. A pesar de ello, en los momentos en los que nos hemos encontrado mayores inconvenientes se han modificado las constantes de cada uno de ellos, para que disminuyendo la velocidad el robot sea capaz de seguir de forma suficientemente fiel la trayectoria deseada.

Tras haber finalizado con las conclusiones obtenidas, me gustaría decir que este trabajo se podría continuar teniendo en cuenta obstáculos móviles. De esta forma no solo se usaría un algoritmo que genere una trayectoria para evitar obstáculos fijos, sino que se combinarían varios y así seguir una trayectoria generada por un algoritmo que evita obstáculos fijos y a la vez un algoritmo que detecta la colisión con un objeto móvil y lo sortea. También sería interesante implementar estos algoritmos en robots reales, para así poder comparar los resultados obtenidos con las simulaciones y ver como dichos robots reaccionan en la vida real.

APÉNDICE

La versión de Matlab usada ha sido Matlab2014a.

En cuanto a los programas, encontramos una carpeta llamada programas y un programa de Matlab llamado startup fuera de ella.

Lo primero que debemos hacer antes de ejecutar los programas es cambiar el directorio de Matlab para así ver en Current folder tanto el programa startup como la carpeta programas. Hacemos clic con el botón derecho sobre la carpeta programas y seleccionamos la opción add to path, aparecerá una opción llamada selected folders and subfolders y la seleccionamos.

Tras ello tan solo tenemos que ejecutar el programa startup y seguir las instrucciones que nos van apareciendo en la ventana de comandos. Dichas instrucciones lo que nos pedirán es que seleccionemos un algoritmo entre una lista que nos muestra, un robot entre los que aparece y un tipo de mapa.

REFERENCIAS

[1] Control de robots móviles, apuntes clase.

[2] Pagina web, <http://clipset.20minutos.es/starship-el-robot-de-reparto-urbano-sobre-ruedas/>.

[3] Peter Corke, Robotics, Vision and Control.

[4] Carlos Cobos Bandera, Simulación y Comparación de Algoritmos de Evitación de Obstáculos para Múltiples Robots, TFG, US, 2015.

[5] Anibal Ollero, Planificación de movimientos basada en modelo, Apuntes clase.

[6] Modelos de robots móviles, apuntes clase.

[7] Pagina web, <http://dmi.uib.es/aortiz/mobots_navegacion.pdf>.

[8] Pagina web, <http://www.esi2.us.es/~vivas/ayr2iaei/DET_PLAN_CAMINOS.pdf>.

[9] Pagina web, <http://webpersonal.uma.es/~VFMM/PDF/cap3.pdf>.

[10] Pagina web, <http://gro.usal.es/web/gradosSalamanca/gradoMG.pdf>.

[11] Pagina web, <http://www.bdigital.unal.edu.co/28853/1/26748-93667-1-PB.pdf>.

[12] Pagina web, <http://eprints.sim.ucm.es/11301/1/MemoriaProyectoSSII.pdf>.

[13] Pagina web, <http://webpersonal.uma.es/~VFMM/PDF/cap3.pdf>.