Universidad Zaragoza
1542

Trabajo Fin de Grado

DISEÑO DE UN VEHÍCULO AUTÓNOMO CON CARACTERÍSTICAS SLAM

DESIGN OF AN AUTONOMOUS VEHICLE WITH SLAM FEATURES

Autor

Eduardo Fraguas Bordonaba

Director

Daniel Mercado Barraqueta

ESCUELA DE INGENIERÍA Y ARQUITECTURA

UNIVERSIDAD DE ZARAGOZA

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

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

D./Dª. EDUARDO FRAGUAS BORDONABA,

con nº de DNI 73090871-E en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO, (Título del Trabajo) DISEÑO DE UN VEHÍCULO AUTÓNOMO CON CARACTERÍSTICAS SLAM,

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

Zaragoza, 26 de Agosto de 2018

Fdo:

DISEÑO DE UN VEHÍCULO AUTÓNOMO CON CARACTERÍSTICAS SLAM

RESUMEN

Dotar de autonomía a un robot móvil para reconstruir y explorar su propio entorno, sin necesidad de establecer condiciones iniciales para su operación es un tema amplio de estudio en la comunidad académica y científica relacionada con el área de la navegación de robots móviles. Además de tratar de resolver el problema de la navegación, enmarcado en la generación autónoma de trayectorias y evasión de obstáculos dinámicos y estáticos.

En este proyecto se pretende dar una solución a este problema abordando el diseño de un vehículo autónomo y los algoritmos que lo van a gobernar.

Para ello, es importante en primer lugar realizar una selección de sus componentes que permitan posteriormente la implementación de los algoritmos de control de manera eficaz.

Los algoritmos mencionados abordan el problema del mapeo del entorno por el que avanza el vehículo, el posicionamiento del vehículo sobre el mapa creado y la selección de trayectorias a seguir para alcanzar un destino seleccionado con anterioridad.

La realización de los algoritmos se van a realizar sobre un simulador del vehículo en el programa Processing, simulador que se realiza desde cero para este proyecto y que va a ser la base para hacer la comprobación de todas las implementaciones que se le quieren realizar al vehículo diseñado.

DESIGN OF AN AUTONOMOUS VEHICLE WITH SLAM FEATURES

ABSTRACT

To provide a mobile robot with autonomy to reconstruct and explore its own environment, without needing to establish the initial conditions for its operation is a subject of great study within the academic and scientific community related to the area of mobile robot navigation. In addition to solving the navigation problem, framed in the autonomous generation of trajectories and evasion of dynamic and static obstacles.

This project aims to provide a solution to this problem by addressing the design of an autonomous vehicle and the algorithms that will govern it. First, make a selection of its components that allows us to implement the control algorithms effectively.

The aforementioned algorithms address the problem of mapping the environment through which the vehicle advances the positioning of the vehicle in the created map and the selection of trajectories to follow to arrive at a previously selected destination.

The realization of the algorithms will be done in a simulator of the vehicle in the Processing program, a simulator that is done from scratch for this project and that will be the basis to verify all the implementations that you want to design the vehicle.

ÍNDICE

1 INTRODUCCIÓN

El mapeo robótico es la rama de la cartografía que se ocupa del estudio y la aplicación de la capacidad de construir un mapa o plano de planta por el robot autónomo

Evolutivamente, la acción ciega puede ser suficiente para mantener con vida a algunos animales. Para algunos insectos, por ejemplo, el medio ambiente no se interpreta como un mapa y sobreviven sólo con una respuesta disparada. Una estrategia de navegación un poco más elaborada mejora notablemente las capacidades del robot. Los mapas cognitivos permiten capacidades de planificación y el uso de las percepciones, eventos memorizados y consecuencias esperadas.

La realización de mapas con robots móviles no es una tarea sencilla. Implica la construcción de un robot con todos los elementos sensoriales y actuadores necesarios para captar el entorno en el que se mueve. Por otro lado, tampoco son sencillos los algoritmos que determinan tanto su navegación como la detección del entorno.

1.1 LA NAVEGACIÓN

La navegación es una rama fundamental de la robótica móvil, presente en casi todos los robots con cierto nivel de complejidad. Se define esta como la capacidad de dado un punto de partida A, alcanzar los puntos de destino B utilizando su conocimiento y la información sensorial que recibe.

De esta forma, todo robot móvil que pretenda cambiar de posición con cierto grado de autonomía necesita contar con un sistema de navegación más o menos complejo, dependiendo de aspectos como la complejidad del sistema locomotor, los grados de libertad en el movimiento o la complejidad del entorno. [1] La navegación implica resolver los siguientes subproblemas:

- Percepción: Interpretar los datos que le suministran sus sensores para extraer información útil.
- Localización: Determinar su posición en el entorno.
- Planificación: Decidir cómo actuar para alcanzar el objetivo.
- Control de Movimiento: Gestionar sus actuadores para conseguir la trayectoria deseada.

1.2 MAPEO

Se conoce como mapeo al proceso de obtención/generación del mapa del entorno necesario para la navegación de robots móviles. Existen diferentes métodos de mapeo, definiéndose el SLAM (Localización y Mapeo Simultáneo) como el método usado cuando se genera el mapa mediante una exploración del entorno. Como su nombre indica, consiste en realizar, en paralelo al proceso de mapeo, un proceso de localización ya que la posición es una variable necesaria a la hora de generar el mapa con este método.

Figura 1. Esquema de aspectos que aborda el SLAM.

La solución a este problema ha sido objeto de estudio por parte de la comunidad científica durante los últimos 20 años. El ruido presente en sistemas sensoriales, los inevitables errores y aproximaciones cometidos en los modelos empleados y la dificultad representativa de los entornos a medida que éstos aumentan en complejidad hace que la tarea de resolver el mencionado problema sea ardua [2].

1.3 MOTIVACIÓN

La principal motivación de este proyecto reside en la curiosidad y el interés sobre la robótica móvil y las constantes soluciones y novedades que se encuentran en la actualidad sobre la temática.

Conocer con más concreción y amplitud la disciplina, así como entrar en contacto directamente con los algoritmos de control con los que se está trabajando a día de hoy resulta un reto muy interesante. Además el poder proyectar parte de los conocimientos adquiridos durante el grado en el proyecto, más concretamente los de

programación, los cuales se abordaron durante el primer curso y me parecieron especialmente interesantes.

Con todos estos alicientes y la motivación de mi profesor y tutor del proyecto Daniel Mercado por este aspecto de la robótica, surgió la idea para abordar este trabajo.

Por otro lado y debido a que existen muchas opciones comerciales o con dependencia de herramientas externas de alto nivel de abstracción del proceso, otra de las motivaciones del proyecto es conseguir los objetivos planteados de una manera más económica e independiente. Tanto a nivel de software y control del proceso como de hardware del robot, combinando la economía con la fiabilidad del proyecto.

1.4 OBJETIVO Y ALCANCE

El objetivo principal de este proyecto consiste en el diseño de un robot autónomo dotado de los componentes necesarios con los que poder implementarle un proceso SLAM eficaz en una habitación o recinto, sin necesidad de ser guiado ni corregido por acciones externas. Para ello se realizará un análisis de las posibilidades que tenemos a la hora de dotar al robot de elementos sensores, actuadores y de control, con el que ser capaces de determinar justificadamente los componentes finales con los que contará el vehículo diseñado.

Se pretende que el sistema sea capaz de desplazarse sin colisionar hasta un punto determinado sin tener conocimiento previo de su entorno, dotándole de un algoritmo de selección de trayectorias. Dentro todo de ambientes de interior.

Como se ha mencionado en la motivación del proyecto, se busca no depender de herramientas externas que permiten abstraernos del proceso SLAM y sus algoritmos, además, se busca que sea capaz de implementarse en una placa portátil, sin necesidad de conexiones remotas con otros procesadores.

Es decir, se pretende diseñar el vehículo de manera que resulte independiente de frameworks (entornos de trabajo con asistencia definida) o softwares de mayor nivel, incidiendo en los niveles más bajos de control de manera directa; e independiente de procesadores grandes externos al vehículo. Para ello es muy importante realizar una buena selección de los componentes del vehículo y realizar los algoritmos de manera que se ahorre memoria y capacidad de procesamiento.

2 ESTADO DEL ARTE

En este apartado se presenta en mayor profundidad el concepto de SLAM y algunos de los problemas que condicionan su estudio y desarrollo.

2.1 JUSTIFICACIÓN DE LA TÉCNICA

El problema del SLAM se aplica cuando el robot no tiene acceso a un mapa del entorno ni conoce tampoco su posición en el mismo. Por ejemplo, se puede dar la situación en la que se tenga que utilizar un robot para tareas de rescate en una zona donde el humano no puede acceder, en este caso el robot no posee conocimiento de su ubicación ni del ambiente.

Por tanto, el robot solo dispone de la información proporcionada por las medidas obtenidas de los sensores y la noción de movimiento propio. El robot intentará obtener un mapa del entorno y simultáneamente localizarse en dicho mapa.

Una primera solución al problema es la conocida como odometría o *dead reckoning*. Esta consiste en ir estimando la posición actual a partir de la velocidad con la que se mueve el vehículo. La ventaja principal de este método es que su implementación es relativamente sencilla y, en la mayoría de los casos, no depende del entorno.

Este método para estimar la posición tiene un gran problema. En cada cálculo que se realiza para actualizar la posición se introducen errores. Estos pueden deberse a diversos motivos: imperfecciones en la construcción del robot, errores en las mediciones, eventos no contemplados por el método de estimación (como el deslizamiento de una rueda), etc. A corto plazo estos errores pueden no ser importantes, pero a medida que pasa el tiempo estos crecen de forma no acotada resultando en una estimación de poca confianza.

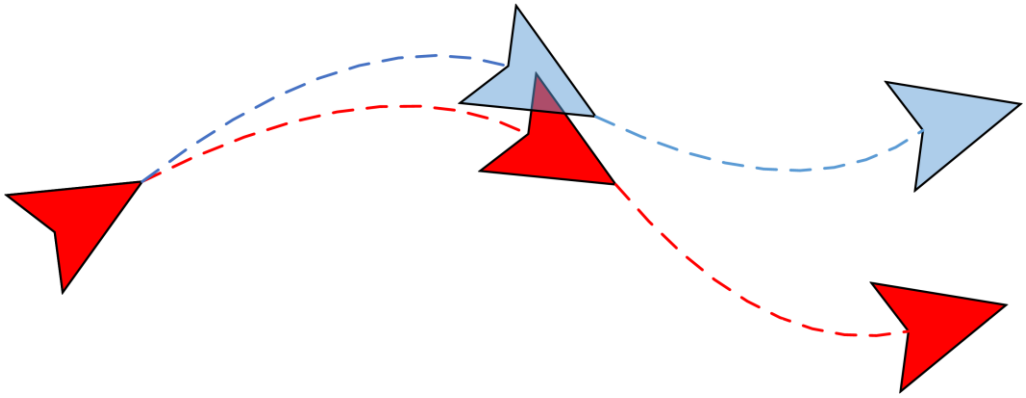

Figura 2. Trayectoria de un robot móvil, real (rojo) y calculada por la odometría (azul).

La manera de solucionar el problema de la odometría es utilizar un mapa y las características del entorno para corregir la estimación.

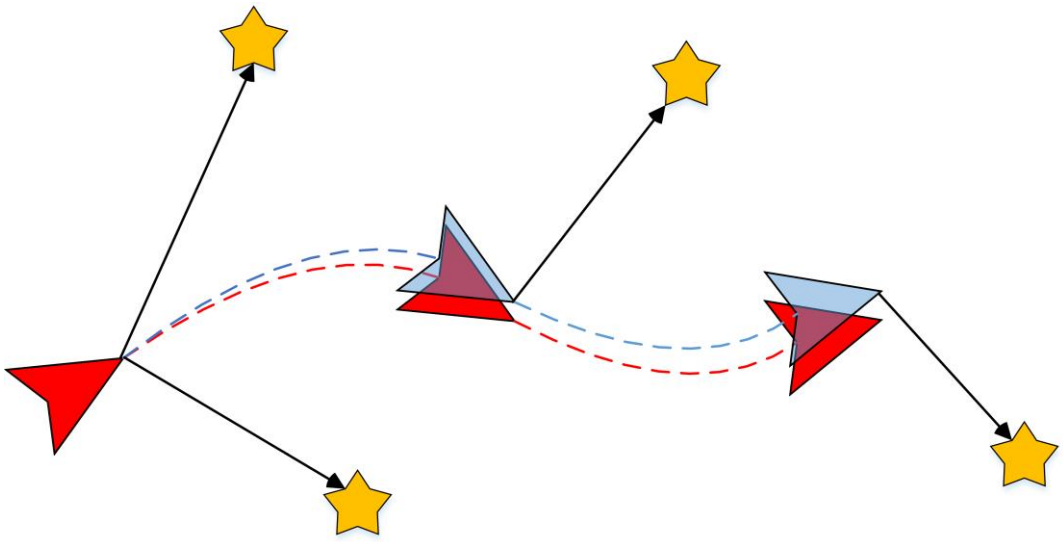

Figura 3. Trayectoria de un robot móvil, real (rojo) y calculada por la odometría con corrección según la observación del entorno (azul).

En ese caso, deberá ser el propio robot el que seleccione qué elementos del entorno deberán usarse como referencias. Esta es una elección muy importante, ya que de ello dependerá el éxito de la localización. Esas características deberán de ser extraídas de los datos proporcionados por los sensores. Dicho proceso de extracción se conoce por su nombre en inglés *feature extraction*, es decir, extracción de características.

En resumidas cuentas, lo que se quiere es poder localizar al robot en un mapa que no existe, y que por lo tanto, deberá de crear el propio robot.

Si uno se fija, se tiene un problema. Con lo visto hasta ahora, se sabe que para localizar el robot se necesita un mapa, y para construir un mapa se necesita saber la localización del robot. A este tipo de problemas se les conoce como el problema del huevo y la gallina por la similitud del dilema.

La solución que se propone es el SLAM, considerado el santo grial de la robótica móvil, porque su logro supone una autonomía real de navegación en esta disciplina [3].

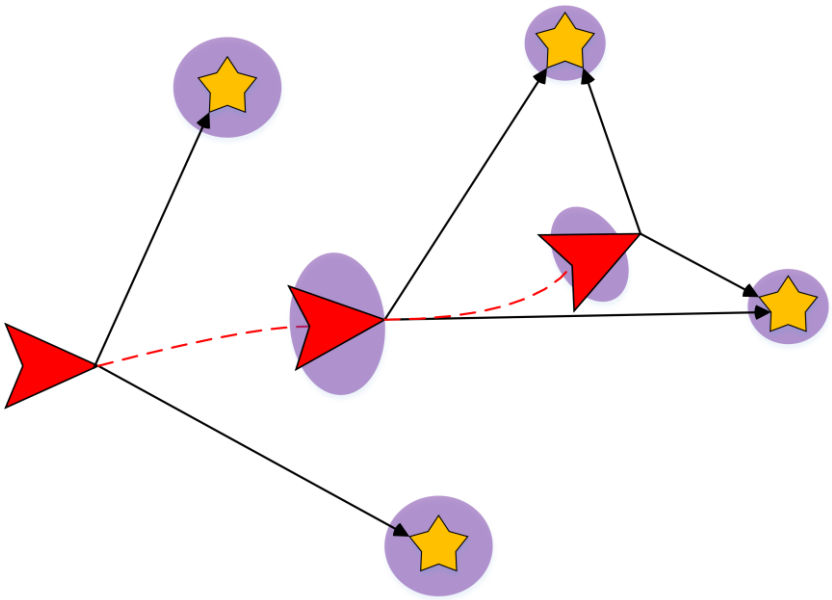

Figura 4. Trayectoria corregida y mapeo simultáneo.

2.2 MANEJO DE LA INCERTIDUMBRE

En el contexto del SLAM existen diversas fuentes de incertidumbre, es decir factores que incrementan la dificultad de estimar correctamente la ubicación y el mapa del entorno. A continuación se incluyen algunos de estos factores [4]:

- Ruido de los sensores: Los sensores utilizados en un robot presentan usualmente ruido en los datos que proporcionan.

- Desplazamiento impreciso del robot: El resultado de un movimiento del robot es de naturaleza no determinista. Esto se debe a que, por ejemplo, las ruedas de un robot pueden deslizar sobre el suelo. Como resultado, no es posible conocer a ciencia cierta cómo fue el movimiento real del robot como resultado de una orden de movimiento (p. e. avanzar, retroceder, girar) o partir de la información de la odometría.

- Simetrías en el entorno: El entorno sobre el que opera el robot puede presentar simetrías que dificultan la determinación de la posición actual del robot en el mapa confeccionado.

- Observabilidad parcial: La ausencia de un mecanismo de visión global del entorno dificulta la estimación de la posición y la confección del mapa.

- Entorno dinámico: Si se trabaja en un entorno dinámico, los cambios en el entorno dificultarán el proceso de estimación de la posición debido a que el mapa confeccionado puede estar desactualizado.

En pos de adaptarse a esta incertidumbre, la gran parte de las soluciones de SLAM plantea la respuesta de la estimación de la posición del robot y el mapa del entorno como distribuciones de probabilidades.

En la actualidad existen métodos de SLAM robustos para mapear ambientes estáticos, estructurados y de tamaño limitado. Realizar SLAM en entornos dinámicos, espacialmente grandes y desestructurados sigue siendo un problema abierto a la investigación [4].

2.2.1 SENSORES

Los sensores son limitados en lo que pueden percibir, y poco precisos en sus medidas. Estas limitaciones vienen dadas por muchos factores. El rango y la resolución de un sensor están sujetos a limitaciones físicas. Un claro ejemplo son las cámaras cuya calidad de imagen es limitada.

Los sensores además están sujetos a ruido estocástico, lo que perturba las medidas de formas impredecibles y hace que la información extraída sea poco fiable.

Otro problema que incrementa el ruido en los sensores es el hecho de realizar medidas con el robot en movimiento, hecho que genera mayor error en las medidas de los sensores.

2.2.2 ACTUADORES

En la fase de exploración del SLAM se envían órdenes de movimiento al robot, conocidas como controles. Una vez que se le envía una orden de movimiento al robot,

los sensores en los motores (odometría) permiten conocer cuál ha sido el movimiento que hizo el robot a partir de la orden que fue emitida. Esta información luego se procesará para actualizar la posición del robot.

El problema ocurre cuando una de las ruedas del robot queda en el aire o simplemente resbala sobre una superficie con baja adherencia, eso implica que el robot no cambió su posición, o lo hizo pero en menor medida de la esperada. Sin embargo los datos devueltos por los sensores de odometría indican que el movimiento fue completo.

Esto genera un error entre la posición real del robot y la posición estimada. En la *figura 5* se puede ver un robot con una rueda en el aire, producto de las irregularidades del entorno en el que se encuentra. Esta es una clásica situación donde la información de odometría no se corresponde con el movimiento real del robot.

Figura 5. Situación que da error a la odometría.

2.2.3 ASOCIACIÓN INCORRECTA Y CAPACIDAD DE CÓMPUTO

El algoritmo de SLAM debe ser lo suficientemente robusto como para no confundir dos lugares diferentes de forma que lo lleve a creer que son el mismo. En caso de que el robot identifique que dos lugares diferentes con características similares son efectivamente el mismo, errará un ciclo donde no lo hay y generará un mapa incorrecto del entorno que puede llevar a errores catastróficos.

Una gran limitación del SLAM es el procesamiento. Los algoritmos de SLAM suelen realizar tareas de procesamiento pesadas debido a la densidad de la representación del mapa y a la complejidad de los cálculos involucrados en la estimación de la posición. Esto, muchas veces, hace difícil el procesamiento dentro del robot y obliga a extraer los datos recogidos para ser procesados externamente [4].

2.3 PRINCIPALES ALGORITMOS PROBABILÍSTICOS

2.3.1 FILTRO EXTENDIDO DE KALMAN

La utilización de un Filtro Extendido de Kalman es una de las soluciones más extendidas al problema de la localización y mapeado simultáneo, y también es una de las que mejores resultados ha proporcionado en la práctica. A esta categoría de soluciones, cuyo fundamento se enraíza en los trabajos introductorios realizados por *Randall Smith, Matthew Self y Peter Cheeseman* a finales de la década de 1980, se la conoce generalmente como SLAM-EKF.

Por su propia naturaleza, el SLAM-EKF requiere disponer de un mapa en el cual las entidades que lo componen sean fácilmente parametrizables. Esto es, los elementos que componen el mapa deben poder ser descritos utilizando un conjunto de parámetros que encajen de forma sencilla en el vector de estado del sistema.

A la vista de las hipótesis realizadas, cabe enumerar las siguientes desventajas de esta clase de algoritmos:

- La premisa de partida del algoritmo, que supone una distribución gaussiana para el estado del sistema, puede no corresponderse con la realidad. En el mejor de los casos, las linealizaciones introducidas en los modelos harán que las estimaciones de los momentos de esta distribución degeneren, a lo largo del tiempo, no correspondiéndose con sus auténticos valores.

- El punto anterior indica además un grave problema de consistencia del algoritmo, que hace que el nivel de confianza de la estimación obtenida no se corresponda con el auténtico error cometido. Esto origina que la exactitud de los resultados obtenidos por el filtro sean a menudo impredecibles, observándose saltos bruscos en la estimación sin causa aparente alguna.

- El coste computacional crece al cuadrado con el número de objetos contenidos en el mapa. Este hecho limita su aplicación en tiempo real a mapas formados por unos pocos cientos de objetos.

- No siempre es sencillo o inmediato extraer características del entorno asimilables a una clase particular de objeto. En ocasiones ni siquiera es preciso extraer información que pueda describirse empleando primitivas geométricas simples como puntos, segmentos, arcos de circunferencia o planos (por ejemplo, en el interior de una mina).

- Es preciso disponer de un método de asociación de datos robusto que permita emparejar las observaciones realizadas con los elementos contenidos en el mapa. Una asociación de datos errónea provocará casi con total seguridad la divergencia irrecuperable de la estimación del filtro.

A pesar de los inconvenientes que surgen al emplear esta solución, se trata de la más extendida y presenta interesantes propiedades que resultan muy atractivas:

- El hecho de describir el entorno a partir de entidades geométricas descriptibles de manera compacta, se corresponden más con una visión antropomórfica del mundo, que representa este último a través del concepto del objeto y sus relaciones.

- Estas soluciones cuentan con una larga tradición, por lo que su estructura, ventajas e inconvenientes son bien conocidos. Históricamente se han planteado múltiples soluciones que intentan paliar algunos de los problemas anteriormente mencionados.

- Al mantenerse la matriz de covarianzas del sistema completa, es capaz de cerrar bucles exitosamente.

2.3.2 MAPAS DE OCUPACIÓN DE CELDILLAS

Los mapas de ocupación de celdillas fueron introducidos por *Hans Moravec y Alberto Elfes* a mediados de la década de 1980. El método se basa en discretizar el espacio, dividiéndolo en unidades de tamaño predefinido, que se clasifican como ocupadas o vacías con un determinado nivel de confianza o probabilidad.

Estas soluciones parten de la hipótesis de que la posición del robot es conocida. En la práctica, se necesita de algún método de localización que estime la posición del robot en cada instante que, en este caso, no es considerada una variable estocástica. La precisión que alcanzan estos mapas en la descripción del entorno (tanto mayor cuanto más fina es la división del espacio), permite que el algoritmo de localización empleado acumule errores reducidos a lo largo de intervalos prolongados de tiempo. Así pues, la mayor desventaja de estos métodos es la pérdida de potencia que se deriva de no

tener en cuenta la incertidumbre asociada a la posición del robot, lo cual origina que su capacidad para cerrar bucles correctamente se vea mermada.

Entre sus ventajas cabe destacar las siguientes:

- El algoritmo es robusto y su implementación sencilla.
- No hace suposiciones acerca de la naturaleza geométrica de los elementos presentes en el entorno.
- Distingue entre zonas ocupadas y vacías, consiguiendo una partición y descripción completa del espacio explorado. Por este motivo es popular en tareas de navegación, al facilitar la planificación y generación de trayectorias empleando métodos convencionales.
- Permite descripciones arbitrariamente densas o precisas del mundo, simplemente aumentando la resolución de la rejilla que lo divide (p. ej. disminuyendo el tamaño de las celdillas individuales). Como es lógico, esto va en detrimento del rendimiento computacional del algoritmo.
- Permite una extensión conceptualmente simple al espacio tridimensional.

Recientemente han aparecido variantes de este método que mejoran sustancialmente algunos aspectos. Por ejemplo, los mapas de cobertura que mantienen una probabilidad de ocupación para cada celdilla que se ajusta más correctamente a casos en los que esta se encuentra sólo parcialmente ocupada, o el DP-SLAM, que mantiene varias hipótesis sobre el mapa construido y utiliza un filtro de partículas para estimar la que más se ajusta a la realidad [5].

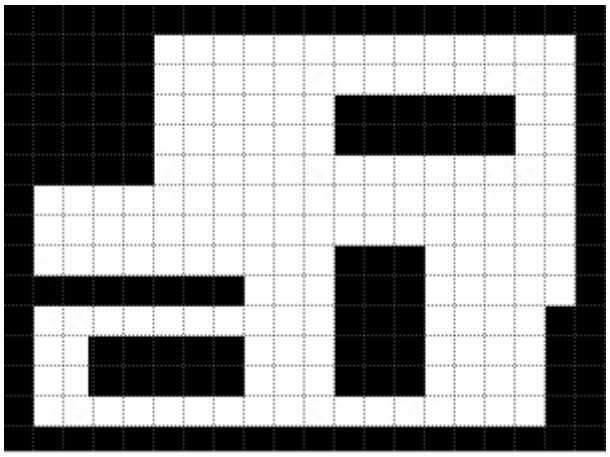

Figura 6. Ejemplo de mapa creado por algoritmo de ocupación de celdillas.

3 SELECCIÓN Y MONTAJE DE COMPONENTES

En este capítulo se hace un análisis de las distintas posibilidades que podemos encontrar a la hora de elegir los componentes necesarios del vehículo autónomo y la justificación de la decisión de selección adoptada. Una vez decididos se explica cómo debería llevarse a cabo el montaje electrónico de los mismos

3.1 SELECCIÓN DE LOS ELEMENTOS SENSORES

Uno de los aspectos más importantes a la hora de diseñar un vehículo autónomo es dotarle de la habilidad para captar el entorno de manera eficaz. Los elementos sensores son los componentes encargados de recoger la información del entorno y convertirla en señales capaces de ser procesadas, señales que se gestionarán y emplearán para poder realizar tanto el mapeado como la navegación.

3.1.1 SENSORES ACÚSTICOS (SONAR)

El sonar es un dispositivo de medida que típicamente utiliza ondas sónicas para calcular la distancia a un objeto. Su funcionamiento es sencillo: se emite un pulso que dispara una onda, esta rebota con un objeto y vuelve a su origen. Durante todo este tiempo, un pulso ha permanecido a nivel alto. A partir de ese tiempo se mide la distancia al objetivo mediante una sencilla regla en función de la velocidad del sonido.

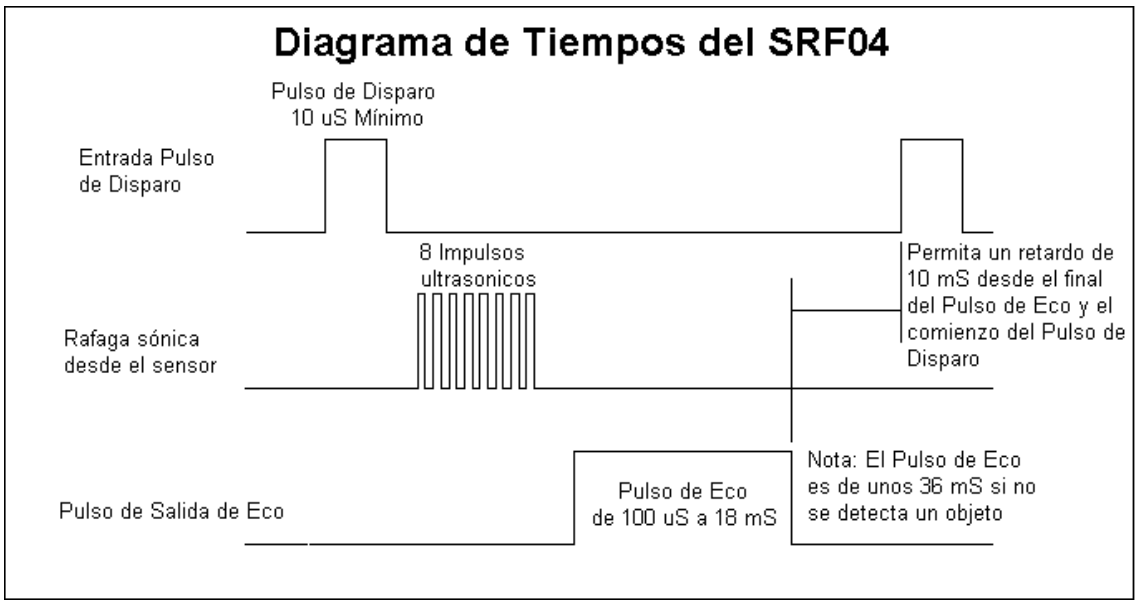

Figura 7. Diagrama explicativo del funcionamiento de un sonar.

El principal problema de los sónares es que suelen ser bastante imprecisos debido a las reflexiones con los objetos lo que hace que se falseen las medidas. Por ello hay que realizar filtros de medidas que sumado a que no suelen ser muy rápidos los convierten en unos sensores algo lentos.

El disparo es en línea recta. Cuando el origen está en movimiento las reflexiones son constantes, la eficacia disminuye aún más y el rango de medidas es limitado [6]. En principio el rango de detección para algunos sónar es de hasta 10 metros, pero son incapaces de detectar objetos pequeños que se encuentran más allá de 5 metros.

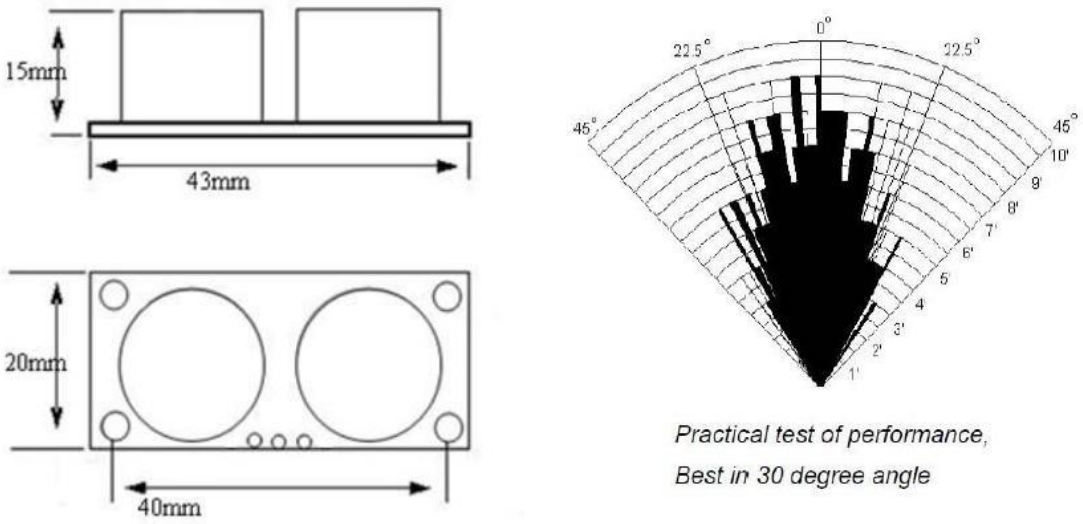

Figura 8. Dimensiones y ejemplo de funcionamiento de un sonar.

3.1.2 SENSORES LASER (LDS)

Son dispositivos que permiten determinar la distancia desde un emisor láser a un objeto o superficie utilizando un haz láser pulsado. La distancia al objeto se determina midiendo el tiempo de retraso entre la emisión del pulso y su detección a través de la señal reflejada o por triangulación entre el foco emisor y el punto de recepción del pulso.

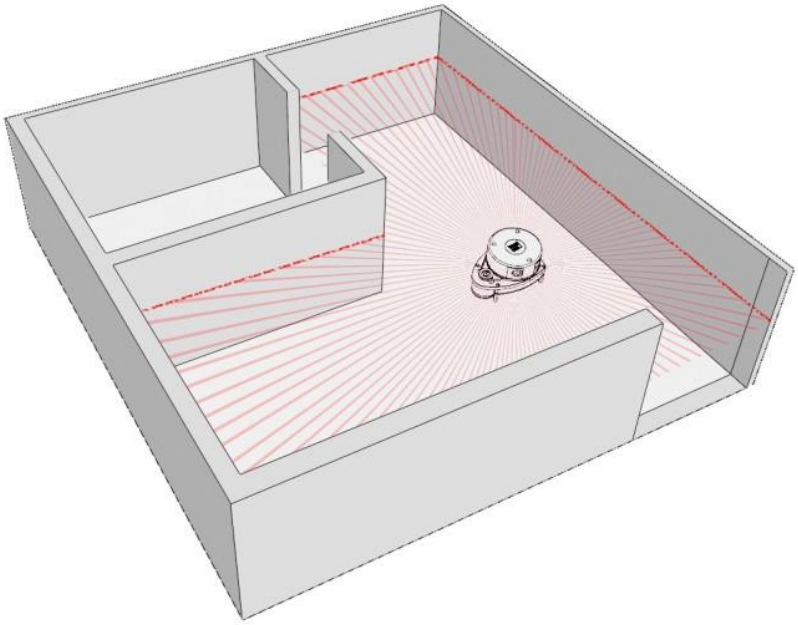

Figura 9. Funcionamiento de un LDS de 360°.

Los sensores LDS (Laser Distance Sensor) ocupan una posición relevante en el mundo de la robótica móvil. Su campo abarca desde los simples sensores de rango que miden distancia en una única dirección, hasta elementos que capturan información 3D precisa en una gran área de trabajo, tanto vertical como horizontal.

Sus principales cualidades son la buena resolución en ángulo y distancia y la elevada tasa de muestreo. La resolución en distancia tiene la característica de permanecer constante en todo el recorrido. Además de ser más rápido a la hora de obtener la información de las distancias, lo que hace que se vea menos afectado por objetos en movimiento [7].

La principal desventaja de este tipo de sensores son los altos precios que tienen en comparación con otros sensores como el sonar.

3.1.3 CÁMARAS

Una de las posibles opciones a la hora de adquisición de medidas de distancia son las cámaras. Esta medida se realiza mediante la triangulación de la medida entre dos cámaras.

También existen cámaras de profundidad como *Kinnect* la cual es capaz de medir profundidad con una sola cámara y un generador de puntos laser.

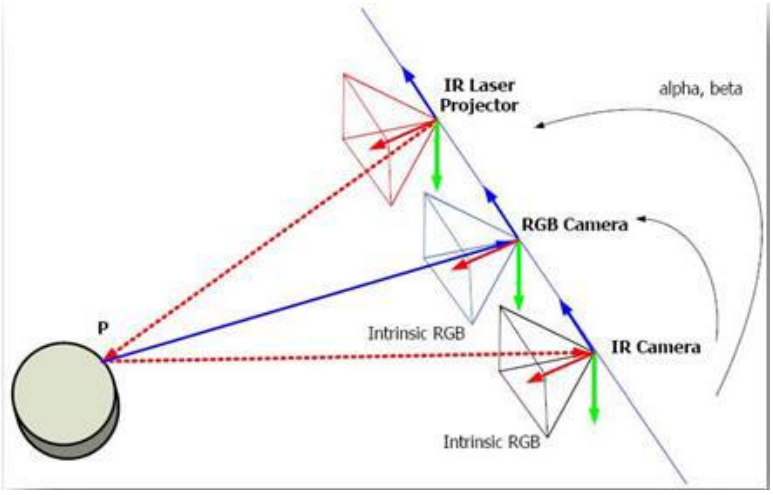

Figura 10. Esquema del funcionamiento de una cámara de profundidad.

El principal problema de las cámaras es que el procesamiento de imagen conlleva mucho cómputo. Sin embargo, las cámaras permiten realizar técnicas de SLAM en 3D, cosa que con otros sensores no es posible [6].

3.1.4 SENSORES DE ODOMETRÍA

Son populares los acelerómetros. Estos son capaces de medir la aceleración en cada uno de los ejes del sistema de referencia. Mediante la integración de estos parámetros se pueden conseguir medidas de posición.

También son bastante utilizados los giróscopos los cuales dan velocidades angulares. A partir de aquí se puede averiguar los ángulos que gira el robot [6].

Por último se encuentran los encoders rotativos, elementos que permiten determinar la posición y velocidad angular de un accionamiento, y registrar la medición desde un procesador o autómata.

3.1.5 DECISIÓN

Finalmente la decisión que se toma en cuanto a los sensores, es el LDS para XIAOMI Roborock S50 S51. Se trata de un LDS con resolución angular de 360° y capaz de tomar máximo 100 escaneos por segundo. Es la decisión, que buscando combinar fiabilidad de las medidas y un precio asequible ofrece mejores resultados, además de no engordar mucho la capacidad de cómputo a la hora de procesar los datos.

Figura 11. LDS seleccionado y ejemplo de visión en un entorno.

3.2 SELECCIÓN DE LOS ELEMENTOS ACTUADORES

Son los elementos que llevan a cabo las acciones necesarias para realizar el proceso. Ejecutan las tomas de decisión en función de la información que proporcionan los sensores. Para este proyecto estos componentes son los encargados de dotar de movimiento al robot.

3.2.1 MOTORES DE CORRIENTE CONTÍNUA

Es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio. El sentido de giro de un motor de corriente continua depende del sentido relativo de las corrientes circulantes por los devanados inductor e inducido y la velocidad de rotación, del voltaje (y la corriente) que se aplica [8].

Es por ello que para controlar su sentido de giro y su velocidad se necesitan drivers de control. Además en caso de querer conocer la odometría, se necesitarían sensores que almacenaran la información del movimiento, consumiendo capacidad del procesador y añadiendo elementos complementarios para su uso.

3.2.2 MOTORES PASO A PASO

Es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de girar una cantidad de grados (paso o fracción de paso) dependiendo de sus entradas de control.

El motor paso a paso se comporta de la misma manera que un conversor digital-analógico (D/A) y puede ser gobernado por impulsos procedentes de sistemas digitales. El número de pasos define el ángulo que girará el motor según las señales digitales de entrada, cuánto mayor sea el número de pasos más fino será el control aunque a costa de una mayor complejidad [9].

Esto los hace más difíciles de controlar a priori que los motores DC pero dan una serie de ventajas como la independencia de elementos complementarios como sensores para la odometría.

Figura 12. Motor paso a paso.

3.2.3 DECISIÓN

La elección para los actuadores es la de motores paso a paso bipolares, ya que se busca optimizar la capacidad del procesador y por sus características es la mejor opción para obtener información de la odometría y realizar el control de manera simultánea.

3.3 CONTROLADORES

Encargados de recibir la información que llega desde los sensores, procesarla y gestionarla de manera que puedan enviarles de manera eficaz las acciones a realizar a los actuadores. Se va a desestimar de primera mano el control remoto desde un PC

externo al vehículo para centrarse en las opciones de ordenadores de placa reducida que puedan incorporarse al vehículo.

3.3.1 ARDUINO

Arduino es una plataforma de hardware libre de propósito general para el desarrollo de proyectos electrónicos. El hardware consiste en una placa con un microcontrolador Atmel y puertos de entrada y salida.

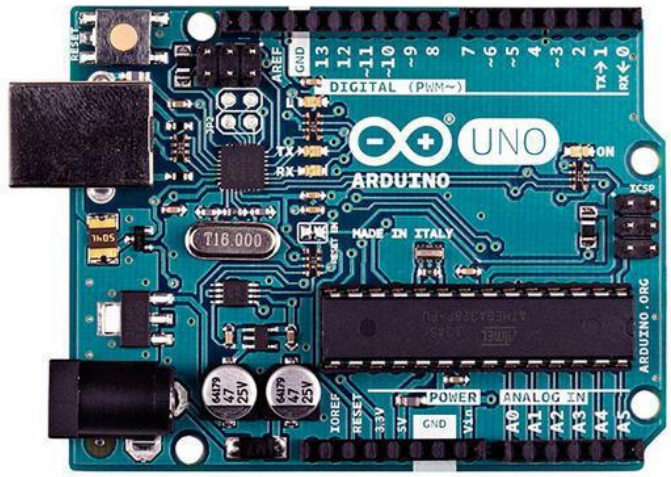

Figura 13. Placa de Arduino UNO.

Por otro lado el software, que también es libre, consiste en un entorno de desarrollo que implementa el lenguaje de programación propio y en un cargador de arranque que es ejecutado en la placa. Esta plataforma es pequeña y ligera, por lo que podría introducirse dentro de la estructura sin mucho problema.

Como ventajas de esta opción se puede destacar que al ser de hardware y software libre se reduce considerablemente el precio. Además la programación en este entorno es sencilla. Los inconvenientes principales de esta opción son la escasa capacidad de almacenamiento que posee y su velocidad de procesamiento lenta en comparación con otras opciones [10].

3.3.2 RASPBERRY PI

Raspberry Pi es un ordenador de placa reducida de bajo coste desarrollado y comercializado por la fundación Raspberry Pi. El diseño hardware contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz, un procesador gráfico (GPU) VideoCore IV, y 512 MB de memoria RAM. Utiliza una tarjeta SD para el almacenamiento permanente, que no está incluida. Esto podría parecer a priori una desventaja pero permite elegir una capacidad de almacenamiento acorde con el uso que se le va a dar, desde un mínimo de 2GB.

Figura 14. Placa de Raspberry Pi.

Tampoco incluye fuente de alimentación ni carcasa. El hardware tiene un tamaño aproximado a la plataforma Arduino. A pesar de que se trata de una plataforma comercial, su precio es competitivo, aunque superando el precio de las placas de Arduino.

Al igual que Arduino incluye pines de entrada y salida. Sin embargo la frecuencia de reloj es mucho mayor en Raspberry Pi (hasta 1 GHz, mientras que en Arduino es de unos 16 MHz). Otra ventaja de esta plataforma es la variedad de sistemas operativos que se pueden utilizar y la versatilidad de los lenguajes de programación como C++ o Python [10].

3.3.3 OTRAS OPCIONES

Además de las opciones expuestas anteriormente cabe destacar el gran número de alternativas que han surgido recientemente.

Un claro ejemplo de ello es la plataforma Nanode, desarrollada por un Ingeniero Electrónico de Reino Unido. Se trata de una plataforma muy similar a Arduino, pero con la ventaja de poderse conectar a Internet a través de un API (Interfaz de Programación de Aplicaciones). Se puede encontrar por un precio más económico y se puede programar desde cualquier sistema operativo a través del mismo entorno que Arduino.

Otro ejemplo, además de desarrollo español, es Libelium Waspmote. Se trata de un dispositivo ideado para crear redes inalámbricas de sensores con unos requerimientos bastante específicos y destinados a ser desplegados en un escenario real. El entorno de desarrollo es el mismo que el de Arduino y el código de este se puede adaptar para ser utilizado. Se trata de una plataforma interesante pero destinada a otro tipo de proyectos [10].

3.3.4 DECISIÓN

La decisión adoptada es Arduino, más concretamente la placa de Arduino Due. Dada las características del proyecto y sus necesidades es la mejor opción de control, además se podrían usar varias placas en paralelo en caso de quedarse sin capacidad de procesamiento o memoria, que es su principal desventaja respecto a Raspberry Pi.

3.4 MONTAJE

Finalmente, una vez elegidos los principales componentes del vehículo, se va a proceder a mostrar cómo deberían conectarse electrónicamente entre ellos.

3.4.1 CONEXIONES ACTUADORES CONTROLADOR

En primer lugar, se presenta la conexión del motor paso a paso con el controlador. Esta conexión se hace a través del driver el DRV8825, que simplifica el manejo de los motores paso a paso y permite manejar los altos voltajes e intensidades que requieren estos motores, limita la corriente que circula por el motor, y proporciona las protecciones para evitar que la electrónica pueda resultar dañada.

Otra ventaja que proporciona el driver es el llamado *microstepping*, una técnica que permite obtener pasos inferiores al paso nominal del motor que se va a controlar variando la corriente aplicada a cada bobina [11].

El esquema de montaje del driver con el motor y la placa de Arduino es el siguiente:

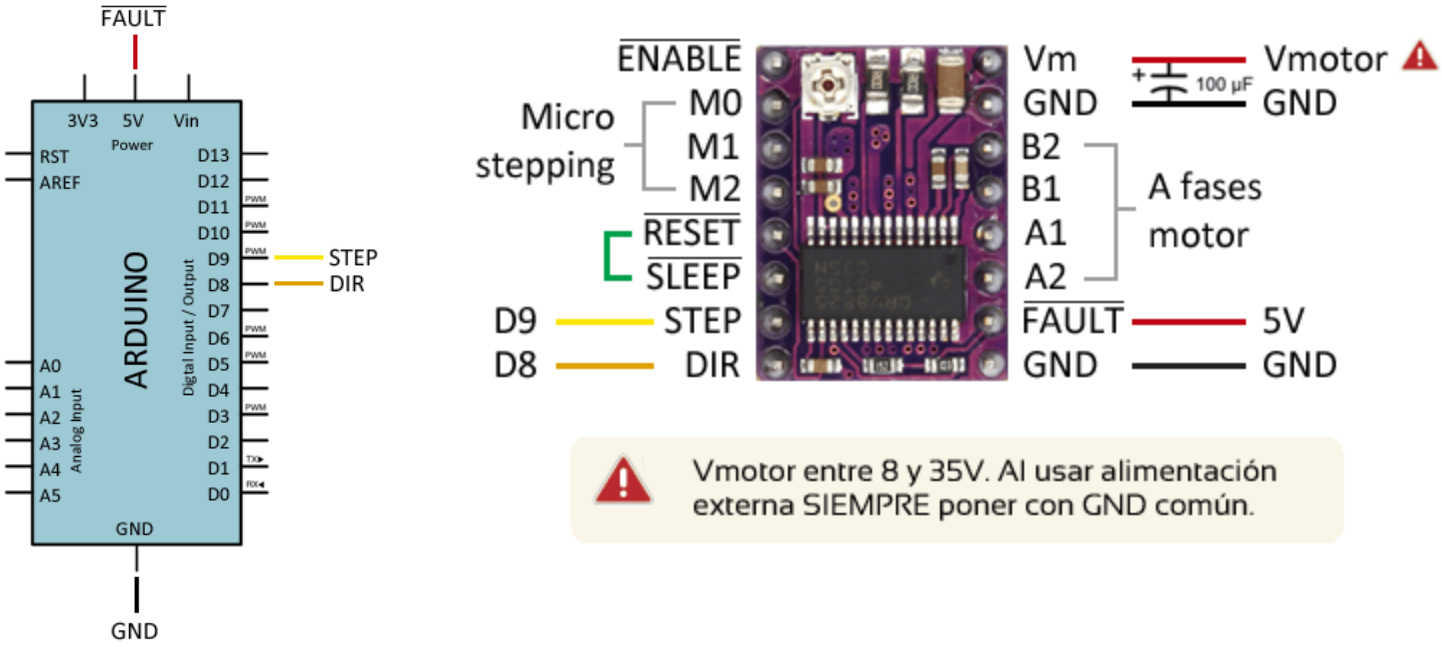

Figura 15. Conexiones driver-controlador.

3.4.2 CONEXIONES SENSOR CONTROLADOR

Una vez vistas las conexiones de los actuadores con el controlador se presentan las conexiones del sensor LDS con el controlador. Estas conexiones son muy sencillas, el LDS, que envía la información por comunicación serie, cuenta con una salida Tx, con los datos de distancias que se envían hacia el controlador, una alimentación y una tierra para su procesador interno, y dos pines para la alimentación del motor que hace girar el láser.

Las conexiones de la alimentación del motor necesitan de un diodo y un mosfet como protección del circuito [12]. De esta manera el esquema completo de las conexiones del LDS es el siguiente:

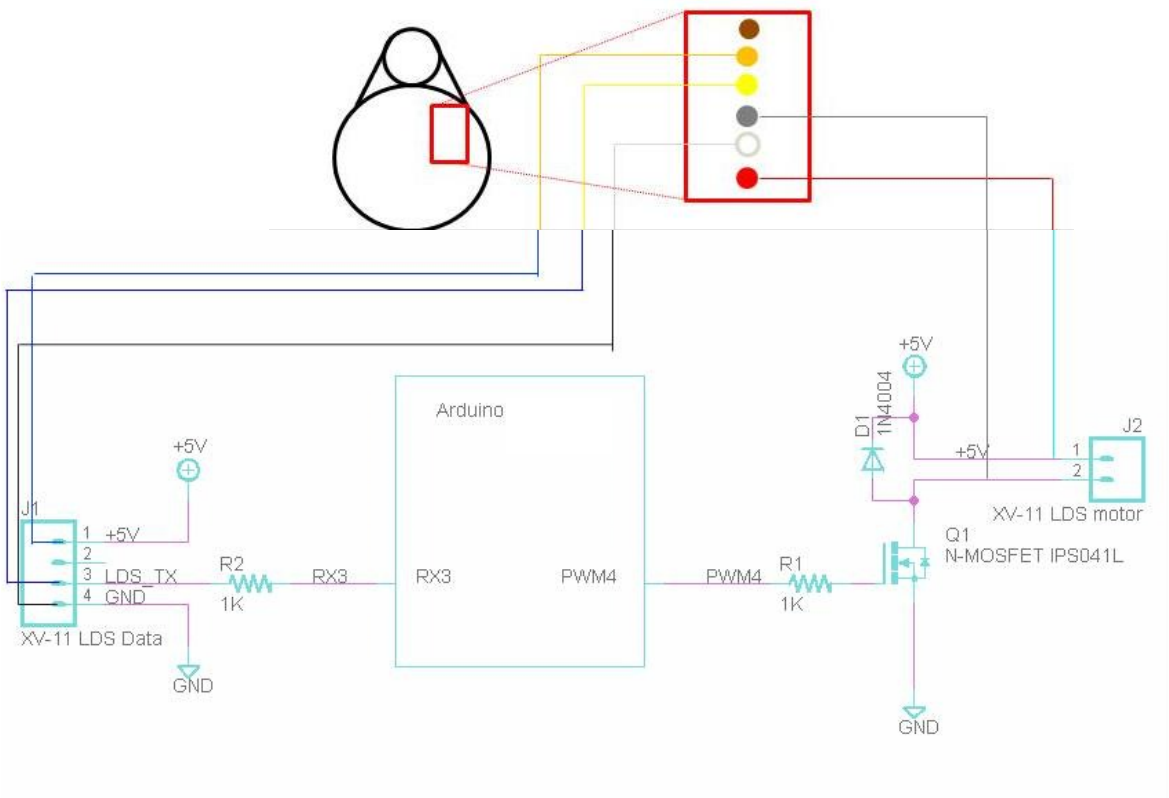

Figura 16. Conexiones LDS-controlador.

3.4.3 MONTAJE COMPLETO

Para finalizar, una vez mostrado el montaje por separado de actuadores y sensor, se muestra como quedaría el montaje completo, con los dos motores paso a paso y sus dos drivers, el LDS XIAOMI (Representado por una cabeza de cinco pines) y la placa de Arduino Due, además de los componentes del circuito necesarios para el buen funcionamiento de los componentes y la electrónica.

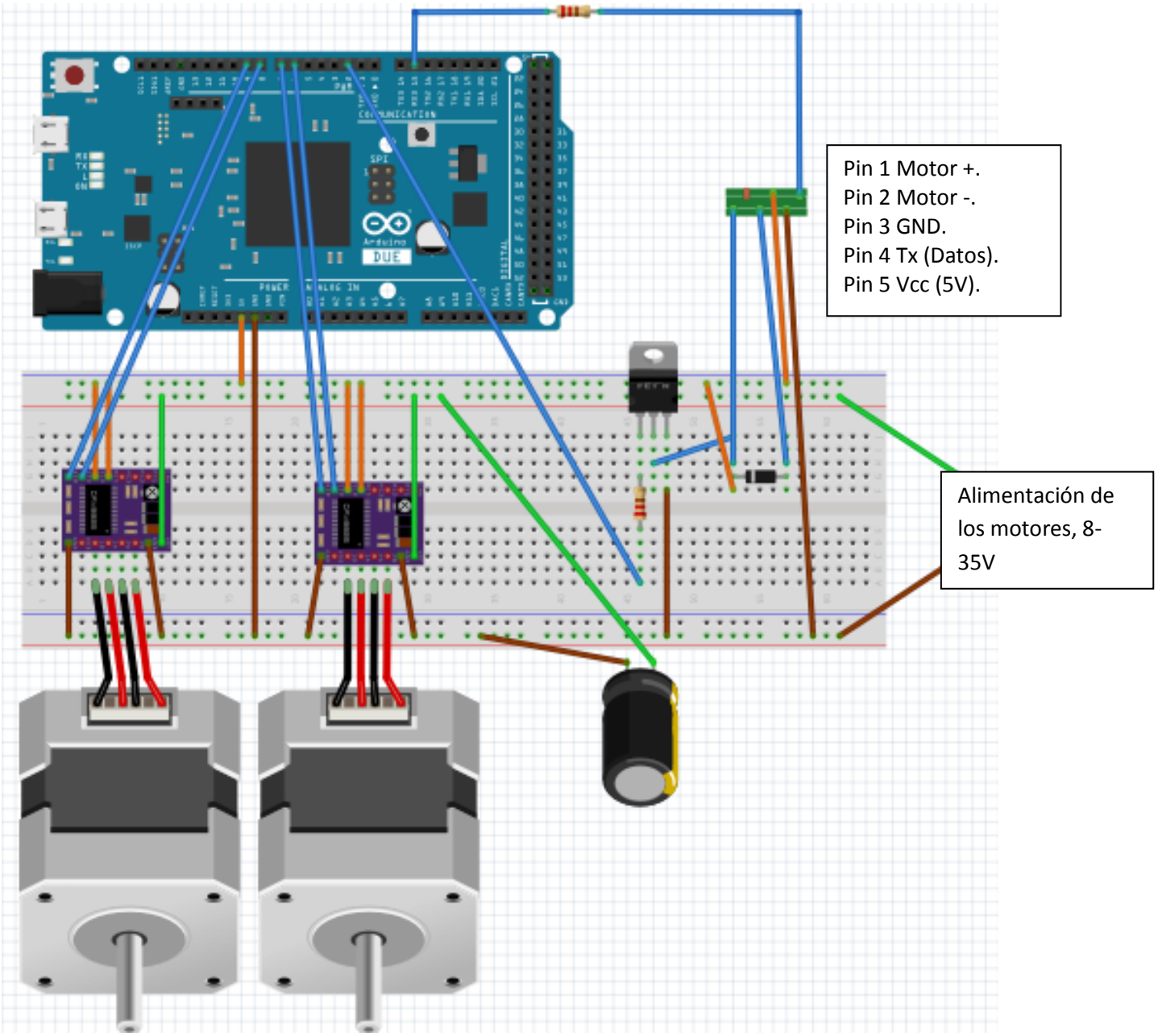

Figura 17. Conexiones de los componentes seleccionados.

4 ASPECTOS PREVIOS Y CINEMÁTICOS

En este capítulo se abordan una serie de problemas y conceptos a resolver previos a la realización del simulador y a la implementación de los algoritmos SLAM y de control.

4.1 INTERPRETAR LOS DATOS DEL SENSOR, CREAR UN MAPA SIMPLE

El sensor seleccionado es un telémetro laser, que realiza un barrido de ángulos. Esta acción da las distancias de los impactos con el objeto más cercano en cada uno de dichos ángulos. Para poder trabajar con estos datos se deben realizar una serie de operaciones.

4.1.1 CAMBIO DE COORDENADAS

Ya que se tienen un conjunto de distancias para diferentes ángulos, se puede decir que estos datos se corresponden con las coordenadas polares de los objetos respecto a la posición del robot, por lo tanto, habrá que realizar un cambio a coordenadas cartesianas donde:

$$X = \rho \cdot \cos(\theta)$$
$$Y = \rho \cdot \text{sen}(\theta)$$

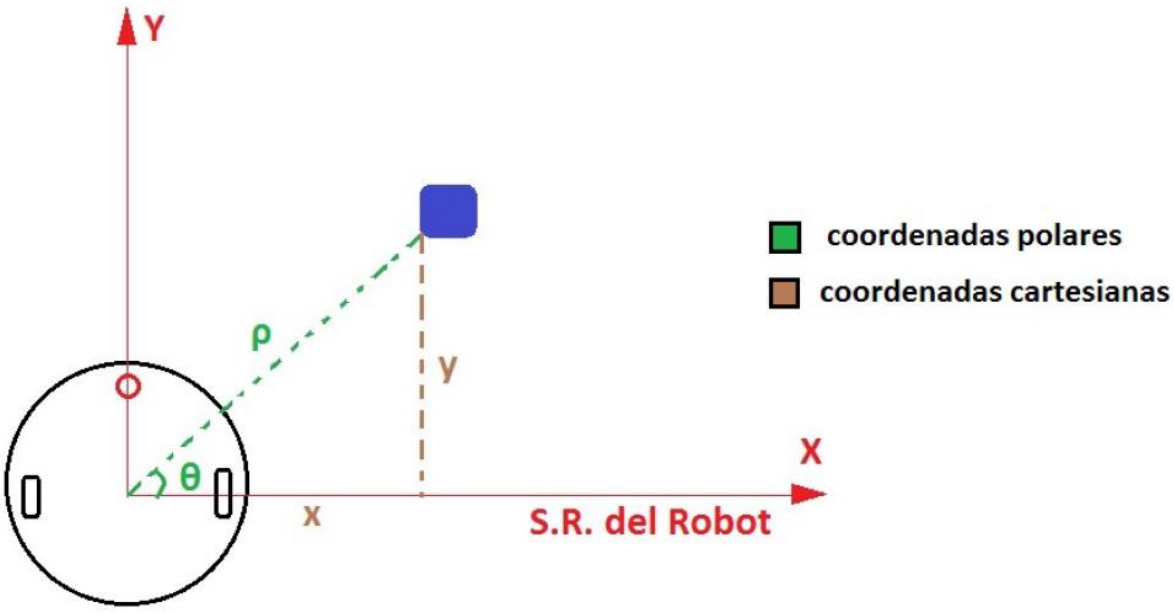

Figura 18. Esquema del sistema de referencia del robot, coordenadas polares y cartesianas.

4.1.2 CAMBIO DE SISTEMA DE REFERENCIA

Una vez obtenidas las coordenadas de manera cartesiana, se llevaran a una referencia fija, debido a que la referencia del robot está en movimiento respecto al entorno y se quiere trabajar con un mapa global. Esta nueva referencia fija, pasará a llamarse *Sistema de referencia global*

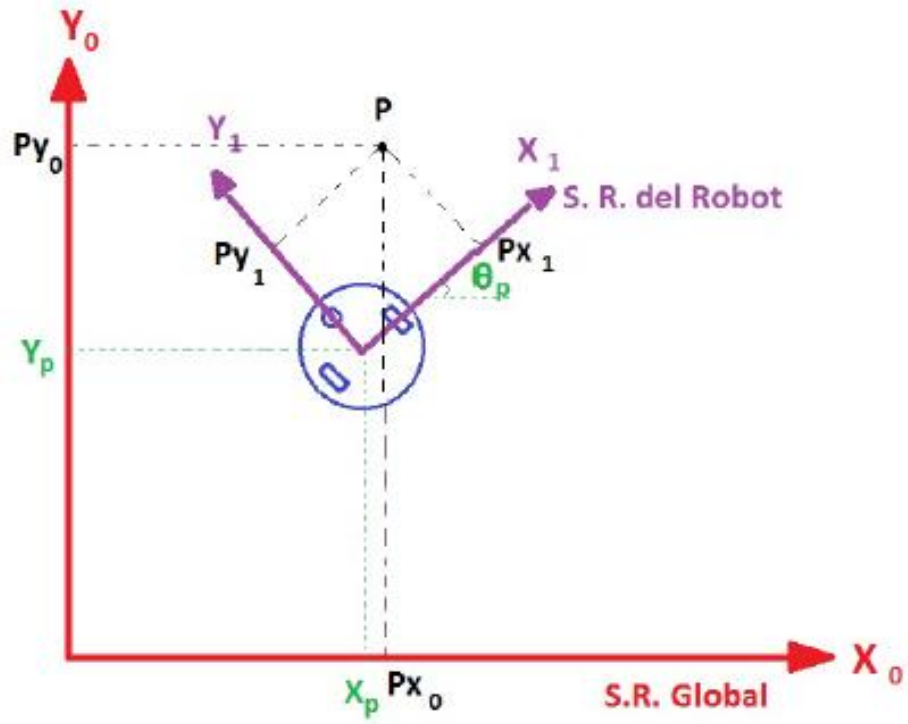

Figura 19. Esquema de los sistemas de referencia global y del robot, coordenadas cartesianas.

La posición del robot en la referencia global pasa a estar definida por sus dos coordenadas X_p e Y_p, y la orientación θ. De esta manera, conociendo la posición del robot respecto a la referencia global, se puede cambiar de la referencia del robot a la referencia global.

$$P_{Robot} = \begin{bmatrix} x_p \\ y_p \\ \theta \end{bmatrix} \qquad P_{RefRobot} = P_1 = \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \qquad P_{RefGlobal} = P_0 = \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

Se define una matriz de cambio T, tal que:

$$T * P_1 = P_0$$

$$T = \begin{bmatrix} \cos(\theta) & -sen(\theta) & x_p \\ sen(\theta) & \cos(\theta) & y_p \\ 0 & 0 & 1 \end{bmatrix}$$

4.2 CINEMÁTICA DEL VEHÍCULO

En un sistema real, la posición del vehículo no viene determinada de antemano sino que se debe calcular en función de ciertos parámetros que dependerán del sistema de localización usado. La mayoría de sistemas de localización avanzados requieren de un mapa previo del entorno para funcionar adecuadamente, pero ya que en este caso no disponemos de mapas previos, se debe usar un sistema más sencillo que dependa de parámetros fácilmente calculables. El sistema que se usará para este propósito es el de *Método de localización por odometría*, que calcula la pose en intervalos finitos de tiempo en función de la pose anterior y la velocidad del robot.

El inconveniente de este sistema es que, al ser un método recursivo, se necesitará partir de una pose conocida. Por eso, a partir de ahora, el sistema de referencia global coincidirá con el sistema de referencia robot en la posición inicial.

El modelo de robot que se estudia se conoce como Modelo de Locomoción Diferencial. Se dispone de dos motores bidireccionales ubicados en el eje central del robot y a ambos lados del centro, siendo simétrica la ubicación de las ruedas respecto al centro del robot.

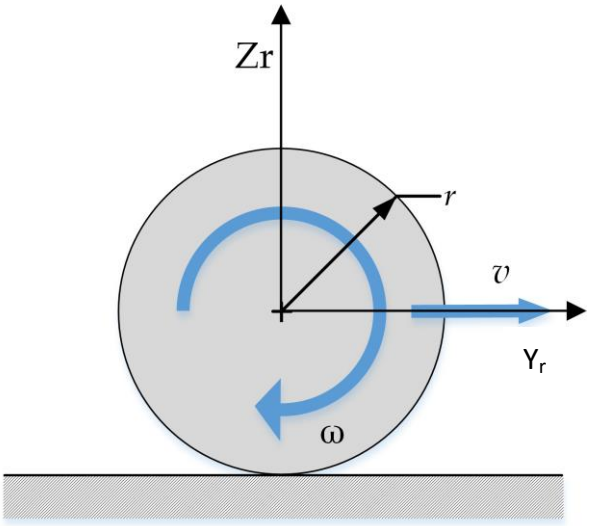

Figura 20. Velocidades angular y lineal de una rueda del robot.

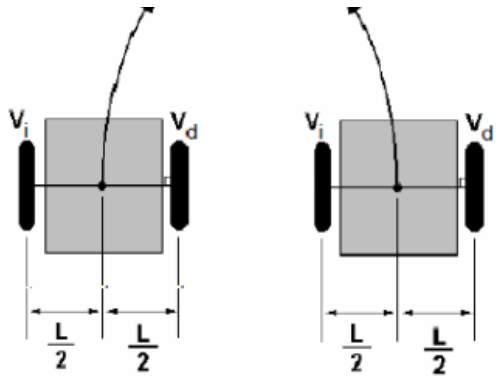

Figura 21. Velocidades y distancias en el eje de centro a ruedas.

Las fórmulas para el modelo cinemático directo son:

$$v_d = w_d \times r$$

$$v_i = w_i \times r$$

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

$$w = \frac{v_d - v_i}{L}$$

$$\theta_{k+1} = \theta_k + w \times \Delta t$$

$$x_{k+1} = x_k + v \times \cos(\theta_{k+1} + \frac{\pi}{2}) \times \Delta t$$

$$y_{k+1} = y_k + v \times \cos(\theta_{k+1} + \frac{\pi}{2}) \times \Delta t$$

Donde:

v_d : Velocidad lineal de la rueda derecha [m/s]
v_i : Velocidad lineal de la rueda izquierda [m/s]
w_d: Velocidad angular de la rueda derecha [rad/s]
w_i: Velocidad angular de la rueda izquierda [rad/s]
r: Radio de las ruedas [m]
v: Velocidad lineal del robot [m/s]
w: Velocidad angular del robot [rad/s]
L: Distancia entre las ruedas [m]
Δt: Incremento de tiempo [s]
θ_k: Componente angular de la pose, instante anterior [rad]
θ_{k+1}: Componente angular de la pose, instante posterior [rad]
x_k: Componente X de la pose, instante anterior [m]
x_{k+1}: Componente X de la pose, instante posterior [m]
y_k: Componente Y de la pose, instante anterior [m]
y_{k+1}: Componente Y de la pose, instante posterior [m]

5 PROGRAMACIÓN DEL SIMULADOR

Con el fin de facilitar la implementación de los algoritmos de control, se realiza un simulador del vehículo y del entorno en Processing. Este punto permite verificar el comportamiento de los algoritmos ante un entorno determinado, sin tener que recurrir al sistema físico.

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

El programa está diseñado para facilitar los aspectos visuales de las distintas variables que se están manejando. Este aspecto es muy adecuado para el simulador ya que se podrán comprobar los distintos algoritmos y pruebas que se realicen de manera gráfica directamente y no sobre resultados numéricos únicamente.

5.1 SIMULACIÓN DEL LDS

En primer lugar se programa la simulación del LDS, es decir la información que se recibiría desde un entorno determinado si se colocara el vehículo en un punto arbitrario del mismo y se activara el sensor elegido previamente.

Para ello se crea el entorno en el cual se realiza la simulación, que viene dado por un conjunto de segmentos que dan lugar a un espacio cerrado, simulando un conjunto de habitaciones. Posteriormente se simula el vehículo, que vendrá representado por un círculo y sus dos ejes X (azul) e Y (rojo), y se posiciona dentro del entorno creado previamente.

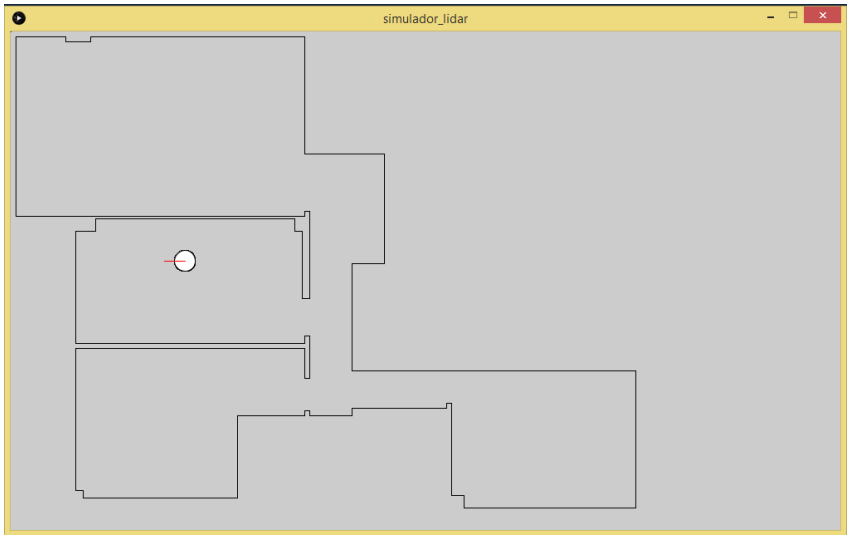

Figura 22. Simulación del entorno y el robot en Processing.

Una vez se tiene tanto el entorno como el robot, se realiza la simulación del funcionamiento del LDS. Para ello se calcula la intersección de las 360 rectas que simulan el barrido laser del sensor seleccionado, desde el centro del robot en la posición que lo hayamos colocados y con un incremento de un grado entre recta y recta, con los segmentos que componen el entorno. El simulador se queda tan solo con la primera intersección que encuentre, es decir con los puntos de intersección más cercanos al robot para cada recta.

De esta manera, se obtiene finalmente una imagen del entorno real desde el vehículo y el alcance del sensor sobre el mismo. Y por otra parte el conjunto de puntos que vería el vehículo y que formarían el mapa del entorno que podría realizar. Los datos de los puntos que ve el LDS son extraídos de la forma ángulo y radio, tal y como vienen dados en el sensor real.

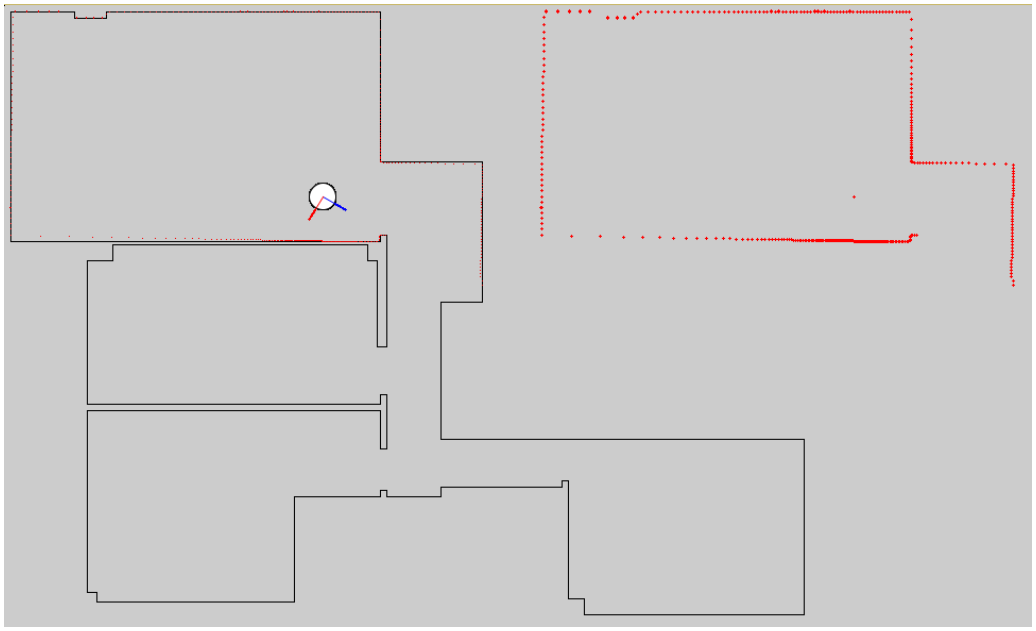

Figura 23. Simulación del funcionamiento del LDS.

A parte de la simulación del LDS, se simula el movimiento del robot programando las ecuaciones de la cinemática del capítulo previo, usando como incremento de tiempo para calcular la posición, el equivalente para la frecuencia de toma de datos del LDS correspondiente. De manera que la posición varía en función de las velocidades de las ruedas derecha e izquierda del robot.

Tal y como se tiene programado el simulador, las distintas variables que se obtienen, se tienen de manera ideal, es decir, tanto los datos del LDS como los datos de la posición en la que se encuentra el robot se corresponden exactamente a la realidad que estamos simulando.

Como se ha comentado anteriormente en la realidad física no ocurre esto y el robot que se está simulando no es ideal, el LDS funciona con cierto error o ruido en las medidas que recoge y la información de la odometría que obtenemos de los actuadores no se corresponde con lo que ocurre en el sistema físico con exactitud. Por ello y con el fin de realizar el simulador lo más cercano a la realidad posible se va a introducir un factor de ruido a las distintas variables.

En primer lugar se alteraran los radios de las medidas del LDS de manera que se almacenarán directamente para su tratamiento las medidas alteradas. De esta forma el radio de los puntos que ve el LDS será:

$$\rho = \rho ideal * (1 + \{-1,1\} * \text{Factor de ruido})$$

Por último se altera la posición que se obtiene de las ecuaciones de la cinemática que corresponderían con el caso de odometría ideal. Aunque para esta variable, se tienen que guardar tanto la posición ideal como la real ya que ambas son necesarias a la hora de continuar con el proyecto.

La posición alterada se usará como posición real (simulada) en la que se encuentra el robot y por tanto posición de la que extraeremos los datos del LDS en la simulación, y la posición ideal será la posición que el robot calcula por odometría. De esta manera y a partir de la posición ideal obtenida previamente en el simulador, la posición real será:

$$\text{Xreal} = \text{Xideal} * (1 + \{-1,1\} * \text{Factor de ruido})$$

$$\text{Yreal} = \text{Yideal} * (1 + \{-1,1\} * \text{Factor de ruido})$$

$$\theta\text{real} = \theta\text{ideal} * (1 + \{-1,1\} * \text{Factor de ruido})$$

Siendo {-1,1} un número aleatorio comprendido entre -1 y 1.

A partir de aquí se continuará usando el simulador para el tratamiento de datos y en cada ocasión que se quiera implementar un nuevo algoritmo al diseño del vehículo, como por ejemplo los cambios de referencia, la selección de trayectorias, etc.

6 IMPLEMENTACIÓN DE ALGORITMOS SLAM

6.1 INTRODUCCIÓN

La idea básica del SLAM consiste en partir de la suposición de que la posición inicial es conocida y construir el mapa a partir de ahí. En función del tipo de observaciones que se realicen, se trabajará de una forma u otra. Si no se observa nada, actuará como en el caso de la odometría. Si se realiza una observación nueva, se añadirá al mapa. Y si se realiza una observación que se encuentra en el mapa, se utilizará para corregir tanto la posición como el propio mapa.

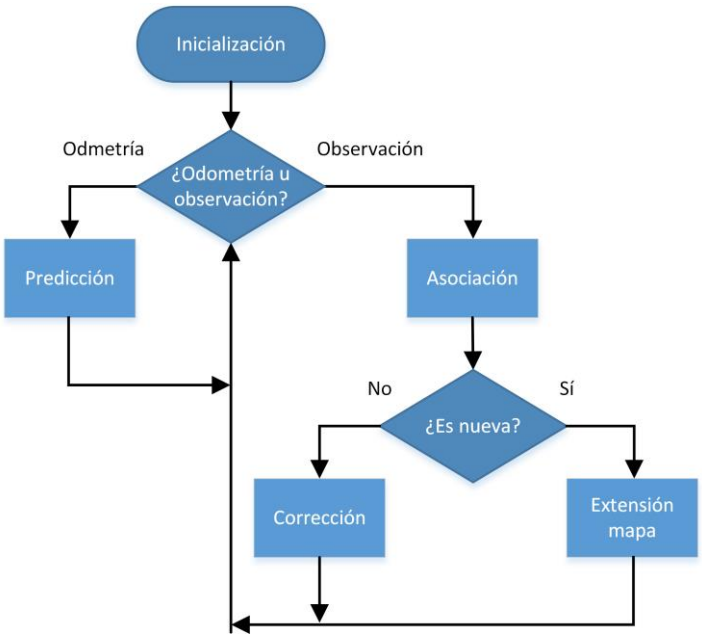

Figura 24. Esquema del algoritmo SLAM.

6.2 EXTRACCIÓN DE CARACTERÍSTICAS *(LANDMARKS)*

Una característica es un elemento del entorno que nos ayuda a posicionarnos. Antes de explicar el proceso en si hay que determinar que propiedades deberá de cumplir una buena característica [13]:

- **Invariante en el tiempo**: si en un momento determinado se ve en una posición, más adelante se debería poder ver en el mismo sitio.

- **Invariante al observador**: no debería importar desde dónde se observara. Si se encuentra en nuestro campo de visión, se debería poder reconocer.

- **Fácil de identificar**: todas las veces que la observemos deberíamos de poder identificarla sin confundirla con cualquier otra característica.

De las tres condiciones, la tercera es la más crítica. Confundir una característica por otra puede resultar desastroso en el proceso de localización. Como se usarán para corregir la posición del robot y del mapa, se estaría usando información errónea, empeorando la estimación en vez de mejorarla.

A partir de los datos que proporciona el LDS y teniendo en cuenta los entornos en los que se moverá el robot, mayoritariamente pasillos y habitaciones, el tipo de características que vamos a extraer son segmentos que se corresponden a las paredes de la habitación [13].

Una vez identificados los tipos de características que se van a emplear, tenemos que seleccionar los algoritmos que permitan extraerlas. Existen gran cantidad de ellos y para el caso que nos ocupa en el que los datos sensoriales provienen de un LDS, se utilizará, en primer lugar, un proceso de división de puntos por zonas (clustering), tras el cual se aplicará el algoritmo *iterative end point fit,* de segmentación.

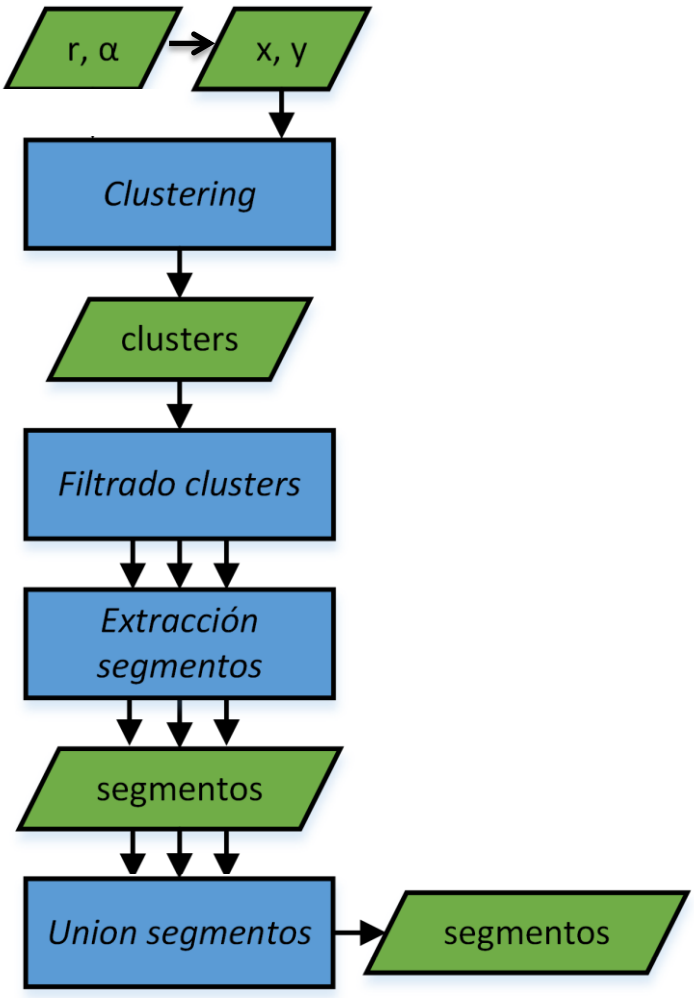

Figura 25. Esquema del algoritmo de extracción de características.

6.2.1 DIVISIÓN POR ZONAS (CLUSTERING)

Aprovechando que los datos del LDS se generan en orden, este algoritmo es usado para diferenciar por zonas la nube de puntos que llega desde el LDS. Para ello, se calcula la distancia que hay entre dos puntos consecutivos de los datos que llegan y si esta distancia supera un umbral determinado pasamos a encontrarnos en un nuevo cluster o una nueva zona [14].

De este modo, el tratamiento de la información se llevará a cabo por separado para las distintas zonas, obteniendo una división de segmentos más eficaz ya que zonas distintas del mapa separadas por pasillos o espacios grandes no tendrán correlación entre ellas en el mapa generado.

Se puede observar el funcionamiento del algoritmo y su eficacia al programarlo en el simulador y ejecutarlo en distintas posiciones del mapa. En las *figuras 23 y 24* se ven las distintas zonas formadas en la comprobación y se remarcan en rojo los puntos en los que empieza cada zona y en azul en los que acaban.

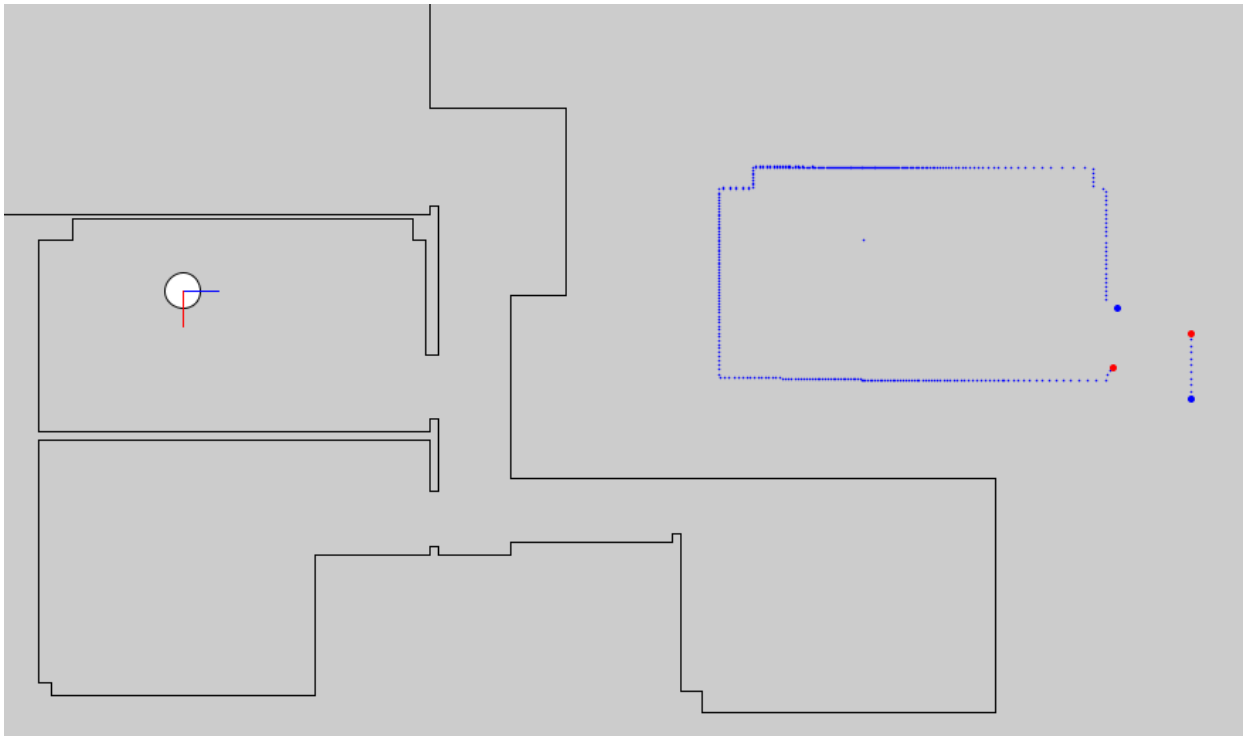

Figura 26. Simulación del algoritmo de Clustering en la habitación 2.

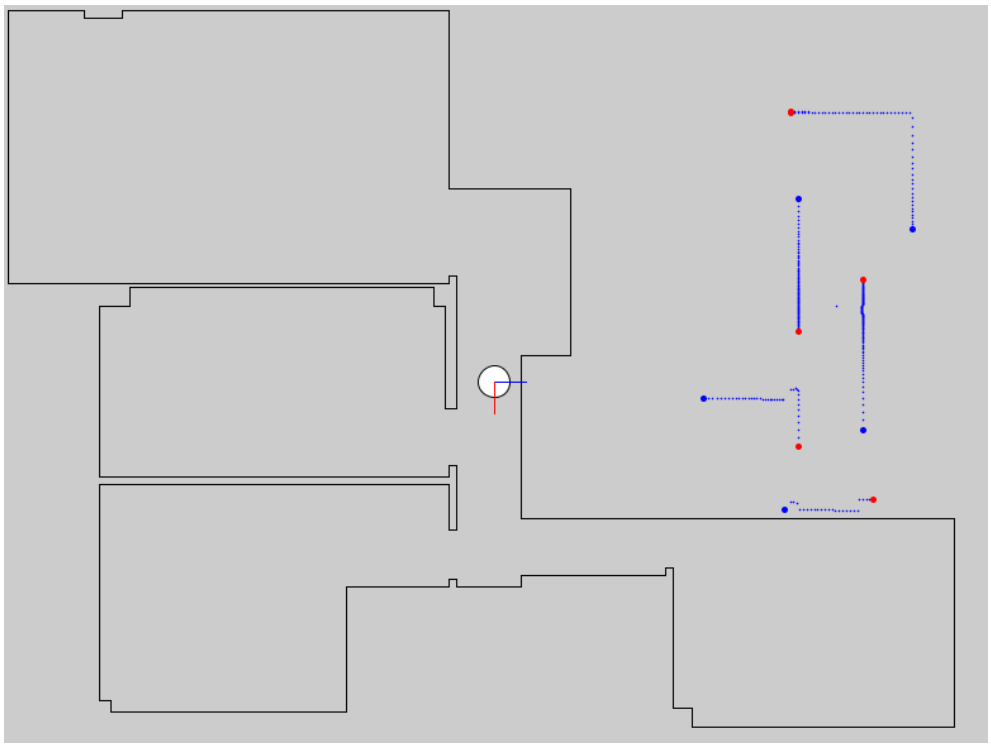

Figura 27. Simulación del algoritmo de Clustering en el pasillo.

Una vez completado el proceso es necesario realizar un tratamiento de los resultados para poder trabajar con ellos de una manera eficaz. Este tratamiento consistirá en eliminar zonas que vengan dadas de manera consecutiva o que no cuenten con un número de puntos suficiente en su conjunto, ya que no representan información valiosa a la hora de segmentarlos y almacenarlos como parte del mapa.

6.2.2 ITERATIVE END POINT FIT

A cada zona de la etapa anterior se le aplica el algoritmo *iterative end point fit*. Un algoritmo que dado un conjunto de puntos ordenado extrae líneas de forma iterativa. El funcionamiento de este algoritmo es el siguiente [15]:

1. En primer lugar se define una línea tomando los puntos inicial (P_0) y final (P_n) de la zona.

2. A continuación se busca el punto más alejado (P_j) con respecto a dicha línea. Si el punto encontrado está a una distancia mayor que un determinado valor umbral, entonces se divide la zona inicial en dos intervalos.

3. Después de esto el algoritmo comienza recursivamente con cada uno de los segmentos restantes; P_0 P_j y P_j P_n.

4. El procedimiento termina una vez que la distancia al punto más alejado sea menor que el umbral fijado, siempre y cuando esta circunstancia se cumpla para todos y cada uno de los segmentos en los que ha sido dividido la zona.

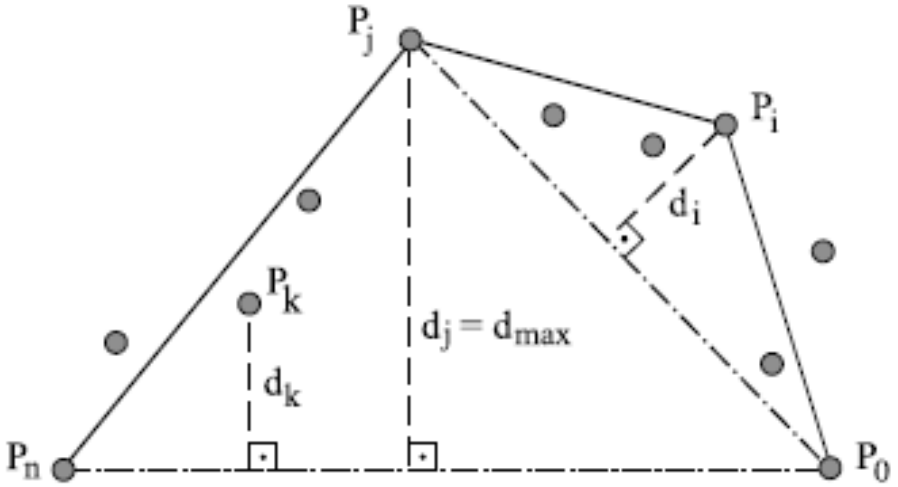

Figura 28. Esquema de funcionamiento del algoritmo *Iterative end point fit.*

Tal y como se ha programado el algoritmo en el simulador, éste devuelve los índices de los puntos más alejados mencionados anteriormente de manera que se pueda seguir aprovechando que los datos del LDS vienen ordenados.

Se tiene por tanto un vector con los índices de los puntos del *Iterative end point fit* (El cual se ordena por valor para que sea más fácil su tratamiento) y otro con los índices de los puntos en los que empieza cada zona (Que vienen directamente ordenados).

Con esos índices se tienen que construir los segmentos que componen el mapa y que de momento vendrán dados por sus puntos inicial y final en forma de índices.

Dentro de cada zona, el punto que da inicio al mismo es el inicio del primer segmento y el punto en el que finaliza la zona es el final del último segmento. De esta manera los segmentos serán consecutivos dentro de cada zona, es decir el final de un segmento coincidirá con el inicio del siguiente, pero existirá separación entre las distintas zonas, diferenciando las zonas mencionadas en la explicación del algoritmo anterior.

Se pueden observar los resultados de este proceso de segmentación tras implementarlo en el simulador, las imágenes mostradas coinciden en posición con las del apartado de clustering para observar el correcto funcionamiento de manera secuencial del proceso completo.

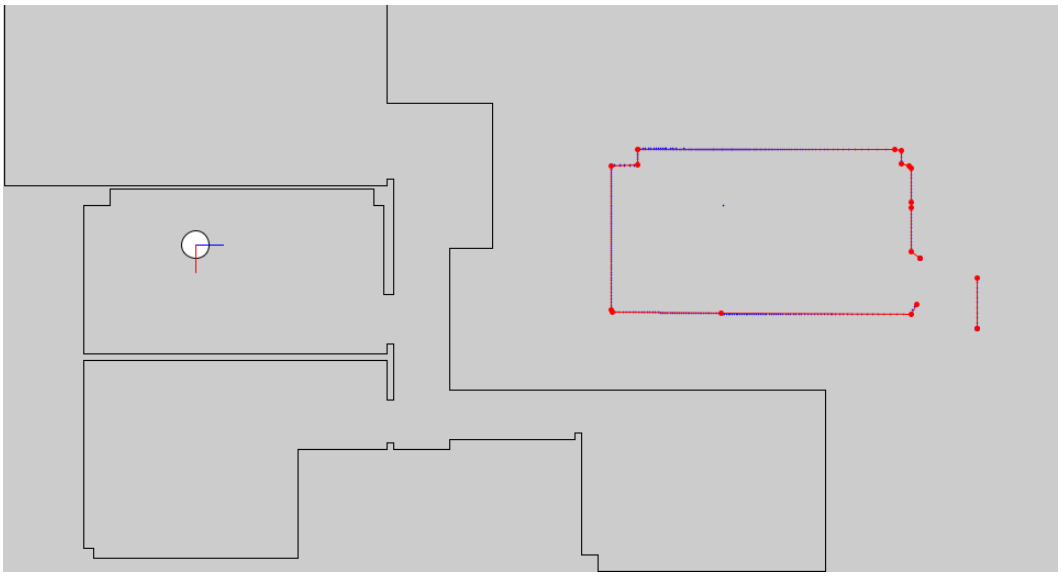

Figura 29. Simulación del algoritmo *Iterative end point fit* en la habitación 2.

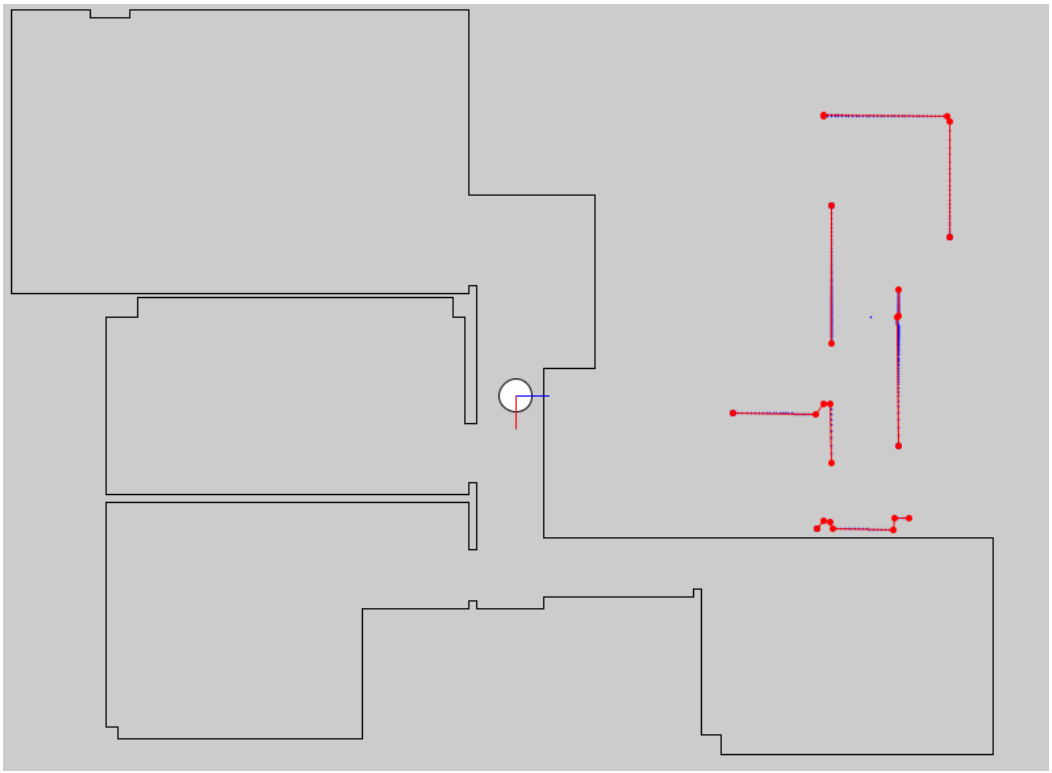

Figura 30. Simulación del algoritmo *Iterative end point fit* en el pasillo.

Se puede observar que el algoritmo funciona correctamente, aunque para optimizar las siguientes fases del proceso SLAM y la capacidad de memoria y procesamiento, los segmentos serán tratados de manera que aquellos que tengan un tamaño inferior a un mínimo no serán contados, y los que sean colineales y cercanos serán unidos y tratados como un solo segmento.

6.3 ASOCIACIÓN DE DATOS (EXTENSION DEL MAPA)

Uno de los momentos clave en el proceso SLAM una vez extraídas las características es la asociación de datos. En esta asociación se tiene que discernir como tratar la información que llega a cada toma de datos del LDS respecto a la información del mapa que se va almacenando, ya que un error en este apartado puede resultar en consecuencias fatales para el algoritmo.

6.3.1 REPRESENTACIÓN DE LOS SEGMENTOS

En primer lugar y con el fin de facilitar la gestión de la información se va a cambiar la representación de los segmentos extraídos. Tal y como llegan del algoritmo de segmentación, vienen representados por los índices de los puntos de sus extremos, estos puntos se llevaran a la referencia global de manera que cada segmento estará definido por X_1, Y_1, X_2, Y_2.

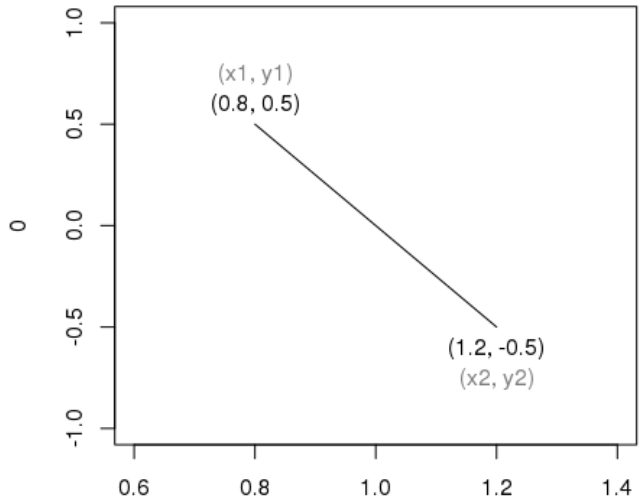

Figura 31. Representación de un segmento por sus extremos en coordenadas cartesianas.

Con el fin de poder comparar los segmentos antiguos que conforman el mapa con la nueva información que viene del LDS se va a calcular la ecuación de la recta que pasa por sus extremos $y = x * m + a$, y de sus parámetros m y a se obtendrán sus parámetros polares θ y ρ.

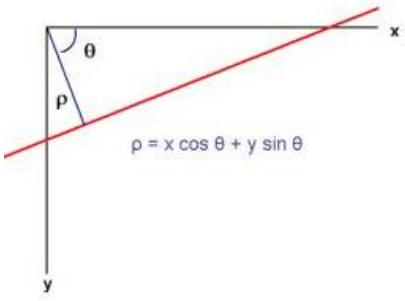

Figura 32. Representación de una recta en coordenadas polares.

Estos parámetros permitirán tratar la recta como si fuera un punto, lo cual gestionará mejor el método de localización. A parte de estos dos nuevos parámetros las rectas seguirán estando definidas por sus coordenadas de inicio y final, las cuales se usarán en el proceso de comparación a la hora de ampliar el mapa. Así pues, los segmentos quedan definidos como $[\theta, \rho, X_1, Y_1, X_2, Y_2]$.

6.3.2 COMPARACIÓN DE SEGMENTOS A LA HORA DE AMPLIAR EL MAPA

Una vez se obtienen los segmentos de la manera adecuada y en la referencia global, se pasan a compararlos con los segmentos antiguos y que forman el mapa.

La comparación se hace entre cada uno de los segmentos extraídos de la nueva observación con todos los segmentos que conforman el mapa. En primer lugar se comprueba si el nuevo segmento observado coincide dentro de una tolerancia con alguno de los segmentos antiguos. Para ello se comparan sus parámetros θ y ρ.

Si no coincide con ningún segmento de los que conformaban el mapa, se añade a esta lista de segmentos y pasa a completar el mapa.

En caso de que coincidan, se pasa a comparar los extremos de los segmentos, teniendo dos opciones en función de cómo se relacionen estos:

- Si los segmentos son coincidentes, es decir, alguno de los extremos de un segmento se encuentra contenido en el otro, se pasa a modificar el segmento antiguo de manera que nos quedaremos con los dos extremos que nos den la mayor longitud de segmento.

  De esta manera conforme avance el tiempo los segmentos nunca se acortaran, solo podrán alargarse ajustándose al mapa real. Suponiendo que el robot se encuentra en todo momento en ambientes estáticos

- Si los segmentos no son coincidentes, es decir, ningún extremo está contenido en el otro segmento, se añadirá el segmento nuevo al mapa.

Para comprobar el correcto funcionamiento de la asociación de datos, se introduce el algoritmo en el simulador y se le hace pasar por una serie de posiciones que lo llevan desde la segunda a la primera habitación, tras lo cual se observa el mapa final dado por el conjunto de segmentos que lo conforma al final del proceso.

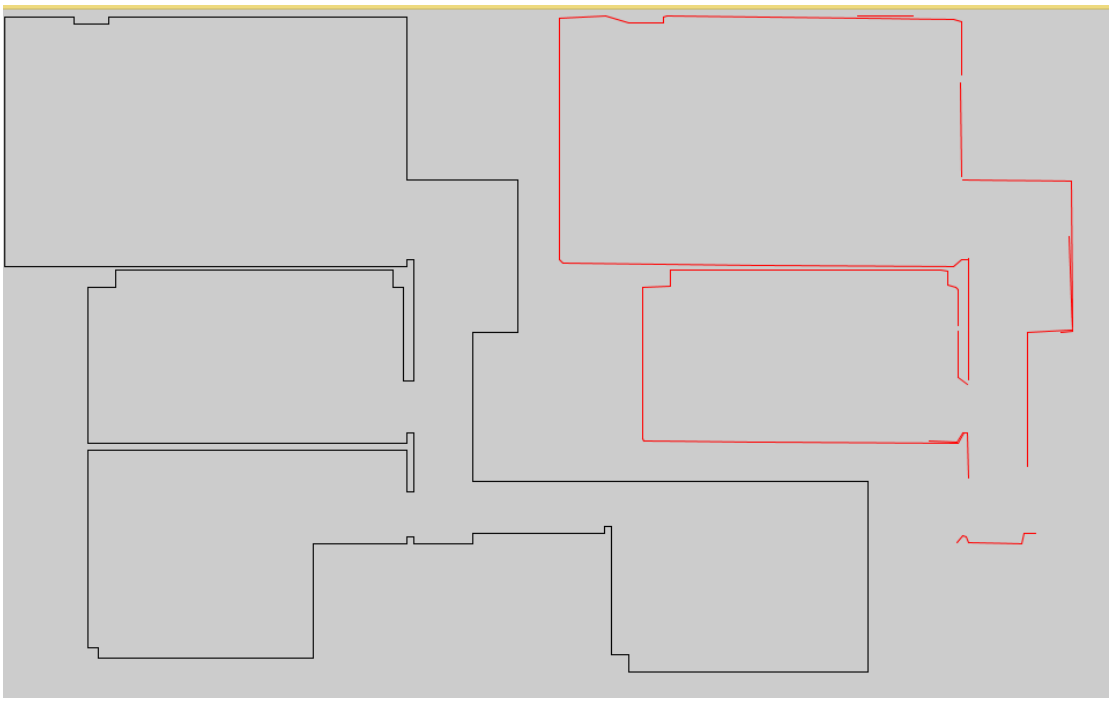

Figura 33. Simulación de la reconstrucción del mapa durante el trayecto de la habitación dos a la uno.

6.4 CORRECIÓN DE LA POSICIÓN USANDO EL MAPA

Por último para completar de manera eficaz el proceso SLAM, es necesario corregir la posición en la que se localiza el robot en el mapa en función de lo que se ve, por las razones que se han expuesto durante todo el trabajo.

Debido a la compleja implementación y a la capacidad de procesamiento y memoria que consume el Filtro extendido de Kalman, principal recurso para corregir la posición que existe en el mundo de la robótica móvil, se busca una solución alternativa que dé solución a este problema con un mínimo de eficacia pero reduciendo el volumen de cálculo necesario.

En lugar de dar pesos a la estimación por la odometría y a la visual como ocurre en el Filtro extendido de Kalman, en la solución al problema que se propone, vamos a fiarnos de los datos visuales, usando la odometría para complementar el proceso.

Según se ha diseñado hasta ahora el vehículo, se cuenta con un mapa de segmentos en una referencia global que es lo que representa el entorno, y que se usará para comparar con los puntos que vienen en cada momento del LDS y estimar la posición en la que se encuentra el robot.

El proceso comienza estimando, desde la última posición en la que se encontraba el robot y mediante la odometría, donde se debería encontrar en el mapa en el momento actual (momento de la última recogida de datos del LDS) esta posición será la posición

ideal comentada en el punto 5, la cual es diferente a la posición real desde la que se recogen los datos del LDS. A partir de esta estimación y aprovechando los algoritmos de simulación, se calculan que puntos debería ver el robot en el momento actual según el mapa que se tiene.

Los puntos calculados en la estimación, vienen representados por las coordenadas X, Y, mientras que los puntos reales procedentes del LDS vienen representados por sus coordenadas xr, yr.

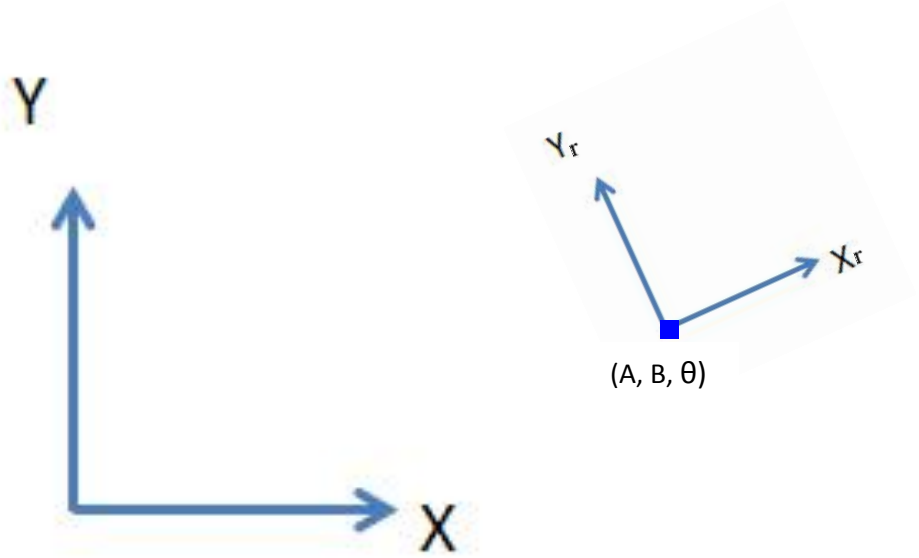

Figura 34. Sistemas de referencia global y robot (en la posición A, B, θ)

De manera que:

$$\begin{pmatrix} x_r \\ y_r \end{pmatrix} = \begin{pmatrix} (X-A)*\cos(\theta) - (Y-B)*sen(\theta) \\ (Y-B)*\cos(\theta) - (X-A)*sen(\theta) \end{pmatrix}$$

Siendo la distancia entre un punto de la estimación y de la realidad:

$$D = [(X-A)*\cos(\theta) - (Y-B)*sen(\theta) - x_r]^2 \\ +[(Y-B)*\cos(\theta) - (X-A)*sen(\theta) - y_r]^2$$

Si la estimación y la realidad se correspondieran, la distancia entre los puntos debería ser la mínima. Se calculan pues según la filosofía de los mínimos cuadrados los parámetros A, B, θ (parámetros que dan la posición corregida). Esto quiere decir, que la derivada del sumatorio de las distancias en función de A de B y de θ tiene que ser cero para que la distancia sea mínima.

$$\Sigma\frac{\partial D}{\partial A} = -2 * \Sigma X + 2 * n * A + 4 * \cos(\theta) * sen(\theta) * (\Sigma Y - n * B) + 2 * \cos(\theta) * \Sigma x_r + 2 * sen(\theta) * \Sigma y_r$$

$$\Sigma\frac{\partial D}{\partial B} = -2 * \Sigma Y + 2 * n * B + 4 * \cos(\theta) * sen(\theta) * (\Sigma X - n * A) + 2 * \text{sen}(\theta) * \Sigma x_r + 2 * cos(\theta) * \Sigma y_r$$

$$\begin{aligned}\Sigma\frac{\partial D}{\partial \theta} = {} & 4 * sen(\theta)^2 * \Sigma(X * Y) - 4 * sen(\theta)^2 * B * \Sigma X - 4 * sen(\theta)^2 * A \\ & * \Sigma Y + 4 * n * A * B * sen(\theta)^2 - 4 * \cos(\theta)^2 * \Sigma(X * Y) + 4 \\ & * \cos(\theta)^2 * B * \Sigma X + 4 * \cos(\theta)^2 * A * \Sigma Y - 4 * n * A * B \\ & * \cos(\theta)^2 - 2 * sen(\theta) * \Sigma(x_m * X) + 2 * sen(\theta) * A * \Sigma x_m \\ & - 2 * sen(\theta) * \Sigma(y_m * Y) - 2 * sen(\theta) * B * \Sigma y_m - 2 \\ & * cos(\theta) * \Sigma(x_m * Y) + 2 * cos(\theta) * B * \Sigma x_m - 2 * cos(\theta) \\ & * \Sigma(y_m * X) - 2 * xoa(\theta) * A * \Sigma y_m\end{aligned}$$

De esta igualdad con las derivadas de los sumatorios quedan finalmente tres ecuaciones con tres incógnitas. Al tratarse de un sistema de ecuaciones no lineales, se recurrirá al método iterativo de Newton-Raphson modificado para sistemas de ecuaciones no lineales para resolver el sistema.

El punto de partida para la iteración que se introducirá en el método será la posición ideal calculada por la odometría en la que se tendría que encontrar, ya que se trata de la solución conocida que es más cercana a la solución real.

Una vez el proceso ha acabado y tenemos los parámetros A, B, θ que cumplen con la condición de mínimos cuadrados, estos pasan a ser la posición corregida del robot y ya se está listo para comenzar el proceso SLAM de nuevo.

7 CALCULO DE LA TRAYECTORIA

Por último se va a dotar al vehículo con un cálculo para la trayectoria a realizar sin colisionar, dado el destino, el cual vienen de manera incremental respecto a la posición inicial en la que se encuentre, y sin conocimiento previo del mapa, tan solo la información que en cada momento recibe del LDS.

Para ello, en primer lugar se calcula la recta que une al robot con el destino, de manera que se obtendrá la dirección de la trayectoria principal en la cual se moverá el robot, e irá buscando constantemente.

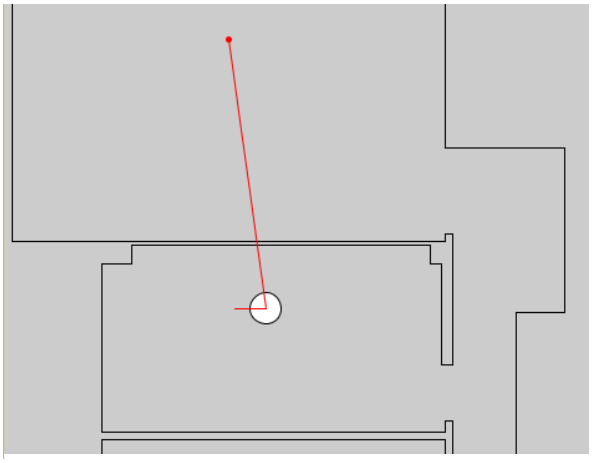

Figura 35. Recta que une el robot con el destino y dirección principal.

A partir de aquí se puede encontrar con dos situaciones según si la distancia al obstáculo más cercano en la dirección de la trayectoria es mayor o menor a una distancia mínima. En caso de no encontrarse con ningún obstáculo, el robot seguirá la trayectoria principal hasta el destino, cuando se encuentre con algún obstáculo el robot pasará a evitar el obstáculo.

Para evitar el obstáculo, en primer lugar, en cuanto es detectado (distancia en la dirección de la trayectoria menor a la distancia mínima) se orientará el robot de manera que la dirección de avance quede paralela al obstáculo. Para poder calcular los ángulos que debe girar se mira en que índice de los puntos del LDS se encuentra la distancia más corta, para saber la orientación del robot respecto del obstáculo.

Tras ello el robot bordeará el obstáculo siguiendo un avance recto respecto a la orientación previa. Este avance acabará cuando la distancia lateral, supere un umbral, o si la distancia en la dirección de avance vuelve a ser menor que la distancia mínima, en cuyo caso, el robot volverá a girarse hasta quedar paralelo al nuevo obstáculo y repetirá el proceso de bordearlo.

Una vez que tanto la distancia lateral como la distancia en la dirección de avance al obstáculo más cercano son mayores que la distancia umbral y la distancia mínima respectivamente, el robot se separará del obstáculo que haya estado bordeando. Para

ello avanzará en la dirección que llevaba una distancia de seguridad, girará 90 grados en la dirección contraria a la que haya hecho el último giro para bordear el obstáculo, y volverá a avanzar una distancia de seguridad. Cuando finalmente termina el proceso de evitar obstáculo con el robot separado las distancias de seguridad, se recalcula la recta que une el robot con el destino propuesto y se empieza el proceso de nuevo hasta la llegada a su destino.

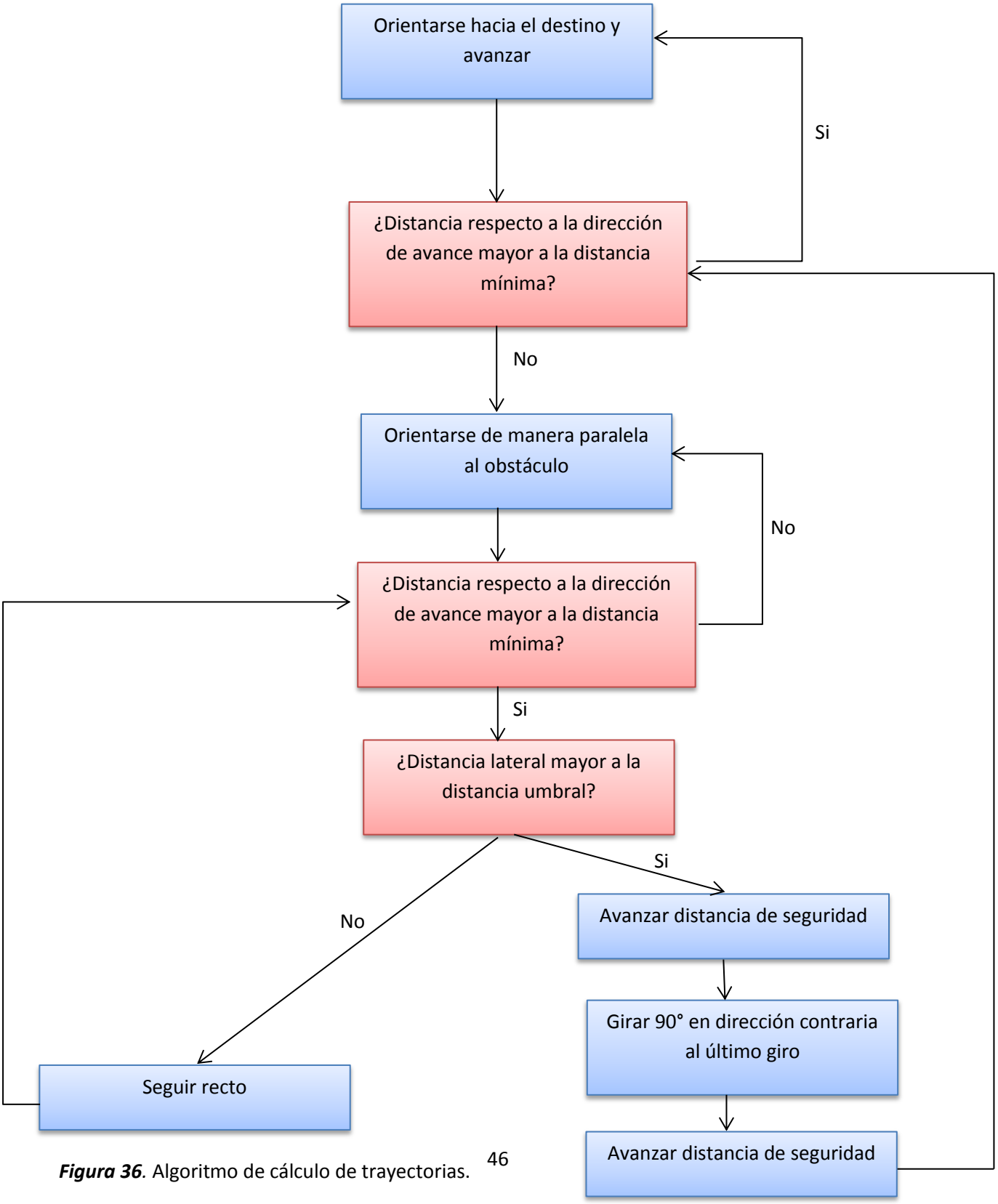

Figura 36. Algoritmo de cálculo de trayectorias.

Se implementa finalmente el algoritmo de cálculo de trayectorias para el robot en el simulador de processing de manera que se pueda comprobar la eficacia del mismo.

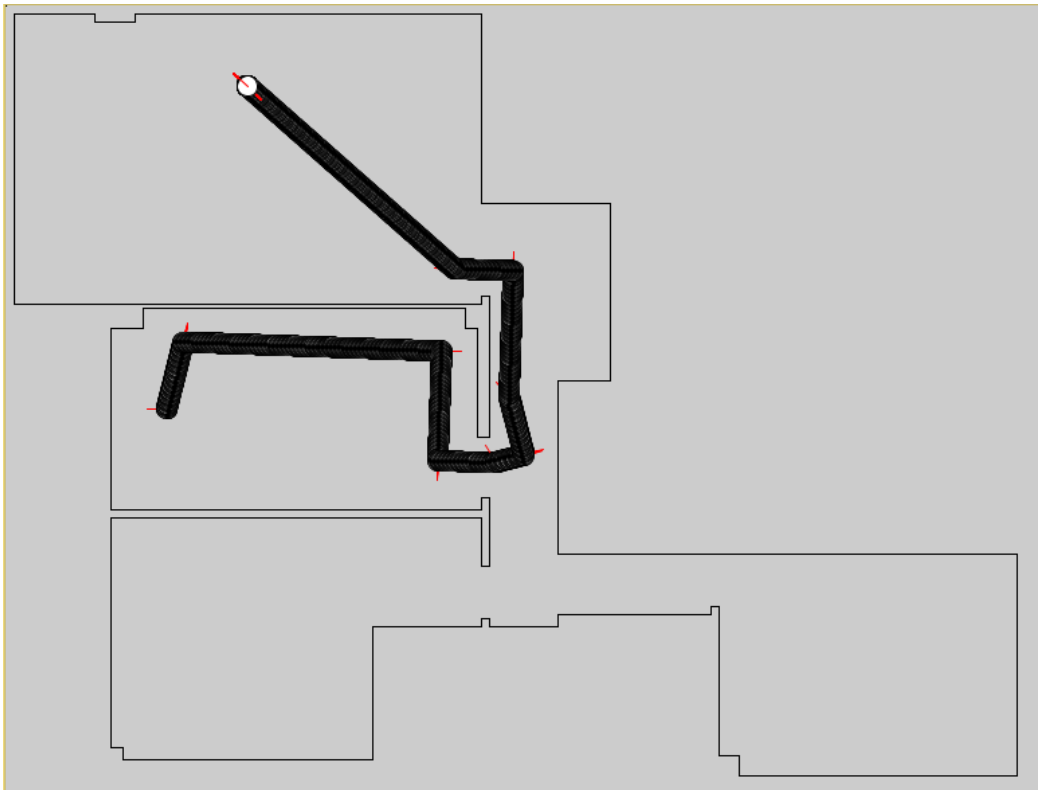

Figura 37. Simulación del algoritmo de cálculo de trayectorias desde habitación dos a habitación uno.

Se puede observar que el comportamiento no es del todo eficaz ya que cuando no está en la habitación en la que se encuentra el destino, antes de salir, bordea la pared más cercana al destino, la cual detecta como un obstáculo y pasa a evitarlo. Incluso entra en otras habitaciones una vez ha salido de la que se encontraba inicialmente ya que son las que se encuentra en la trayectoria principal.

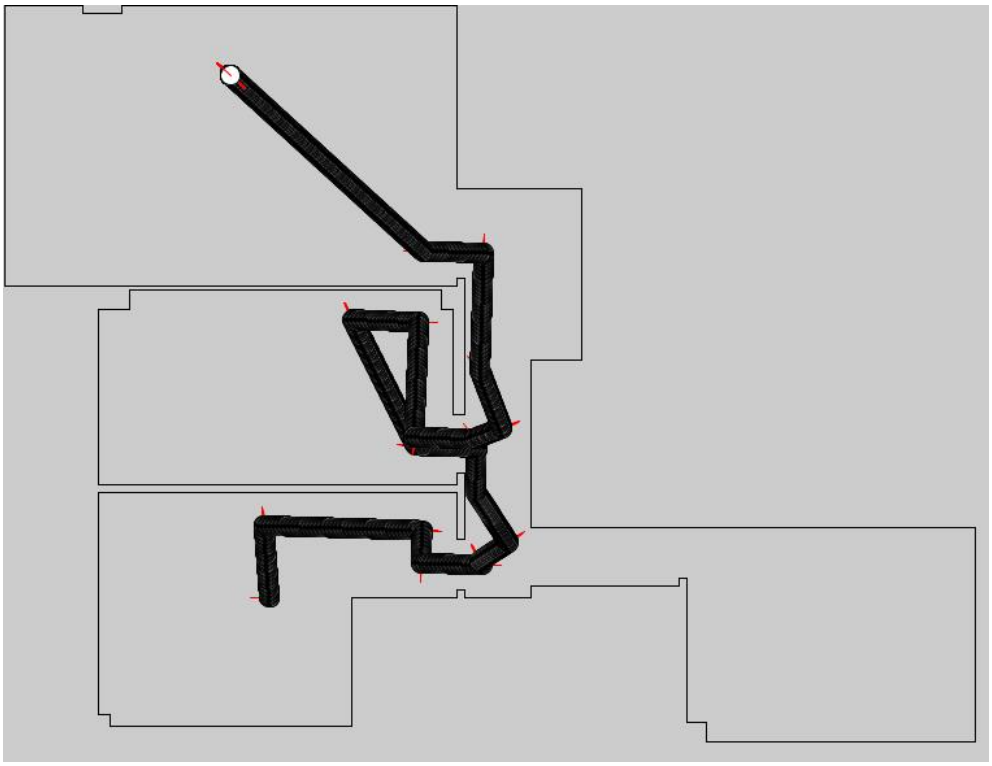

Figura 38. Simulación del algoritmo de cálculo de trayectorias desde habitación tres a habitación uno.

Para evitar estos comportamientos ineficaces en la trayectoria, lo ideal sería tener un conocimiento previo del mapa completo antes de planificar la trayectoria. De todas formas, usando el mapa que se va construyendo conforme se avanza de la manera que se ha expuesto en los apartados previos, podríamos ser capaces de detectar con anterioridad si el destino se encuentra en la habitación en la que estamos o en la que acabamos de acceder, además de detectar los huecos o puertas por los que salir o avanzar.

De esta manera se podrían planificar trayectorias mucho más eficientes para llegar a destinos concretos. Por ejemplo yendo directamente a los huecos en el mapa que representan puertas cuando se detecte que el destino no está en la habitación a la que se ha entrado, dotando al vehículo de un comportamiento más “humano”.

Estos algoritmos de selección de trayectorias y los distintos algoritmos SLAM explicados en el punto 6, se encuentran en lenguaje de Processing (Java). Por ello lo único que queda es cambiarlos al lenguaje de programación de Arduino para concluir el proyecto. Para el código de Arduino se pueden ahorrar todas las funciones de representación gráfica de los resultados, así como operaciones intermedias que se hayan necesitado para el buen funcionamiento del simulador. Quedándonos solo con los algoritmos y rutinas necesarias para el funcionamiento del robot en la realidad.

8 CONCLUSIONES Y LÍNEAS FUTURAS

En primer lugar, recalcar que como se ha podido comprobar durante el desarrollo del proyecto, la robótica móvil no es una disciplina trivial y está llena de complicaciones. Algunas de ellas eran conocidas al comienzo del proyecto, pero otras han ido surgiendo durante su desarrollo y se han tenido que resolver para poder seguir avanzando hasta completar los objetivos.

Se puede concluir que los objetivos del proyecto se han cumplido.

- Se ha realizado un simulador del robot que agiliza notablemente la verificación del funcionamiento de los distintos algoritmos planteados.

- Se ha planteado una opción para la resolución de la problemática asociada al SLAM sobre un sistema con limitaciones de cálculo y memoria. A pesar de ser una solución que dista de ser genérica y adecuada a todo tipo de entornos, ha servido para la toma de contacto con la problemática comentada en la introducción. La solución adoptada presenta como característica principal el hecho de haber sido realizada desde niveles bajos de programación, sin recurrir a rutinas y funciones de biblioteca.

- Se ha planteado una solución para el problema de navegación autónoma

Como posibles líneas de trabajo futuro se pueden plantear:

- La realización del sistema físico, el montaje de los distintos componentes seleccionados y la carga del algoritmo sobre la placa del controlador. Este paso permitiría, además de verificar la eficacia del sistema diseñado, comprobar la adecuación del simulador a las condiciones del sistema real.

- La implementación y comprobación de soluciones basadas en distintos algoritmos.

9 BIBLIOGRAFÍA

[1] Aníbal, B, (2001), *ROBÓTICA, Manipuladores y Robots Móviles,* Barcelona (España), Marcombo.

[2] Localización y modelado simultáneos. (s.f.). En Wikipedia. Recuperado el 16 de Abril de 2018 de https://es.wikipedia.org/wiki/Localizaci%C3%B3n_y_modelado_simult%C3%A1neos

[3] Brian Carvajal Meza (2015). *Sistemas de localización y construcción de mapas*. Universidad Politécnica de Cataluña, Barcelona.

[4] Bruno Siciliano and Oussama Khatib, editors. Springer Handbook of Robotics. Springer, 2008.

[5] Localización y modelado simultáneos. (s.f.). En Wikipedia. Recuperado el 22 de Junio de 2018 de https://es.wikipedia.org/wiki/Localizaci%C3%B3n_y_modelado_simult%C3%A1neos

[6] Alejandro González González-Regueral (2016). *Algoritmo de localización y mapeo simultáneo con un robot móvil utilizando ROS y Gazebo*. Universidad de Sevilla, Sevilla.

[7] T.J. Chong, X.J. Tang, C.H. Leng, M. Yogeswaran, O.E. Ng, Y.Z. Chong. (2015). *Sensor Technologies and Simultaneous Localization and Mapping (SLAM).* IEEE International Symposium on Robotics and Intelligent Sensors.

[8] Geekbot Electronics (Abril 2015). Motores DC-Geekbot Electronics. Recuperado de http://www.geekbotelectronics.com/motores-de-dc/

[9] Motores paso a paso. (s.f). En Wikipedia. Recuperado en Junio de 2018 de https://es.wikipedia.org/wiki/Motor_paso_a_paso

[10] Cristina Blanco Abia (2014). *Desarrollo de un sistema de navegación para un robot móvil.* Universidad de Valladolid, Valladolid.

[11] Luis Llamas (Agosto 2016). MOTORES PASO A PASO CON ARDUINO Y DRIVER A4988 O DRV8825. Recuperado de https://www.luisllamas.es/motores-paso-paso-arduino-driver-a4988-drv8825/

[12] Anton D. (16 de septiembre de 2017). XIAOMI Vacuum LDS Sensor PinOut and Arduino connection. [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=b5hEG3b_Kec

[13] Søren Riisgaard, Morten Rufus Blas. (2005). *SLAM for Dummies A Tutorial Approach to Simultaneous Localization and Mapping By the 'dummies'.* Instituto Tecnológico de Massachusetts.

[14] Julie Stephany Berrío, Lina María Paz, Eduardo Caicedo Bravo (2012). *Segmentación y parametrización de líneas en datos láser 2D basado en agrupamiento por desplazamiento de media.* Universidad del Valle.

[15] Algoritmo de Ramer–Douglas–Peucker. (s.f.). En Wikipedia. Recuperado en 10 de Junio de 2018 de https://es.wikipedia.org/wiki/Algoritmo_de_Ramer%E2%80%93Douglas%E2%80%93Peucker.

10 LISTADO DE FIGURAS