UTN.BA FACULTAD REGIONAL BUENOS AIRES

TESIS DE MAESTRÍA

Maestría en Ingeniería de

Sistemas de Información

Título:

"Modelo Híbrido De Planeamiento De Caminos Para La Navegación Autónoma En Ambientes Desconocidos E Invariantes En El Tiempo"

Autor: Ing. Ignacio Javier Bonelli
Director de tesis: Dr. Ing. Alejandro Hossian

Buenos Aires, 14 de Agosto de 2022

(versión revisada 26 de Noviembre de 2023)

Agradecimientos

En primer lugar voy a agradecer a mi familia, tanto mi esposa como mis hijos me han dado mucho apoyo. En especial a mi esposa que no solo me ha apoyado, si no que leyó el “manuscrito” del presente trabajo. Luego dentro de la Facultad mi director de tesis fue una gran ayuda para poder lograr concretar este trabajo. Finalmente, también quiero agradecerle a la directora de la carrera que me ha ayudado a no perderme por el camino.

A todos, ¡muchísimas gracias!

RESUMEN

El planeamiento de caminos para robots es una técnica importante para cualquier tipo de navegación autónoma. Esta técnica es utilizada para todo tipo de robots en áreas tan dispares como los almacenes o la exploración espacial.

El lograr que un robot encuentre su camino a un objetivo prefijado en distintos ambientes es un desafío en sí mismo. En este trabajo se creó una nueva combinación de algoritmos que puedan encontrar el camino a un objetivo o en su defecto agotar las alternativas hasta demostrar que no es posible encontrarlo.

El mundo a recorrer no se conoce de antemano y es invariante en el tiempo. Esto hace necesaria la recolección de información a medida que se avanza en la navegación del ambiente. También es muy importante elegir una estrategia reactiva que permita navegar el mundo eficientemente sin conocerlo.

La solución propuesta utiliza una arquitectura híbrida que tiene tres algoritmos en la capa reactiva: APF, uno que sigue caminos y A*. Los tres algoritmos se articulan en la capa deliberativa para poder llegar al objetivo.

En los resultados se compara la navegación lograda con otra que conozca el mundo de antemano. Dependiendo de la configuración del mundo, los resultados son bastante similares.

Palabras Clave: Navegación Autónoma, Planeamiento de Caminos Reactivo y Deliberativo, Navegación Híbrida

ABSTRACT

Path planning for robots is an important technique for any kind of autonomous navigation. This technique is used for all kinds of robots in areas as different as warehouses or space exploration.

For a robot to find his path to an objective in different environments is a challenge in itself. This work presents a novel combination of algorithms that can find a path to his objective, or exhaust the alternatives until it can declare that there is no possible path.

The world is not known and will not change over time. This makes data recollection an important task to do, as the robot progresses in his path. It also makes important the choosing of the right reactive algorithm which allows the robot to navigate the world efficiently without knowing it.

The proposed solution uses an hybrid architecture with three reactive algorithms: APF, one that follows a path and A*. These three algorithms work together with the deliberative module in order to accomplish the navigation goal.

In the result section of this work there is a comparison between the proposed navigation and one that can plan a path by having full knowledge of the world before navigating it. Depending on the world configuration, the results are similar.

Keywords: Robot Navigation, Path Planning Reactive Deliberative, Hybrid Navigation

Índice General

Índice de Figuras

Índice de tablas

Glosario

Término	Definición
Algoritmos	Un algoritmo es una secuencia de pasos finitos bien definidos que resuelven un problema en particular. Dependiendo del dominio del problema a resolver existen distintos algoritmos que permiten resolverlo y se estudian aisladamente para ver cual aplica mejor en cada circunstancia.
Navegación	Dentro del presente trabajo se entiende por navegación al recorrido que realiza el agente o robot para lograr desde un punto inicial llegar al objetivo deseado.
Navegación completa	En el contexto de este trabajo se denomina navegación o solución completa al trabajo de encontrar un camino posible al objetivo o declarar el objetivo como inalcanzable (Sección 1.2).
Sistema Reactivo	Es un sistema que solo responde a estímulos externos en función de su estado interno. No tiene memoria.
Sistema Deliberativo	Los sistemas deliberativos toman decisiones en base a su historia, estrategias y la información recolectada. Planifican y evalúan toda la información antes de tomar una decisión.
Sistema Híbrido	Cuando un sistema tiene comportamientos reactivos y deliberativos que articula para trabajar en conjunto, se dice que el sistema es híbrido.
APF	El acrónimo en inglés de Campos Potenciales Artificiales. Se explica en la subsección 2.2.3 y usa como base de la capa reactiva.
Astar	El acrónimo en inglés de A Estrella. Es un algoritmo de búsqueda exhaustiva básico, pero muy utilizado (Hart et al., 1968).
FollowPath	Algoritmo de navegación diseñado en este trabajo con el fin de seguir un camino prefijado por la capa deliberativa.

1. INTRODUCCIÓN

En este capítulo se presentan los **objetivos de la tesis** (sección 1.1), el contexto actual en que el estudio se desarrolla en **visión general de la tesis** (sección 1.2), luego se ven detalles del campo particular de la **navegación autónoma** (sección 1.3), la **metodología de desarrollo de la tesis** (sección 1.4), **publicaciones científicas vinculadas a esta tesis** (sección 1.5) y finalmente la **estructura general de la tesis** (sección 1.6).

1.1 Objetivo De La Tesis

En este trabajo se explora y presenta una combinación novedosa de algoritmos que le permite a un robot navegar un mundo invariante en el tiempo sin conocerlo.

Los objetivos de este trabajo de maestría se dividen en un objetivo general a alcanzar (subsección 1.1.1) y un conjunto de objetivos específicos que definen los pasos a seguir para alcanzar el objetivo general (subsección 1.1.2)

1.1.1. Objetivos Generales

El objetivo general del presente trabajo es crear un algoritmo novedoso que logre navegar de forma autónoma cualquier tipo de ambientes desconocidos e invariantes en el tiempo. Si no existe un camino, se debe declarar el ambiente sin solución.

Para poder navegar, el algoritmo deberá encontrar un camino entre un punto de origen y uno de destino, evitando durante el camino los obstáculos que encuentre. Debe encontrar el camino al destino definido desde el origen planteado. Si no puede alcanzarlo, debe declarar que no hay camino posible. En ambos casos el algoritmo habrá encontrado la solución al problema.

1.2.2. Objetivos Específicos

Dentro del objetivo general planteado se busca lograr los siguientes objetivos específicos:

1. Presentar el área de conocimiento, su contexto y particularidades.
2. Encontrar algoritmos candidatos para formar una arquitectura híbrida.
3. Identificar distintos tipos de ambientes que permitan validar la arquitectura.
4. Ensayar distintas combinaciones de algoritmos para lograr el objetivo general.
5. Comparar los resultados de la arquitectura planteada con otros algoritmos.

Los objetivos han sido numerados para poder hacer referencia a cada uno de ellos más tarde en este mismo documento.

1.2 Visión General De La Tesis

Uno de los primeros estudios sobre algoritmos para navegación autónoma de robots fue realizado por W. G. Walter (1950). En la figura 1.1 se puede ver cómo eran estas “tortugas”.

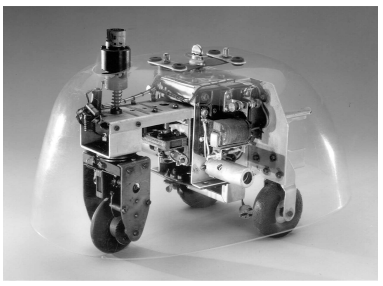

Fig. 1.1 Tortuga Walter

Fig. 1.2 Roomba iRobot

El área ha avanzado mucho luego de eso y el campo ha dejado de ser teórico (Shitsukane y col. 2018). Actualmente, la navegación autónoma consiste en una tecnología utilizada diariamente en distintas situaciones: Aspiradoras (como las Roomba de iRobot que podemos ver en la figura 1.2), Robots para rescates como el DRC-HUBO+ (ver figura 1.3) ganador del DARPA Robotics Challenge (Jung y col. 2018), Ayuda en el hogar como el Spot

Mini 2 (ver figura 1.5) de Boston Dynamics (Ackerman 2018 para la revista Spectrum del IEEE), Manejo de depósitos de Ocado (ver figura 1.6 y artículo «Estos robots te arman el pedido del supermercado en 5 minutos», 2017), manejo autónomo de autos (Casner et al., 2016) y finalmente Robots para la exploración de otros cuerpos celestes como el Mars Curiosity Rover (figura 1.4) que es un laboratorio del tamaño de una camioneta 4x4 que está explorando Marte (Guzewich y col. 2019).

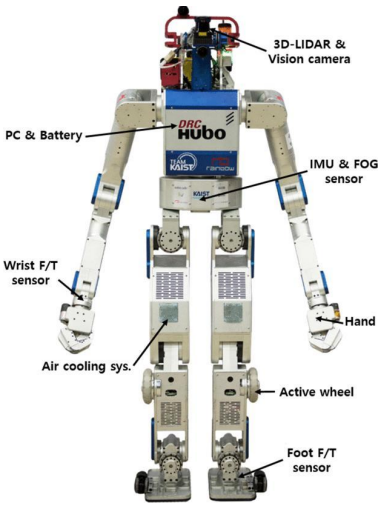

Fig. 1.3 DRC-HUBO+

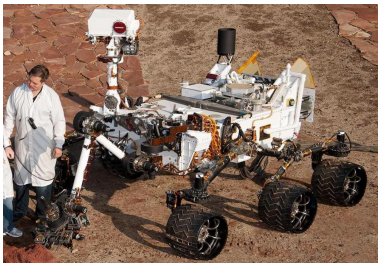

Fig. 1.4 Mars Curiosity Rover

Fig. 1.5 Spot Mini 2

Fig. 1.6 Depósito Ocado

Con este recorrido se podría decir que el área de aplicación de la navegación autónoma comprende: el hogar, la industria, el aire libre y otros planetas. Si se intenta categorizar las diferentes líneas de investigación dentro del área; estás se podrían dividir entre mejoras en los algoritmos de navegación, mejoras en los algoritmos de decisión y mejoras en la forma de percibir el ambiente (Choset y col. 2005).

Lo que generalmente se busca en el área de algoritmos de navegación es mejorar la eficiencia, ya sea desde un punto de vista de recursos necesarios para el procesamiento, como resultados conseguidos haciendo uso del mismo tipo de robots (Shitsukane y col. 2018).

Desde un punto de vista de las decisiones que toman los algoritmos; es importante que aprendan con menos trabajo (Hong y col., 2017; Hossen y col., 2015), que se equivoquen menos (Lee y col., 2017), que logren incorporar más información del ambiente (Kameyama y Hidaka, 2017) y que puedan manejar de manera exitosa la incertidumbre (Iswanto y col., 2016).

Con respecto a la percepción del ambiente se busca integrar nuevos tipos de sensores (Kameyama y Hidaka, 2017), nuevas formas de que el robot conozca su ubicación dentro del ambiente (Hsiao y Lee, 2017), mejorar la generación de mapas del entorno en tiempo real (Xi y col., 2017) y realizar visión artificial con los recursos limitados que tenemos en un robot (Jin y col., 2014). Se prueban distintos tipos de sensores (como GPS, Kinetic, LIDAR entre otros) que obtendrán mayor información del ambiente. Los objetivos en las mejoras de sensores son: reducir los costos con resultados similares o mejores, conseguir más información que permita tomar decisiones más informadas o buscar nuevas formas de lograr ubicarse (Zhang y col., 2015).

Los objetivos a lograr suelen ser: determinar el camino más corto, el más seguro, el más rápido o el que demande menor gasto energético al robot. Para lograr un comportamiento integral se suelen combinar distintos tipos de algoritmos (Iswanto y col., 2016; Hong y col., 2017; Poorva y col., 2018). Esto nos lleva a que los trabajos no solo

presenten mejoras para algoritmos puntuales, sino también distintas combinaciones que mejoren los resultados ante determinados escenarios (Lee y col., 2017).

Como se detalló en los objetivos generales de la tesis (subsección 1.1.1) se busca crear un algoritmo que logre navegar de forma autónoma cualquier tipo de ambientes desconocidos e invariantes en el tiempo. Y que si no existe un camino, el algoritmo logre acumular suficiente información para poder decidir que el mundo no tiene solución. De ahora en más en el trabajo denominaremos esto como **navegación o solución completa**, ya sea porque encuentra la solución o por que se da cuenta de que no hay solución.

Finalmente existe una nueva área de trabajo que intenta buscar formas de integrar los algoritmos de planeamiento de caminos en un hardware específico. Los trabajos en este sentido están en sus primeros pasos y no hay muchos ejemplos sobre el tema, pero se pueden ver algunas de las aproximaciones actuales en el artículo Moore (2019) que tiene como base la patente de intel Tschirschnitz y col. (2019).

Para poder acotar el área de estudio el presente trabajo se va a enfocar en ambientes estáticos e invariantes en el tiempo.

1.3 Navegación Autónoma

Existen distintos objetivos para la navegación autónoma. Este trabajo se enfoca en ambientes cerrados e invariantes en el tiempo, y la búsqueda del camino más óptimo sin información previa del ambiente. En las siguientes secciones se describe esto con más detalle, pero el objetivo es darle al robot autonomía para llegar a su objetivo.

Esta restricción del alcance deja fuera trabajos relacionados con la conducción autónoma de vehículos como el Tesla Autopilot (Morando et al., 2021) o también el análisis de imágenes en tiempo real como en el AWS DeepRacer (Lueck, 2020). Este trabajo se centra en robots dentro de ambientes estructurados, como en los almacenes de Amazon (Feiner, 2021) o JD.com (Hornyak, 2018). En los almacenes los robots se encuentran en

ambientes variantes en el tiempo, este trabajo estudia un sub-caso de estos ambientes (los invariantes en el tiempo).

Dentro de esta sección se revisarán los conceptos básicos del campo de estudio y se presentan los algoritmos y arquitecturas necesarias para poder resolver el problema.

1.3.1 Planeamiento De Caminos

Existen diversos tipos de algoritmos de planeamiento de caminos. Algunos se basan en mapas, otros en descomposición en celdas, otros en campos potenciales y otros en planeamiento de trayectorias (Gasparetto et al., 2015). Una vez que uno divide el mundo a explorar en celdas, se puede hacer planeamiento de caminos utilizando algoritmos de búsqueda como Astar, Dstar, y otros (Le et al., 2018).

Dependiendo del caso de estudio, puede alcanzar un solo algoritmo de éstos para resolver un problema o ambiente planteado. Pero esto solo suele suceder para ambientes o problemas sencillos. Cuando enfrentamos problemas o ambientes complejos el utilizar un solo algoritmo no alcanza, esto se puede ver en (Bounini et al., 2017) (Lee et al., 2017) y en muchos otros trabajos.

Dentro de los algoritmos de mapas existe una técnica llamada localización y mapeo simultáneo (normalmente denominado SLAM por sus siglas en inglés) que es de particular utilidad cuando ya se ha recorrido al menos una vez el camino. Dentro de esta categoría de algoritmos hay al menos 2 subgrupos: los basados en sensores LIDAR y los que utilizan cámaras y visión artificial (Wang et al., 2019). Si bien es un área interesante de estudio, estos algoritmos necesitan conocer el ambiente previamente y por lo tanto quedan fuera del alcance de este trabajo.

1.3.2 Caracterización De Algoritmos

Para el propósito del trabajo fue importante hacer una clasificación de los algoritmos de manera de saber cuáles eran los más adecuados o posibles candidatos cuando fuera necesario combinar varios de ellos para obtener el objetivo deseado.

Dentro de los algoritmos utilizados tenemos los de búsqueda (como el Astar), los de navegación (como APF), la descomposición del mundo en celdas, y los de caracterización del ambiente (como el Brushfire). Todos estos algoritmos son utilizados en conjunto para poder resolver el problema planteado.

1.3.3 La Necesidad De Una Arquitectura Híbrida

Cuando se desea tener una navegación completa en distintos tipos de ambientes (como fue definida en la sección 1.2 y se puede ver en el glosario) se hace necesario utilizar una arquitectura híbrida (Zhang et al., 2007). Los algoritmos híbridos se componen de dos capas, una reactiva y otra deliberativa (Hossian et al., 2014).

La capa reactiva toma los datos del ambiente y con ellos decide el camino a seguir en base a lo que determina el algoritmo actuante. Al navegar e ir descubriendo el ambiente, la navegación reactiva se encuentra con puntos de los que no puede salir o situaciones que pueden llevar a un camino muy largo hasta llegar a la solución. Lo que hace la capa deliberativa es decidir qué algoritmo reactivo es el más apropiado en la situación actual o si debe cambiar de estrategia de navegación.

Para poder utilizar un algoritmo deliberativo debemos acumular información sobre el ambiente y la navegación. Cuanto mejor sea esta información, mejores decisiones podrá tomar el algoritmo deliberativo. Adicionalmente el algoritmo deliberativo puede tomar decisiones con respecto a qué algoritmo reactivo usar, si son necesarios objetivos intermedios (o locales) que mejoren la navegación y si seguir navegando es necesario.

Solo con una capa deliberativa, utilizando la información recolectada de la navegación, se podrá decidir si el ambiente tiene una solución o no.

1.4 Metodología De Desarrollo De La Tesis

En esta sección se describen los pasos metodológicos correspondientes con el desarrollo de la presente tesis. Se siguió un enfoque de investigación clásico (Creswell, 2015) con el objetivo de desarrollar una arquitectura que permita navegar ambientes desconocidos de manera autónoma y logre determinar un camino al objetivo o declare la imposibilidad de alcanzarlo.

El primer paso consiste en una investigación documental sobre diferentes algoritmos de navegación autónoma, arquitecturas y soluciones, realizando revisiones sistémicas de los artículos científicos (Pallas & Jiménez Villa, 2019) en función de los objetivos propuestos en esta tesis.

En el segundo paso, y a partir de la investigación documental, se avanza en el desarrollo de un prototipo evolutivo experimental (Basili, 1993) para dar cumplimiento a los objetivos de la tesis, es decir, para lograr la navegación completa (sección 1.2). Este prototipo evolutivo experimental es refinado continuamente a lo largo del desarrollo de la tesis mediante su aplicación a casos de estudio de complejidad creciente.

Se proponen casos hasta que se logre demostrar la validez del modelo propuesto, y por último se elabora un informe final en el cual se presenta la metodología propuesta y se indican futuras líneas de investigación.

1.5 Publicaciones Científicas Vinculadas A Esta Tesis

Una versión reducida de este trabajo fue publicada en el XII Congreso Internacional sobre Aplicación de Tecnologías de la Información y Comunicaciones Avanzadas (ATICA2021)

bajo el título “Propuesta de arquitectura híbrida para navegación autónoma de agentes robóticos móviles en ambientes desconocidos e invariantes en el tiempo”.

Bonelli, I., & Hossian, A. (2021). Propuesta de arquitectura híbrida para navegación autónoma de agentes robóticos móviles en ambientes desconocidos e invariantes en el tiempo. En Aplicación de Tecnologías de la Información y Comunicaciones Avanzadas y Accesibilidad ATICA2021. Universidad de Alcalá. (173-180). ISBN: 978-84-18979-68-2

1.6 Estructura General De La Tesis

En este capítulo de **Introducción** se presenta el contexto del trabajo de tesis y su estructura.

En el capítulo **estado de la cuestión** se presentan los distintos elementos que se tienen en cuenta a la hora de buscar y explorar soluciones para la navegación autónoma.

En el capítulo **descripción del problema** se presentan varios estudios sobre la navegación autónoma con diversas aproximaciones. En particular, se hace foco en los últimos estudios sobre algoritmos de búsqueda y campos potenciales.

En el capítulo **solución propuesta** se presentan arquitecturas posibles de una solución, sus problemas y las soluciones encontradas.

En el capítulo **casos de experimentación** se detallan los ambientes de prueba y simulaciones sobre la solución propuesta.

En el capítulo **conclusiones** se revisan los resultados finales y futuras líneas de investigación.

Finalmente, el **apéndice** incluye información sobre el desarrollo de las simulaciones y código generado para poder llevarlas a cabo.

2. ESTADO DE LA CUESTIÓN

En este capítulo se presentan los distintos elementos que se tienen en cuenta a la hora de buscar y explorar soluciones para la navegación autónoma. En **percepción y movimiento** (sección 2.1) se explica por qué es una parte importante de la disciplina y qué sensores y actuadores se pueden usar hoy. En **algoritmos** (sección 2.2) se hace una revisión de las distintas disciplinas de las que se alimenta la navegación. En **arquitecturas reactivas, deliberativas e híbridas** (sección 2.3) se revisan las distintas posibles formas de combinar algoritmos de navegación. En **tipos de robots** (sección 2.4) se describen los distintos tipos y sus particularidades. En **tipos de ambiente** (sección 2.5) se intenta caracterizar los ambientes y las complejidades que estos presentan. Y en **casos reales** (sección 2.6) se intenta demostrar los problemas que presenta la navegación.

2.1. Percepción Y Movimiento

2.1.1. Sensores Y Percepción Del Ambiente

Una parte importante del área de estudio de la navegación está dada por las posibilidades tecnológicas actuales para poder percibir el ambiente. La experimentación con distintos tipos de sensores (como GPS, Kinetic, LIDAR entre otros) nos permite evaluar cuán buenos son estos sensores para percibir el ambiente.

Fig. 2.1 Sensor LIDAR 360

Fig. 2.2 Microsoft Kinect

La información que se busca obtener del ambiente es: dónde nos encontramos, qué tenemos entre nosotros y nuestro objetivo, qué obstáculos tenemos alrededor nuestro, qué tamaño y forma tienen esos obstáculos, cuán cerca nuestro están y cuánto esfuerzo nos llevará transitar un camino.

Los sensores más básicos utilizados son los de proximidad. Detectan de manera poco precisa la distancia que tenemos entre el sensor y un objeto u obstáculo enfrente de él. La percepción o visión de este tipo de sensores se encuentra limitada en un ángulo, solo ven una región cónica frente a ellos. Por lo tanto para tener una visión espacial amplia deberíamos usar varios de estos sensores colocándolos estratégicamente alrededor de nuestro robot. Estos sensores siguen siendo utilizados entre otras cosas por su bajo costo. Como principio de medición pueden utilizar un haz infrarrojo, o sonido.

Fig. 2.3 Sensor por ultrasonido

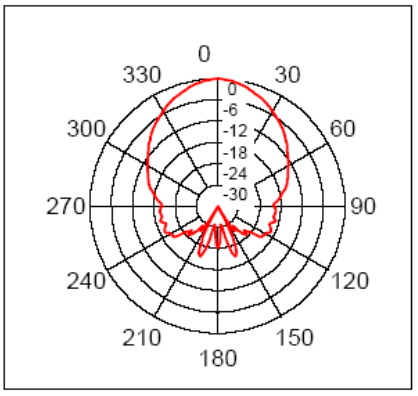

Fig. 2.4 Campo "visión" de un sensor por ultrasonido

Luego tenemos las cámaras de video. Estas suelen ser económicas como sensores, pero traen acarreado un alto costo de procesamiento y requieren algoritmos bastante demandantes desde un punto de vista de procesamiento. El uso de estos sensores es un

área de estudio específica denominada visión artificial. Desde un punto de vista de navegación lo que se busca obtener de esta conjunción sensor más algoritmo de visión artificial es la ubicación del robot, los obstáculos que se tienen alrededor y estimar distancias. Para realizar esto generalmente se utilizan dos cámaras de video y un módulo de visión artificial.

Existe actualmente un sensor denominado Kinetic que fue desarrollado para la consola de juegos XBOX de Microsoft. Se han realizado varias investigaciones utilizando este sensor (Kameyama & Hidaka, 2017) (Smisek et al., 2013) ya que tiene un costo moderado y resuelve varios de los problemas de la visión artificial. El sensor está compuesto por las cámaras y una plataforma de cómputo que entrega resultados ya procesados de lo que ven los sensores.

Otro tipo de sensores son los de detección y medición láser (Light Detection and Ranging o LIDAR). Estos sensores son más caros que los de proximidad, pero son mucho más precisos y tienen una visión de 180, 270 y hasta 360 grados. Generalmente, tienen una visión circular, no cónica. Por lo tanto, solo ven cosas que se encuentren directamente a su altura. Esto hace que sean utilizados para obtener mediciones precisas del ambiente que permitan generar un mapa, pero para la navegación se complementan con otros sensores como los de proximidad o cámaras de video.

2.1.2. Sensores Y Ubicación Espacial

Además de los sensores que nos permiten percibir el ambiente que rodea al robot, tenemos los sensores de ubicación que permiten conocer su ubicación X, Y y Z dentro de un marco de referencia prefijado.

El más conocido es el sistema de ubicación global (Global Positioning System o GPS) que a través de un red de satélites ubicados alrededor de la tierra nos permiten saber la ubicación que tenemos. La red GPS solo funciona en la tierra y al aire libre, pero existen otras formas de obtener el mismo tipo de información para ambientes cerrados.

Una forma de ubicarse en ambientes cerrados es utilizar imágenes QR que identifiquen la ubicación y cámaras de video/foto para leerlas (Zhang et al., 2015). En la navegación en otros planetas se puede utilizar visión artificial y una amplia base de imágenes para lograr el mismo objetivo. En resumen, existen varias formas de obtener nuestra ubicación, ya sea directamente con sensores o mediante una conjunción de sensores y un módulo de procesamiento.

Fig. 2.5 Uso de códigos QR para ubicación

dentro de un depósito

Existen otras investigaciones sobre cómo ubicarse en ambientes cerrados (Mautz, 2009) utilizando por ejemplo los access point WiFi en ambientes con IEEE 802.11 (Kul et al., 2014) y también existe una gran cantidad de métodos agrupados en lo que llaman iGPS (Franceschini et al., 2011) o RTLS (Real-Time Location System) provistos por compañías como Blueiot, Eliko, Sewio y Amrikart.

2.1.3. Motores Y Actuadores

Se puede definir un actuador como un dispositivo que convierte energía en un movimiento físico. La gran mayoría de los actuadores producen movimientos rotacionales o lineales. Dentro del conjunto de los actuadores rotacionales los más usados son los motores y de estos, los más comunes en robótica son los eléctricos. Dentro del conjunto de los actuadores lineales se encuentran los pistones, estos pueden ser hidráulicos o neumáticos.

Existen otros ejemplos de actuadores (como los piezoeléctricos) y a medida que el tamaño de los robots se reduce, también se hace necesario investigar nuevos tipos de actuadores.

Si bien todos los tipos de actuadores son utilizados para crear robots, los más comunes para los estudios de algoritmos de navegación no simulados son los motores eléctricos. Dentro de los motores eléctricos existe una clasificación que los divide en base a su fuente de alimentación: corriente alterna y corriente continua. Los motores de corriente alterna suelen ser más eficientes y en general se usan para tareas industriales. Pero dentro del área de estudio los más comunes son los de corriente continua.

Los motores se pueden utilizar de manera directa o como parte constitutiva de lo que termina siendo un actuador compuesto. Dentro de algunos ejemplos se pueden mencionar los motores con cajas de engranajes rotativos (fig. 2.6) o de movimiento lineal (fig. 2.7).

Fig. 2.6: Motor con caja de reducción

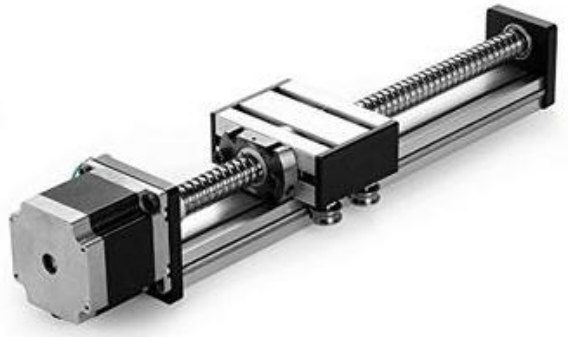

Fig. 2.7: Motor acoplado a un sistema de desplazamiento lineal

Cuando se trabaja con motores es importante tener en cuenta el control de los mismos. Para controlarlos se construyen actuadores del tipo servo (fig. 2.8) que nos permiten moverlos dentro de un rango limitado, contadores de vueltas o sensores de fin de carrera. También existen los motores paso a paso que si bien son rotativos, tienen la

particularidad de moverse una cantidad fija de posiciones, grados o pasos dentro del movimiento circular (fig. 2.9).

Fig. 2.8: Motor servo

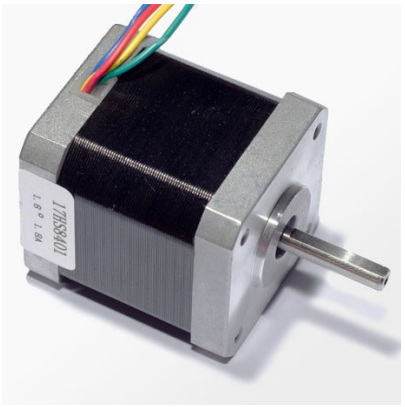

Fig. 2.9: Motor paso a paso

Existe una nueva generación de actuadores altamente integrados que no solo incluye al motor, si no también a sensores y un bus de comunicación. Un ejemplo de este tipo de actuadores son los actuadores X-Series de HEBI Robotics (fig. 2.10). Estos actuadores sensan en todo momento la posición del motor. Adicionalmente se pueden conectar en cascada facilitando la conexión de varios motores.

Fig. 2.10: Actuador de HEBI Robotics X-Series

Si bien es importante tener en cuenta todos los tipos de actuadores que existen a la hora de diseñar un robot, dentro del área de estudio de este trabajo presentan un interés menor. La razón fundamental para la falta de foco en este tema es que todos van a presentar el mismo problema: no pueden proveer una idea real de dónde se encuentra el robot. Los rozamientos y deslizamientos con el suelo terminan generando un error importante en cualquier intento de posicionamiento del robot mediante el actuador. La elección del motor a utilizar está más relacionada con la capacidad de movimiento que le queremos dar al robot que con el planeamiento de caminos del mismo.

2.2. Algoritmos

Existen varios tipos de algoritmos que pueden ayudar en la navegación. En esta sección se mencionan los distintos tipos que pueden llegar a utilizarse. En el capítulo siguiente se amplía sobre los más relevantes para el presente trabajo.

Una de las ramas de estudio sobre algoritmos usada por la navegación es el recorrido sistemático de árboles. Estos algoritmos no necesitan conocer el ambiente para poder trabajar, pero si lo conocemos podemos ahorrar energía al desplazarnos y planear el camino en forma anticipada usando nuestro conocimiento del ambiente. Por lo tanto, una rama de estudio es la que busca mejores sensores, configuraciones y algoritmos para generar un mapa o modelo del ambiente. A esta técnica se la denomina sistemas de ubicación y mapeo simultáneo (en inglés el acrónimo usado es SLAM). El objetivo es no solo recorrer el ambiente en pos de un objetivo, sino también generar un mapa del ambiente mientras se navega.

Con respecto a los métodos de navegación existen dos grandes grupos: los algoritmos reactivos y los deliberativos.

Los algoritmos reactivos carecen de historia o conocimiento previo del terreno. En base a la información instantánea de los sensores se toma la decisión de cómo navegar.

Estos algoritmos pueden ser sencillos o complejos, pero siempre van a estar limitados a la hora de lograr objetivos en ambientes complejos. Tienen el problema de poder caer en ciclos repetitivos y no lograr su objetivo. Pero a la hora de tomar decisiones y navegar son extremadamente rápidos y son muy buenos para evitar los obstáculos inmediatos.

Por otro lado, tenemos el otro gran grupo de los algoritmos deliberativos. Estos algoritmos suelen ser más demandantes desde un punto de vista de procesamiento y necesidad de memoria, pero tienen la ventaja de poder manejar situaciones complejas y tener un grado de éxito mucho mayor.

El uso de algoritmos deliberativos nos lleva a tener una mayor necesidad de información del ambiente. Cuanto más sepamos del ambiente, nuestros algoritmos podrán tomar decisiones más informadas. Esto se hace evidente cuando se recorren ambientes complejos. Particularmente, los del tipo laberinto presentan un problema ya que si no conocemos de antemano su disposición se puede necesitar invertir mucho esfuerzo en lograr encontrar el camino a nuestro objetivo.

Los tipos de algoritmos a usar suelen ser de localización, de búsqueda (generales o de camino), para mantener un registro del camino realizado y para generar un modelo del ambiente (o mapeo). Luego tanto en el área deliberativa como en la reactiva se pueden utilizar redes neuronales artificiales o lógica difusa. Como el área de búsqueda es muy amplia la lista de algoritmos también lo es, pero incluso hoy en día se pueden ver trabajos usando algoritmos tan clásicos con él A estrella (Hart et al., 1968).

A continuación se ve con más detalle algunos algoritmos utilizados o evaluados para generar el modelo que resuelve el problema planteado en este trabajo.

2.2.1. Algoritmos De Búsqueda

La búsqueda de algoritmos para calcular el mejor camino no comienza con la navegación autónoma de robots. Esta disciplina comenzó antes junto a problemas como el clásico viajante de comercio (Papadimitriou, 1977). Existen muchos algoritmos clásicos que

permiten resolver este tipo de problemas como A estrella (Astar), Dijkstra y otros. Estos algoritmos son de búsqueda exhaustiva. Hay algunos mejores que otros, pero todos hacen una búsqueda completa por "fuerza bruta" hasta, en el peor de los casos, haber recorrido todo el espacio de búsqueda hasta encontrar la solución.

Para lograr resultados con un mayor grado de "inteligencia" el foco ha cambiado a otros tipos de algoritmos o combinaciones de ellos. Por ejemplo, es bastante común la inclusión de lógica difusa en al menos algún aspecto de la búsqueda ya que permite una mejor toma de decisiones (incluso cuando no se tiene toda la información). Pero no se debe descartar la búsqueda exhaustiva ya que en ciertos ambientes (como los laberínticos) son la mejor forma de actuar para encontrar el destino que buscamos. Las aproximaciones usadas actualmente suelen combinar distintos tipos de navegación y decidir de acuerdo a la situación cuál utilizar.

2.2.2. Redes Neuronales Artificiales

Las redes neuronales artificiales son unas de las primeras técnicas utilizadas en este campo para la navegación reactiva. El trabajo pionero de W. Walter (1961) utilizaba, por ejemplo, dos neuronas artificiales para que su robot pudiera decidir qué camino tomar.

Las neuronas artificiales son circuitos que modelan el comportamiento de las neuronas reales. Una neurona real (fig. 2.11) está compuesta por múltiples entradas (dendritas) y múltiples salidas (terminales sinápticas). Para poder modelizarla (fig. 2.12) se toma cada salida por separado en función de todas las entradas. Las terminales sinápticas pueden ser inhibidoras, excitadoras o neutras. Finalmente, la salida de una neurona artificial se define como la sumatoria de las funciones de entradas afectadas por una función de activación. Este modelo fue inicialmente propuesto por McCulloch y Pitts (1943) y evolucionó en el perceptrón (Rosenblatt, 1958) que presentaba solo una salida.

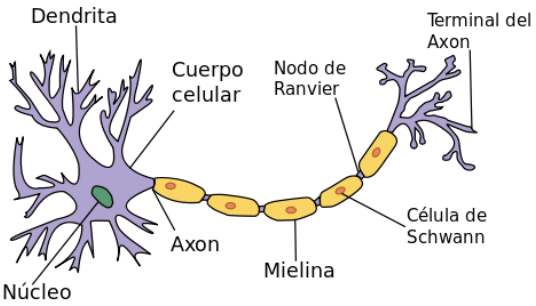

Fig. 2.11 Neurona Biológica

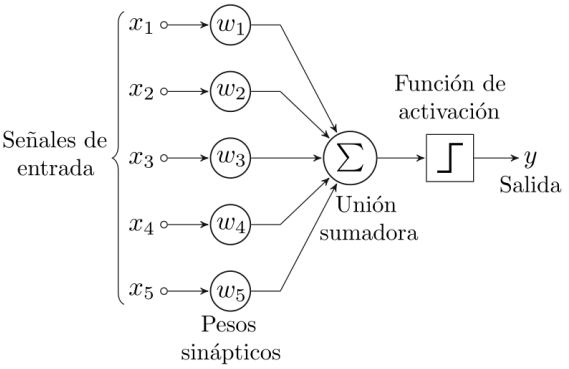

Fig. 2.12 Neurona Artificial

Para calcular la salida que tendremos en una neurona artificial como la de la figura 2.12 debemos aplicar la fórmula:

$$y = f_{act}\left(\sum_i w_i * x_i\right) \qquad (2.1)$$

Donde f_{act} es la función de activación que puede variar dependiendo del modelo de red neuronal que estemos usando.

Estas neuronas artificiales se agrupan en redes que tienen varias neuronas. Dependiendo de la topología de la red, formas de interconexión entre las neuronas, cantidad de capas y el método de aprendizaje utilizado tendremos distintos tipos de redes. Un ejemplo de red bastante útil es la RNA backpropagation (fig. 2.13) la que usando hasta tres capas permite aprendizaje que converge en una red estable (Hossian, García y Verónica 2014).

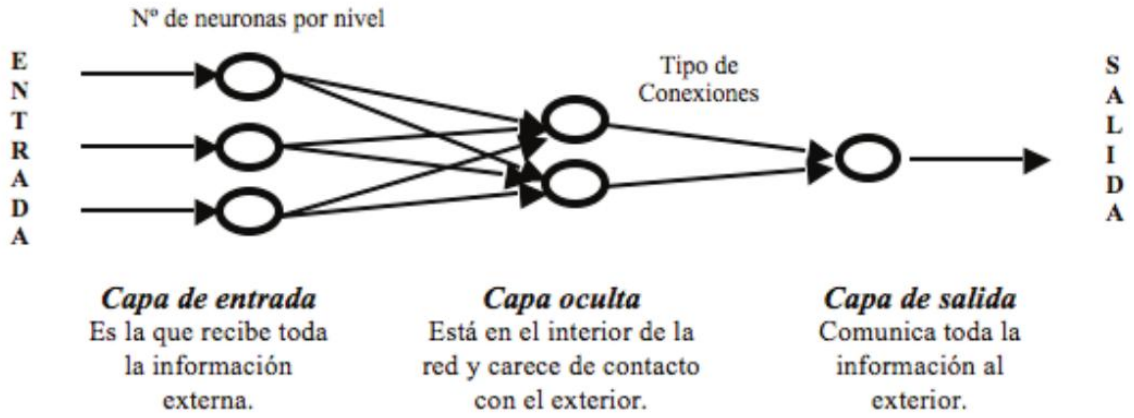

Fig. 2.13 Red neuronal del tipo RNA

Las redes neuronales pueden usarse tanto para buscar soluciones de navegación como en el trabajo de Balbuena, Colón y Scillato (2006) o para generar soluciones de visión artificial.

2.2.3. Campos Potenciales Artificiales

El uso de campos potenciales artificiales (APF en inglés) para la navegación está muy difundido (ver Baturone, 2005) y hay muchos trabajos utilizándolos en ambientes desconocidos como por ejemplo el trabajo de Lee et al., 2017 que mejora APF utilizando un nuevo punto de atracción o el trabajo de Poorva et al., 2018 que utiliza APF y Astar para controlar un conjunto de robots en forma conjunta. Los trabajos pioneros en esta área fueron de Khatib, 1986 y de Krogh y Thorpe, 2005 (primeramente publicados en 1986).

La idea detrás de usar los campos potenciales artificiales para la navegación es crear dos campos a componer: Uno atractivo que lleve al robot al punto de destino, y otro repulsivo que lo aleje de los obstáculos que hay en el camino (ver fórmula 2.2). La conjunción de ambos le proporcionará al robot la dirección con mejores posibilidades de navegación desde el punto en el que se encuentra.

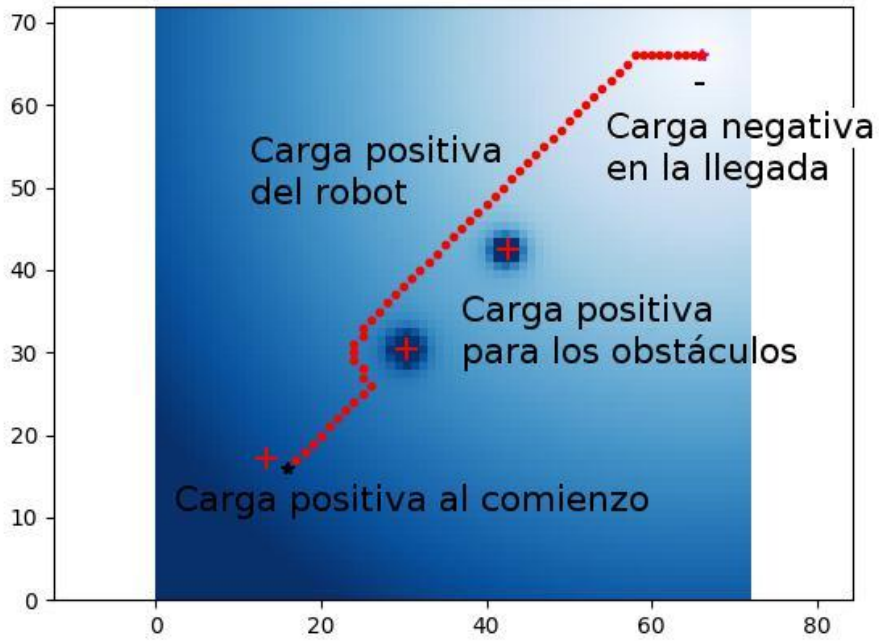

Fig. 2.14 : Explicación funcionamiento APF

Se puede ver como si los obstáculos y el robot tuvieran la misma carga eléctrica y por lo tanto hubiera entre ellos una fuerza eléctrica repulsiva. Y como el destino tiene una carga distinta, lo atrae. En la fig. 2.14 podemos ver un ejemplo de ambiente con dos obstáculos y las cargas asociadas al robot, los obstáculos y el destino.

La siguiente fórmula describe al campo atractivo y el repulsivo que se componen para determinar el camino:

$$U(q) = U_{atr}(q) + U_{rep}(q) \qquad (2.2)$$

Donde U_{atr} es el campo atractivo que nos lleva al objetivo, U_{rep} es el campo repulsivo que nos permite evitar los obstáculos y q es la posición a calcular. El resultado U es un campo que nos dará una fuerza resultante para cada punto del ambiente a utilizar para

dirigir al robot. La variable q es la posición en x,y del ambiente (plano) para la cual estamos calculando el campo.

Dada la ecuación 2.2 se puede ver la navegación como un problema de optimización donde debemos encontrar el mínimo global U que nos lleve desde la posición inicial hasta la final. El algoritmo más sencillo para resolver este problema es un gradiente descendente (sección 5.2 del libro Spong, Vidyasagar y Hutchinson, 2006).

Podemos considerar al gradiente negativo de U como la fuerza F que actuará sobre el robot para atraerlo a su destino.

$$F(q) = -\nabla U(q) = -\nabla U_{atr}(q) - \nabla U_{rep}(q) \qquad (2.3)$$

Para el modelo del campo atractivo (que nos lleva al objetivo) se puede utilizar un modelo lineal (C=R) o cuadrático ($C=R^2$). Si usamos el modelo lineal la velocidad hacia el objetivo siempre será la misma. Si usamos un modelo cuadrático la velocidad disminuirá a medida que nos acerquemos al objetivo. Generalmente, se consiguen mejores resultados usando una combinación de ambos dependiendo de la distancia del objetivo a la que nos encontremos. Suele ser útil tener una velocidad constante en la mayoría del camino (lineal, ver fig. 2.15) y pasar a un modelo de velocidad variable (cuadrático, ver fig. 2.16) cuando nos acerquemos al objetivo.

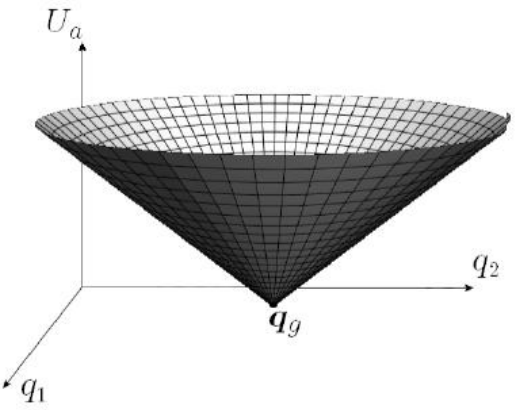

Fig. 2.15: Modelo lineal o Cónico

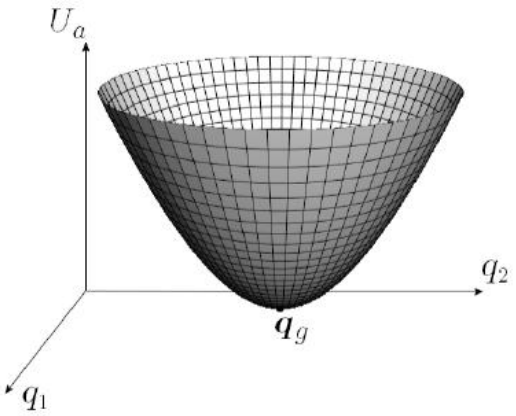

Fig. 2.16: Modelo Cuadrático o Parabólico

En la Fig. 2.17 podemos ver una simulación de navegación utilizando campos potenciales artificiales en un ambiente cerrado y estático. Los puntos oscuros representan obstáculos y sus fuerzas repulsivas. Vemos también un campo atractivo general que es oscuro en el inicio del camino y se hace claro hacia el objetivo.

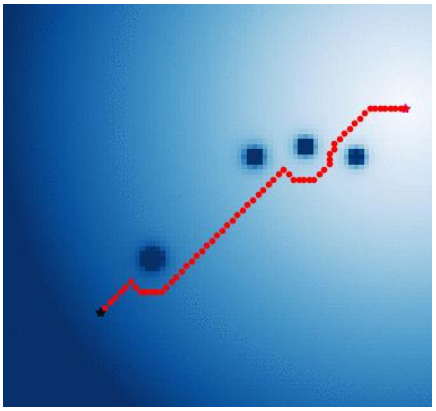

Fig. 2.17: Navegación APF

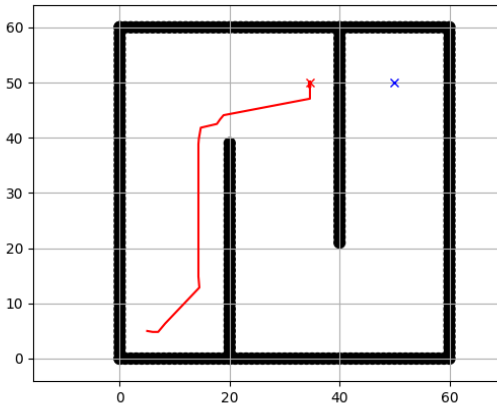

Fig. 2.18: Mínimo local

Para los obstáculos se utiliza un campo repulsivo que los rodea. Es posible considerar que cada lectura de un sensor de proximidad genera una fuerza repulsora proporcional o simplificar el espacio de navegación en celdas donde puede o no haber un obstáculo. En las celdas donde haya un obstáculo tendremos una fuerza repulsora puntual que actúa desde el centro de la celda. Esta división del espacio de navegación en celdas ayuda mucho a simplificar los cálculos y el volumen de información a manejar, pero también trae otros problemas aparejados como qué tamaño tiene que tener la celda. Querremos mayor cantidad de celdas cerca de un obstáculo y menor cantidad de celdas en el espacio libre. Se puede acudir al uso de celdas de tamaño variable, pero esto también conlleva otros problemas a resolver. Para mayor simplicidad se suele tomar celdas de tamaño fijo y se busca un tamaño de celda que tenga sentido dentro del ambiente en el que nos encontramos (subsección 12.4.1 del libro Ollero Baturone, 2001). También existen técnicas de construcción de celdas como los diagramas de Voronoi que permiten además hacer planeamiento de caminos (ver Bhattacharya y Gavrilova, 2008 o Sakai, 2017).

Como se ve en la figura 2.18 uno de los problemas de este modelo es la aparición de mínimos locales (o también líneas equipotenciales) para los cuales es difícil encontrar una solución. Esto trae dos problemas: Cómo detectar que estamos en un punto mínimo local o equipotencial y una vez detectado cómo evitarlo. La detección de un mínimo local se puede lograr registrando los movimientos históricos del robot, si tenemos muchos movimientos en un área pequeña del ambiente (ya que no siempre se va a quedar quieto). La forma más sencilla de evitar estos puntos de indecisión es utilizar otro algoritmo de navegación en estos casos. La primera y más sencilla aproximación es pasar a una navegación aleatoria, que si bien puede ser bastante útil para recorrer una línea equipotencial, no suele dar resultados tan buenos en mínimos locales como el presentado en la figura 2.17. Lo mejor en estos casos es tener un segundo algoritmo deliberativo que nos permita decidir el mejor camino a seguir. En la sección 3.1 se hace revisión de un trabajo de Lee, Jeong, Kim y Park, 2017 que

ha logrado mediante una modificación del algoritmo una forma de evitar el problema que presentan estos ambientes con mínimos locales.

2.2.4. BrushFire

El algoritmo BrushFire sirve para calcular distancias a obstáculos dentro de una representación del tipo grilla en dos dimensiones (dividiendo el mundo en celdas todas del mismo tamaño). Cada celda tiene un valor 0 si no contiene obstáculos, y 1 si al menos está parcialmente ocupada por un obstáculo. Este es el principio de construcción de Brushfire, pero la idea es guardar información de cada celda de manera que unas tengan conexión con otras. Se pueden conectar con un principio de 4 direcciones (sin tener en cuenta las diagonales) o de 8 direcciones.

El objetivo es llenar la grilla que representa el mundo con valores que representan la distancia al obstáculo más próximo. Las celdas con un valor de 0 adyacentes a un 1 tendrán un valor de 2. Las celdas con 0 adyacentes a un valor de 2 tendrán un valor de 3, y así sucesivamente. De esta manera cada celda libre de obstáculos representará la distancia al obstáculo más próximo (Choset y col. 2005 subsección 4.3.2).

Este algoritmo (así como los de búsqueda exhaustiva) necesitan tener la información completa del mundo para poder ser calculados, esto fuerza (en el caso del presente trabajo) a usar una versión modificada que se calcula a medida que se va ganando conocimiento del mundo.

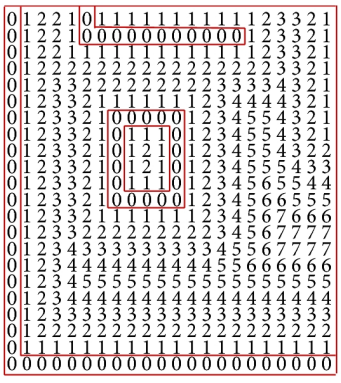

Fig. 2.19: Mapa con potenciales Brushfire

Fig. 2.20: Camino determinado por el algoritmo Wavefront

En la figura 2.19 se puede ver un mapa con todos los potenciales Brushfire calculados. Es interesante ver que determina zonas de equipotencial que nos dan un camino que permite evitar los obstáculos. En las figura 2.20 vemos una mapa calculado con el algoritmo de Wavefront que si bien es similar tiene un objetivo diferente, permite encontrar el camino. Esto lo realiza calculando de una manera similar, pero no distancias a obstáculos, si no al objetivo de navegación.

2.3. Arquitecturas Reactivas, Deliberativas E Híbridas

Cuando se desarrollan sistemas autónomos, es importante determinar cuán sofisticados van a ser. Los sistemas reactivos sólo pueden tener un comportamiento prefijado (como reflejo) sin mantener un estado interno (Nolfi, 2002).

En contraposición a esto, los agentes deliberativos se comportan como si pensaran, es como si buscaran el conjunto de comportamientos que tienen disponible, y teniendo en cuenta el estado actual, cuáles son las acciones a tomar más adecuadas para lograr el efecto deseado.

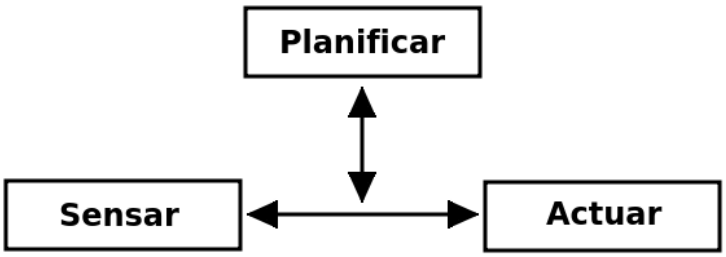

Fig. 2.21: Esquema simplificado de una Arquitectura Híbrida

Si bien los límites entre un sistema reactivo y deliberativo pueden ser difusos, normalmente un agente reactivo no guarda estado, y uno que puede predecir comportamientos debe hacerlo.

Hay una gran cantidad de algoritmos reactivos que pueden resolver problemas de navegación sencillos, pero cuando el sistema se complejiza los sistemas reactivos suelen quedar atrapados en mínimos locales (Lee y col., 2017) y situaciones donde es necesario usar sistemas deliberativos (o de búsqueda exhaustiva) para lograr los objetivos deseados (Poorva y col., 2018).

Cuanta menos información manejamos, más eficiente será el modelo reactivo (cada estímulo tiene una respuesta). Y cuanta más información manejamos, más sentido tendrá usar el modelo deliberativo (planeamos y luego actuamos).

Algunas razones para unir sistemas reactivos y deliberativos pueden ser mejorar la eficiencia en la toma de decisiones o mejorar el comportamiento del sistema. En cualquier caso, cuando se juntan ambos paradigmas se dice que creamos un sistema híbrido (Murphy, 2001).

2.4. Tipos De Robots

La variedad de robots que existe es muy amplia y el planeamiento de camino es una necesidad común a todos ellos. Pero dentro del área específica de la investigación hay una tendencia muy fuerte al uso de las Tortugas de Walter (W. Walter, 1961). Este modelo de robot tiene una ventaja muy importante a la hora de hacer investigación pura de navegación ya que carece de complejidades. Es la expresión más sencilla posible de lo que un robot que se mueva puede ser. En algún punto los robots Roomba de iRobot (ver figura 1.2) y los robots de los depósitos de Amazon (ver figura 2.22 y 2.23) son muy similares en diseño básico a lo que es una tortuga de Walter (ver figura 1.1).

Fig. 2.22: Depósito de Amazon con sus robots

Fig. 2.23: Robots para depósitos de OTTO AMRs

2.5. Tipos De Ambiente

Hay pocos trabajos que traten la clasificación de ambientes, pero en la revisión documental se encontró al menos uno que toma una aproximación similar y a la vez distinta. Si bien intenta clasificar, no lo hace con ambientes sino con situaciones (Na & Oh, 2003). Utilizando una red neuronal simple logran clasificar 16 tipos de situaciones y establecen patrones de reacción exitosos a las mismas. En otro trabajo (Sharma & Doriya, 2020) vemos un intento de clasificación más sencillo y orientado a caracterizar los algoritmos.

Es muy difícil caracterizar a los tipos de ambiente debido a la gran variedad que existen, pero en lo que se refiere al planeamiento del camino se podría decir que existen dos grandes tipos: los abiertos que permiten más de un camino para alcanzar el objetivo, y los cerrados (o laberínticos) que tienen pocos o un solo camino posible.

Para los ambientes abiertos generalmente un algoritmo reactivo es suficiente. Si bien algunos algoritmos muy sencillos pueden beneficiarse del agregado de otro algoritmo deliberativo, existe mucho investigado y hay algoritmos como el TangentBug (Kamon, Rimon & Rivlin, 1998) que son bastante completos y capaces de manejar este tipo de ambientes.

Cuando entramos en la navegación de ambientes cerrados, parte del trabajo es encontrar ese el camino al objetivo. La otra posibilidad es tener la información del ambiente (mapa) y poder planearlo previamente. Esta diferencia hace mucho a las necesidades que vamos a tener sobre los algoritmos. Si no tenemos información previa, las posibilidades de

entrar en una condición de carrera que no nos permita resolverlo son muchas. Si no agregamos un algoritmo deliberativo, difícilmente se logren los objetivos de navegación.

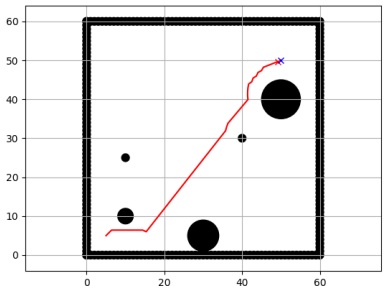

Fig. 2.24: Ejemplo de ambiente abierto

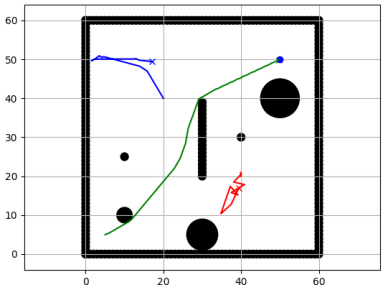

Fig. 2.25: Ambiente abierto con varias trayectorias

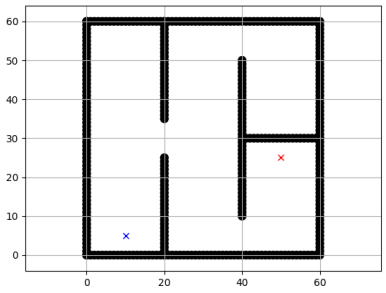

Fig. 2.26: Ambiente cerrado con solución

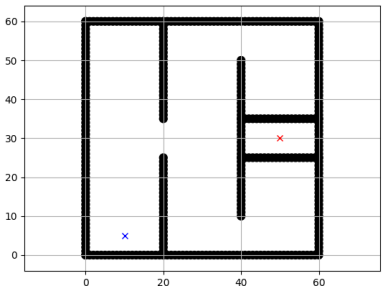

Fig. 2.27: Ambiente cerrado sin solución

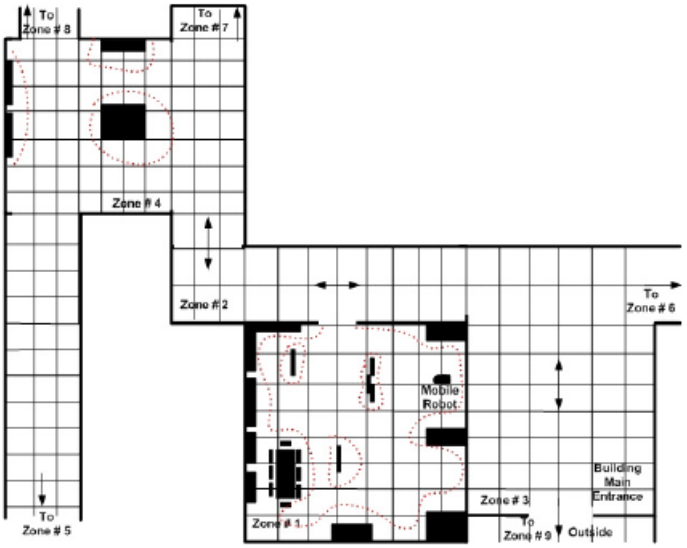

Fig. 2.28: Como vemos en esta imagen un plano de planta de una oficina se parece a un ambiente cerrado

2.6. Casos Reales

Hay muchos ejemplos de la necesidad de planeamiento y ejecución de la navegación hoy en día. Desde cuadricópteros para filmaciones que buscan seguir un objetivo evitando obstáculos, hasta autos que se manejan solos. Pero centrándonos en lo que vimos hasta ahora vamos a ver casos para cada uno de los ambientes propuestos.

Dentro de lo que denominamos como ambientes abiertos en este trabajo (ver 2.5) se puede mencionar la necesidad de navegación que tuvo por ejemplo el Mars Curiosity Rover (ver fig. 1.4) en Marte. Este es uno de muchos de los rovers utilizados actualmente para la exploración de otros cuerpos celestes. Solo para mencionar algunos ejemplos tenemos el Mars Pathfinder, los gemelos Spirit y Opportunity (ver fig. 2.29) que también exploraron Marte o el Yutu de China que exploró nuestra Luna (ver fig. 2.30). Como comenta el trabajo de Wong, Yang, Yan y Gu, 2017 en los robots de exploración espacial no se va a suprimir la intervención humana en un futuro cercano. La experiencia de los operadores y mejores técnicas de simulación permiten una necesaria asistencia a la navegación del robot en situaciones críticas. Solo se busca ganar autonomía y mejorar la navegación en situaciones

normales, pero para hacer esto debemos aplicar todas las técnicas de navegación que venimos viendo. La diferencia en estas situaciones es que deben tener una efectividad del 100 por ciento, y para llegar a esas cifras la intervención humana es invaluable.

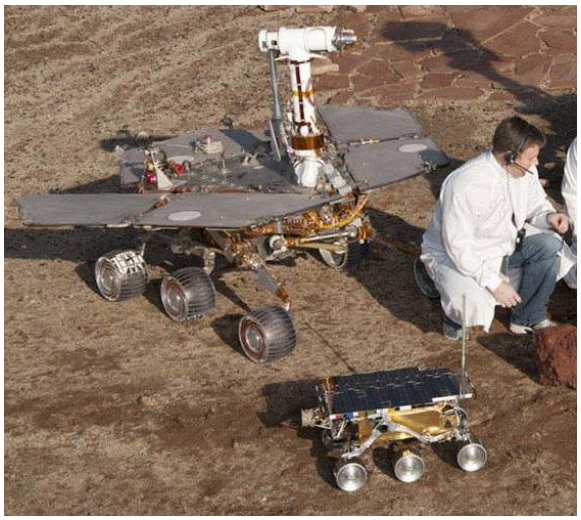

Fig. 2.29: El Mars Pathfinder (mas pequeño) y el Opportunity (detrás)

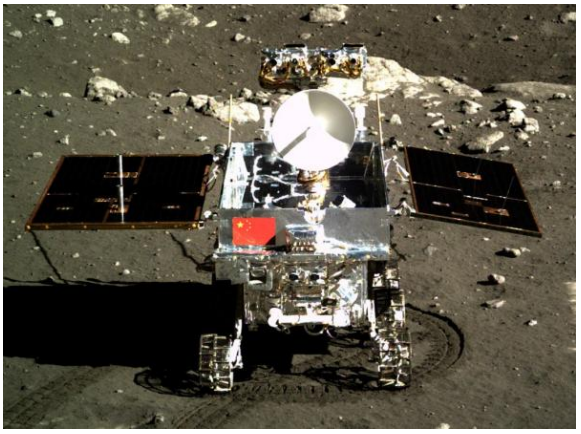

Fig. 2.30: El Yutu en la luna fotografiado por la ChangE

Si nos enfocamos en los ejemplos de ambientes cerrados mencionados en 2.5 podemos mencionar por ejemplo los diversos usos de robots en depósitos. Tanto los depósitos de Amazon (ver fig. 2.22), los de Ocado (ver fig. 1.3 y artículo «Estos robots te arman el pedido del supermercado en 5 minutos», 2017) o más recientemente los almacenes de JD.com sin operarios (ver fig. 2.31, fig. 2.32 y artículo «Así es el primer depósito sin operarios humanos que funciona solo con robots», 2018).

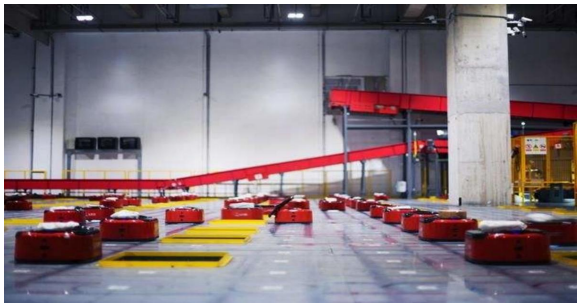

Fig. 2.31: Una visión general de la parte del depósito de JD.com con robots del tipo tortuga

Fig. 2.32: Un detalle del trabajo del robot dentro del depósito de JD.com

A diferencia de los robots de ambientes abiertos, en ambientes cerrados la imprevisibilidad se reduce y la automatización se lleva al máximo. Estos robots, ambientes, sensores y algoritmos se diseñan para no depender de la intervención humana. Esto es posible ya que al tener un ambiente controlado y poder incrementar la cantidad de información que recibe el robot del ambiente, los algoritmos pueden lograr un grado de certidumbre que les permita saber cómo proceder siempre. Si algún escenario presenta problemas se busca la forma de aumentar la información que recibe el robot y reducir la incertidumbre para lograr que siempre sepa cómo proceder.

3. DESCRIPCIÓN DEL PROBLEMA

En este capítulo se presentará primero una **introducción** al problema (sección 3.1), luego los **algoritmos de navegación** evaluados (sección 3.2), en **combinando los algoritmos** (sección 3.3) se revisan los algoritmos juntos con sus configuraciones reactivas/deliberativas, y finalmente **recolectando información del mundo** (sección 3.4) se da una introducción a cómo se guarda la información recolectada del mundo.

3.1. Introducción

Desde un punto de vista general la navegación presenta distintos tipos de problemas detallan en el capítulo 2 sobre el estado de la cuestión. Cada uno de los temas es un área de estudio en sí misma y esta tesis se enfoca en el problema de la navegación en un ambiente desconocido e invariante en el tiempo.

Un agente sin conocer el ambiente deberá lograr llegar al destino con el menor esfuerzo posible. Dentro de este trabajo se entiende por menor esfuerzo posible, en la menor cantidad de pasos posible. El objetivo es lograr hacerlo en distintos ambientes. Para caracterizarlos se utilizará la clasificación propuesta en **tipos de ambiente** (sección 2.5).

Se busca crear un agente que pueda resolver cualquier tipo de ambiente. Para lograr esto es importante combinar distintas estrategias (Zhang et al., 2007), ya que presentando ambientes de cualquier tipo (abiertos o cerrados) y sin conocer el ambiente en forma previa, ningún algoritmo reactivo logra por sí solo encontrar una solución completa (Arkin & MacKenzie, 1994). Combinando distintos tipos de algoritmos se buscará acercarse lo más posible al planeamiento que se puede hacer conociendo el mundo de antemano. Y adicionalmente detectar la imposibilidad de encontrar un camino cuando no exista.

Para relevar el ambiente existen diferentes aproximaciones posibles (Algabri et al., 2015). En este caso se utilizará un solo agente autónomo que aprenderá la topología y

obstáculos que hay en el mundo al que se lo expone por sí solo. En otros casos se recurre a un grupo de agentes que trabajan en forma comunitaria y comparten información (Poorva et al., 2018), pero ese no es el objetivo de la presente tesis.

Existen distintos desafíos para resolver este problema:

1. Elegir entre la gran variedad de algoritmos de navegación existentes.
2. Lograr una combinación que gracias a la cooperación de los algoritmos permita resolver la mayor cantidad de ambientes posible.
3. Buscar estrategias de navegación que permitan recolectar la mayor cantidad del ambiente posible.

En las siguientes secciones de este capítulo se describe la visión del problema, los posibles caminos y la dirección que se toma en esta tesis.

3.2. Algoritmos De Navegación

Los algoritmos que se evaluaron para ser combinados en una solución híbrida fueron: tangentBug, Wave Front, Dynamic Window Approach, RRT, APF, Brushfire, Dijkstra, Astar. Se revisaron más algoritmos, pero por problemas de compatibilidad entre ellos o dificultades a la hora hora de articularlos se dejaron de lado.

Inicialmente se evaluó el uso de Dynamic Windows Approach (Fox et al., 1997) para la capa reactiva. Pero durante las pruebas se lo descartó ya que sus fortalezas (permitir ajustar la velocidad de exploración) no eran compatibles con los objetivos a resolver. Si bien el siguiente paso fue utilizar el algoritmo tangentBug (Kamon et al., 1998), este terminó siendo dejado de lado en favor de APF. tangentBug es bueno navegando, pero no permitía interactuar con la capa deliberativa con un lenguaje común. Esto se debe a que para la recolección de la información del mundo se decidió utilizar una matriz homogénea de ocupación o grilla con celdas de igual tamaño (similar a lo que vemos en la figura 2.19).

Otro algoritmo candidato para la capa reactiva podría haber sido el D-Bug (Abafogi et al., 2018). Pero este algoritmo es bastante reciente, y si bien presenta algunas ventajas, no hay estudios actuales que permitan inferir mejoras significativas sobre tangentBug.

Para la recolección o evaluación de la información recolectada del mundo se evaluaron Wave Front y Brushfire (Kalra et al., 2009). Ambos son bastante similares, pero Brushfire permite una mejor articulación con APF. Se ajusta mejor a las necesidad de la arquitectura elegida porque solo recolecta información y descarga la responsabilidad de navegación en otro algoritmo. Se terminó implementando la interacción entre algoritmos en la capa deliberativa, pero durante el trabajo también se evaluó la posibilidad de que ambos trabajaran en conjunto para evitar los problemas de mínimos locales que presenta APF.

En el avance del desarrollo del trabajo se constató la necesidad de tener un tercer algoritmo que permitiera evaluar de forma completa toda la información recolectada. Esto permite evitar puntos de no retorno y determinar si el objetivo es alcanzable o no. Para esta capa se evaluó utilizar Dijkstra (Dijkstra, 1959) o Astar (Hart et al., 1968). Se constató la mayor eficiencia de Astar y se decidió utilizarlo.

3.3. Combinando Los Algoritmos

Para resolver el problema el primer paso fue decidirse por un algoritmo reactivo. De alguna manera esta elección fue condicionando los otros algoritmos a elegir. La razón del condicionamiento es que cada algoritmo tiene sus fortalezas y debilidades, por lo tanto todos deben complementarse y contribuir a la solución completa (sección 1.2).

El primer algoritmo elegido de la capa reactiva fue APF. Si bien se consideraron otros algoritmos (ver capítulo anterior), su eficiencia en ambientes desconocidos y sus problemas de indecisiones (Rostami et al., 2019) permite un punto de articulación con la capa

deliberativa. Al ser tan conocido y claro su problema de indecisión, es sencillo detectarlo y decidir pasar el control a la capa deliberativa cuando APF necesita ayuda.

Esta elección de APF, también llevó a buscar algoritmos que permitieran evitar su problemas de mínimos locales. La decisión de utilizar un grilla homogénea o matriz para guardar la información relevada del mundo llevó a elegir Brushfire (Kalra et al., 2009) que no solo permitía esto, si no también complementa a APF. La elección de Astar (Hart et al., 1968) tuvo dos razones: la necesidad de llegar a una solución completa y utilizar la mejor opción exhaustiva disponible.

El algoritmo Astar utilizado tiene una variación mínima en cuanto al cierre. Puede terminar por tres razones: al encontrar la solución, al confirmar que no existe o también puede llevar al robot al punto más cercano al objetivo donde no se tenga información del mundo. En este último caso lleva al robot al punto menos conocido y le devuelve el control a la capa reactiva. Desde ese punto desconocido el robot vuelve a buscar alcanzar su objetivo de manera reactiva y continúa recolectando información del mundo.

Se utilizó también un algoritmo de seguimiento que permite navegar en línea recta hasta un objetivo fijado por la capa deliberativa. Esta fue otra necesidad de asistencia que surgió de la experimentación. Fundamentalmente, permite a la capa deliberativa fijar objetivos que nos saquen de los mínimos locales que afectan a APF.

El algoritmo de decisión o deliberativo principal (que veremos en detalle en el siguiente capítulo) surgió de la necesidad de articular eficientemente entre el algoritmo reactivo APF, los problemas de mínimo locales y el algoritmo de búsqueda exhaustiva Astar (o deliberativo secundario). No sigue una arquitectura o algoritmo conocido, sino que surgió orgánicamente del estudio de los resultados obtenidos con los algoritmos elegidos.

3.4. Recolectando Información Del Mundo

Una de las ventajas más importantes que nos da la utilización de un algoritmo reactivo como APF es que nos permite navegar de manera eficiente (alcanzar el objetivo con la menor cantidad de pasos posible) con muy poca información. Esta navegación es aprovechada para recolectar información del mundo en el que nos encontramos.

Para guardar la información recolectada se utilizó un modelo de sensores que permitiera definir cuántas direcciones puede evaluar del mundo, y hasta qué distancia puede evaluarlo. Inicialmente, solo se recolectó información de ocupación dentro de la grilla, pero luego se fue enriqueciendo la información recolectada con datos que surgen de la evaluación de los datos de APF y Brushfire. Esta información adicional es utilizada en la toma de decisiones de la capa deliberativa.

4. SOLUCIÓN PROPUESTA

En este capítulo se presenta una posible solución al problema planteado, y como fue el desarrollo para llegar a la misma. En **el camino a una solución** (sección 4.1) se desarrolla como se desarrolló el trabajo, en **el por qué de los mundos elegidos** (sección 4.2) se da más información sobre los mundos, en **la simulación del sensado del ambiente** (sección 4.3) se dan detalles sobre cómo se simularon los ambientes, en **la capa deliberativa** (sección 4.4) se explica las generalidades de la arquitectura construida y finalmente en **cómo se construye la información del mundo** (sección 4.5) se dan detalles de cómo se guarda la información del mundo para poder tomar decisiones informadas.

Adicionalmente en el anexo A.2 de revisión sistemática de la literatura se muestra el camino realizado para soportar las elecciones de algoritmos que se realizaron.

4.1. El Camino A Una Solución

En esta sección se explica cómo se fue desarrollando el trabajo y se justifican las decisiones tomadas. Se arranca con **la necesidad de una arquitectura híbrida** (sección 4.1.1), luego el porqué de **la elección de APF para la capa reactiva** (sección 4.1.2), se revisa **la necesidad de guardar información del ambiente** (sección 4.1.3), cómo APF **articula con el algoritmo FollowPath** (sección 4.1.4), se explica **la necesidad de plantear un objetivo local** (sección 4.1.5), el **por qué del uso de Astar en la capa deliberativa** (sección 4.1.6) y finalmente se presenta **la arquitectura general de la solución** (sección 4.1.7),

4.1.1. La necesidad de una arquitectura híbrida

Se hizo una revisión sobre algoritmos de navegación, estrategias y arquitecturas. Toda esta investigación y las experimentaciones permitieron concluir que para lograr una navegación completa (sección 1.2), se debe utilizar una estrategia híbrida (Wang et al., 2018) (Sedighi et al., 2019) (Ajeil et al., 2020). Para navegar un ambiente cerrado (como fue caracterizado el el sección 2.5) que no conocemos, no va alcanzar con un sistema reactivo, ni con un sistema deliberativo.

Si conocemos el ambiente de antemano, el usar un sistema deliberativo es suficiente. Pero al no conocer el ambiente de antemano, debemos recorrer todo el mundo. Esto es un esfuerzo exhaustivo y no permite alcanzar el objetivo.

Si en su lugar se comienza a navegar con un algoritmo reactivo, avanzamos en simultáneo con el conocimiento del mundo, y vamos avanzando hacia el objetivo. Pero los algoritmos reactivos pueden llegar a situaciones de las que no puede salir.

Estas limitaciones de cada arquitectura y la necesidad de navegar sin conocer el mundo (impuesta por el planteo del problema) nos deja con la necesidad de tener un sistema híbrido.

Una vez establecida la arquitectura híbrida, se podrá resolver cualquier tipo de ambiente, ya sean del tipo abierto o cerrado (cómo fueron caracterizados en la sección 2.5). En general, las estrategias híbridas exitosas conjugan un mecanismo reactivo y otro deliberativo. En el caso del presente trabajo y por la búsqueda de resolución de cualquier ambiente, veremos que se conjugaron varios algoritmos de cada tipo de acuerdo a las necesidades de navegación.

En el área de estudio del planteamiento de caminos existe mucho trabajo realizado buscando mecanismos de navegación que se complementen unos a otros. Debido a la gran cantidad de algoritmos que existen las combinaciones posibles son muchas. En el presente trabajo se buscó una combinación nueva que permitiera navegar exitosamente cualquier ambiente.

El primer desafío del trabajo fue encontrar un algoritmo reactivo que nos permitiera tanto navegar, como así también ir relevando las características del terreno. La idea es que el algoritmo esté enfocado en navegar todo lo posible sin requerir asistencia de otro algoritmo, y aprovechar esta navegación para ir recolectando información del ambiente. Esta información del ambiente será luego utilizada por la capa deliberativa. Si bien no es un requerimiento, también era deseable que el algoritmo a utilizar no debiera ser entrenado. Al menos no para un ambiente particular, el objetivo es que el autómata no necesite conocimientos previos del terreno a navegar.

Dentro de los algoritmos que podrían ser utilizados en la capa reactiva se consideraron los algoritmos del tipo insecto (siendo el más significativo el TagentBug Ng, 2005), los de ventana dinámica (Dynamic Windows Approach), las redes neuronales artificiales (RNA) y los de campos potenciales artificiales (APF).

Las RNA tienen necesidad de entrenamiento, por lo tanto, si bien fueron consideradas, no eran un candidato fuerte. Los primeros intentos fueron con el algoritmo TangentBug, que si bien tiene sus ventajas y puede resolver ambientes complejos,

dependiendo del ambiente puede elegir caminos ineficientes (dando más pasos de los estrictamente necesarios).

Luego se intentó utilizar el algoritmo de ventana dinámica que tiene sus ventajas como no necesitar discretizar el ambiente, pero por el otro lado la discretización nos permitirá acumular información sobre el ambiente ya navegado.

4.1.2. La elección de APF para la capa reactiva

Finalmente se decidió utilizar para la capa reactiva el algoritmo de campos potenciales artificiales (APF) ya que se utiliza extensamente y ha sido muy estudiado. Esto hace que sean bien conocidas sus ventajas y desventajas.

La mayor falencia de APF es la existencia de puntos de indecisión debido a mínimos locales (Choset y col. 2005, sección 4.4). Cuando se utiliza en ambientes abiertos es menos probable encontrar un mínimo local y permite encontrar un camino eficiente en la gran mayoría de los casos (en el capítulo 5 de experimentación hay ejemplos). Es incluso exitoso y eficiente cuando no se conoce el terreno. Pero cuando lo utilizamos en un ambiente cerrado y/o con objetos no puntuales la probabilidad de caer en mínimos locales aumenta. Cualquier objeto que se interponga en el camino y no tenga un punto de escape claro nos dejará a un autómata utilizando APF en un punto del cual no podrá salir (ver un ejemplo en la figura 2.18).

Si bien APF presenta esta falencia, siempre que utilicemos un método reactivo (cualquiera sea) y no tengamos la posibilidad de calcular la ruta de manera global (lo que nos daría conocimiento completo del terreno y los obstáculos) existirá la posibilidad de que nuestro algoritmo reactivo falle. Por esta razón se buscaron otros algoritmos que complementen al reactivo.

Se encontraron diversas combinaciones con APF que permiten evitar los mínimos locales:

- Modified Artificial Potential Field Approaches for Mobile Robot Navigation in Unknown Environments (Singh et al., 2020)
- An Improved Artificial Potential Field Method with a New Point of Attractive Force for a Mobile Robot (Lee et al., 2017)
- Obstacle Avoidance Path Planning Based on Improved APF and RRT (An et al., 2021)
- A fuzzy based local minima avoidance path planning in autonomous robots (Teli & Wani, 2021)
- A new APF strategy for path planning in environments with obstacles (Agirrebeitia et al., 2005)

Estos trabajos y otros más plantean el uso de APF como algoritmo reactivo base, y buscan alguna manera novedosa de detectar y evitar los mínimos locales.

Una de las razones para elegir un algoritmo como el de APF es la posibilidad de explorar el terreno y acumular información del ambiente mientras lo recorremos con una probabilidad alta de ir en la dirección correcta. Al menos en lo que respecta a navegación libre de obstáculos. Si al progresar en la exploración del ambiente se encuentran obstáculos, al menos estos se presentan sobre el camino que era el más eficiente a priori (sin información del terreno).

Al principio se intentaron aproximaciones similares al trabajo de Lee et al. 2017, pero las pruebas demostraron que para utilizar esas aproximaciones de manera exitosa se debía tener un rango de visión amplio. Incluso con un campo de visión amplio existían situaciones en las que era difícil determinar cuál estrategia era la mejor, o situaciones donde se llegaba a un punto en el que APF fallaba.

Primero se intentó limitar los grados de libertad de APF de forma que solo pudiera navegar en las direcciones que se detectaban como mejores. Pero esta aproximación no dio los resultados esperados y se buscó otras alternativas.

4.1.3. La necesidad de guardar información del ambiente

Una vez elegido APF y comenzadas las pruebas fue evidente que se debía guardar la información del ambiente de una manera que pudiera ser utilizada por una capa deliberativa para tomar mejores decisiones de las que eran posibles con APF. Se buscaron distintas alternativas y la construcción de un wavefront del tipo brushfire permitió aportar mayor cantidad de información del ambiente con el menor esfuerzo. A medida que recorremos el mundo, el conocimiento aumenta y se pueden tomar mejores decisiones. El solo hecho de recolectar el espacio libre explorado y los obstáculos nos da bastante información, pero al sumarle el peso de cada celda nos puede ayudar a tomar mejores decisiones.

Así a cada paso se almacena la decisión de navegación y todos los indicadores para cada dirección de exploración. Estos datos incluyen: el potencial APF, si existe un obstáculo, la distancia al obstáculo, un indicador del tamaño del obstáculo y cuánto conocemos del mundo en esa dirección. Todo esto permite mejorar las decisiones a futuro. La construcción de estos indicadores está basada fuertemente en los datos del algoritmo BrushFire y la interpretación de ellos para cada punto visitado. Estos datos al navegar se van actualizando, pero las decisiones tomadas se basan en la información que estaba disponible en ese momento.

4.1.4. La articulación con el algoritmo FollowPath

Si bien el algoritmo principal de la capa reactiva es APF, existe una segunda opción que es fijar un punto y caminar en línea recta hasta alcanzarlo. A este algoritmo se lo denominó FollowPath. Se encontró que en ciertas situaciones puede asistir a APF y evitar los mínimos locales de una forma similar al algoritmo NP-APF de Lee, et. al. (2018). La decisión entre el uso de estos algoritmos fue la génesis de la capa deliberativa. A esta primera articulación se llegó buscando, de alguna manera, suplir los problemas de APF y dotarlo de un comportamiento similar al que se consigue con el algoritmo tangentBug (sección 3.2).

La decisión de involucrar a la capa deliberativa en la navegación y definición de mínimos locales está basada en la información recolectada y la interpretación de la misma en 3 niveles básicos:

- Falla del sistema reactivo APF
- Falla del sistema de navegación FollowPath (o seguir dirección)
- Detección de situaciones “trampa”

Se detectaron 2 tipos de situaciones trampa. Las que en este trabajo denominaremos tipo 1 (figura 4.1), se puede interpretar como una esquina que tiene un mínimo local y debemos evitar. También se definen trampas del tipo 2 (figura 4.2), que se pueden interpretar como un callejón sin salida. Una vez dentro de ellas, para evitarlas, solo podremos retroceder.

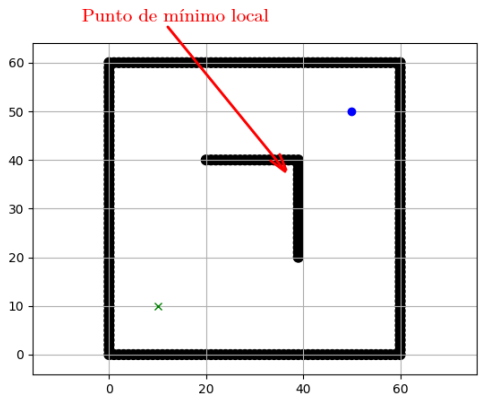

Fig. 4.1: Trampa tipo 1

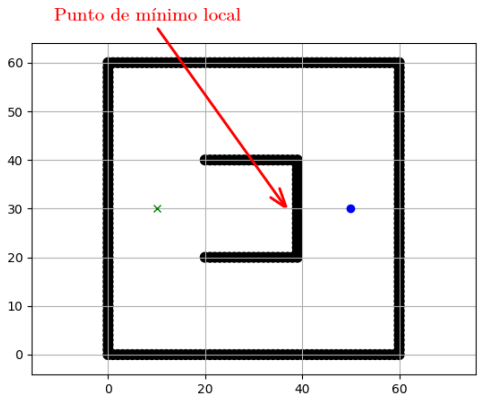

Fig. 4.2: Trampa tipo 2

Cuando se detecta una trampa o hay una falla en las navegaciones básicas, se cambia del sistema reactivo APF a una navegación también reactiva a los obstáculos pero coordinada por el sistema deliberativo. Puede ser de seguimiento de dirección o APF, pero cambiando el objetivo global por uno local que nos permita vencer el mínimo local.

4.1.5. La necesidad de plantear un objetivo local

El objetivo global es el que queremos alcanzar, pero cuando hay problemas en la navegación ayuda a cualquier algoritmo el plantear un objetivo local a alcanzar, que permita evitar el problema presentado para alcanzar el objetivo global. Este objetivo local debe estar planteado para que una vez alcanzado, permita continuar la navegación hacia el objetivo global.

La elección de un objetivo local que nos permite evitar el mínimo local la realiza la capa deliberativa. Lo hace en base a la información con la que dispone. Puede decidirlo en base a la información de navegación, el camino realizado, o su conocimiento del mundo actual.

Siempre que sea posible se intenta navegar de forma reactiva. De esta manera, una vez evitado el problema de navegación mediante un objetivo local, la capa deliberativa le devuelve el control al sistema de navegación reactivo APF.

4.1.6. La capa deliberativa y la necesidad de Astar

En el proceso de búsqueda de una solución se descubrió que en ciertas situaciones la capa deliberativa no logra resolver los problemas con algoritmos reactivos. Cuando esto sucede, la capa deliberativa necesita algún algoritmo de navegación exhaustivo o completo. Si bien el uso de este algoritmo se encuentra restringido al mundo conocido hasta el momento, permite en base al mapa generado en memoria encontrar un nuevo punto desconocido o validar que no existe camino posible. Para esta búsqueda exhaustiva se eligió usar una versión de Astar modificada. Si bien se orienta al objetivo global, termina cuando encuentra el punto más cercano a ese objetivo que no conoce. Lo establece como objetivo local y le devuelve el control a la capa reactiva APF o FollowPath. Solo al llegar al punto desconocido se retorna al objetivo global.

Esto permitió tener una navegación completa (sección 1.2) en un mundo desconocido. Si bien el algoritmo Astar puede resolver mundos conocidos, para poder

conocerlo debe recorrerlo. Y la capa deliberativa/reactiva permite navegar en la mayoría del mundo recolectando la información, que de ser necesaria, terminará dando una resolución segura Astar.

4.1.7. La arquitectura general de la solución

La solución planteada se puede ver desde el punto de vista de navegación como un sistema de decisiones que se grafica en la figura 4.3. Adicionalmente podemos visualizar el comportamiento de la capa deliberativa con el diagrama de la figura 4.4, en este podemos visualizar además el uso del algoritmo exhaustivo (buscar nuevos puntos de exploración).

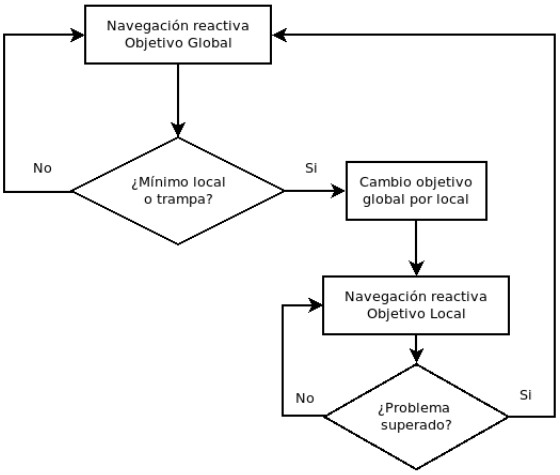

Fig. 4.3: Sistema de navegación general

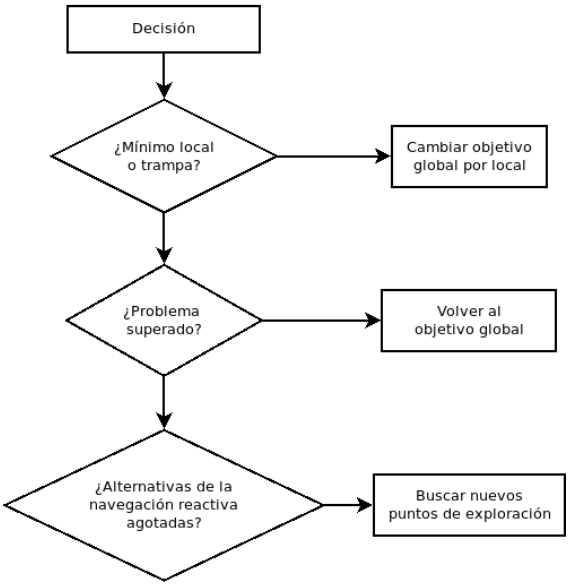

Fig. 4.4: Capa deliberativa

Como base del desarrollo se utilizó en un principio el trabajo de PythonRobotics (Sakai, 2017). A partir de esa idea inicial se expandió a una estructura de clases, se agregaron los algoritmos necesarios y se completó el sistema de desarrollo con creación de ambientes, registros de la navegación y gráficos en Python. Esto permite gran flexibilidad a la hora de evaluar distintos ambientes 2D y el registro de los datos de navegación.

4.2. El Por Qué De Los Mundos Elegidos

Para poder demostrar el funcionamiento del algoritmo de planeamiento de caminos debemos presentar un conjunto de mundos a resolver idealmente con una complejidad creciente. Esto permite ver paso a paso la necesidad de las distintas partes del sistema resultante.

Antes de proceder a la elección de los mundos que se utilizaron en el presente trabajo, se realizó una revisión exhaustiva de los ya utilizados, a los efectos de probar algoritmos de planeamiento de caminos. Se observa que muchos trabajos eligen los mundos inteligentemente para poder demostrar las fortalezas de los algoritmos. Por ejemplo, en Iswanto et al., 2016 se ve que los mundos elegidos solo pretenden demostrar un punto, no exploran los límites del algoritmo en forma exhaustiva. En el trabajo de Paliwal & Kala, 2018 también vemos una intencionalidad en los mundos elegidos. Estos son solo 2 ejemplos, esta elección de los mundos de manera discrecional se puede observar en gran cantidad de trabajos de planeamiento de caminos.

Existen dos razones para realizar esta elección discrecional de los mundos, la primera es la necesidad de demostrar las razones por las cuales se crea el algoritmo. Es común que para que la situación sea clara, se deba elegir o crear un mundo que muestre el problema que se intenta solucionar. La segunda, un poco menos obvia, es que hasta hace muy poco no existían propuestas de pruebas estandarizadas para la evaluación de algoritmos de planeamiento de caminos.

Los trabajos de Wen et al., 2021 (A unified benchmark for the evaluation of mobile robot local planning approaches) y Heiden et al., 2021 (A motion planning benchmark for wheeled mobile robots) son dos propuestas que si bien son muy recientes y no tienen trabajos que las avalen o usen, demuestran un avance que permitirá en el largo plazo poder comparar los algoritmos de mejor manera. Se estima que trabajos como estos o posteriores

permitirán en el largo plazo una base de evaluación común para los diferentes algoritmos de navegación.

Estos últimos trabajos son muy recientes y el presente trabajo intenta demostrar el camino que llevó a la solución creada. Por lo tanto, así como en muchos otros trabajos, a continuación se presenta un conjunto de mundos con complejidad creciente que permitan demostrar el funcionamiento del algoritmo.

El primer set de ambientes solo presenta objetos sencillos. Según la definición planteada en la sección 2.5 estos son ambientes abiertos que pueden normalmente ser resueltos por algoritmos reactivos.

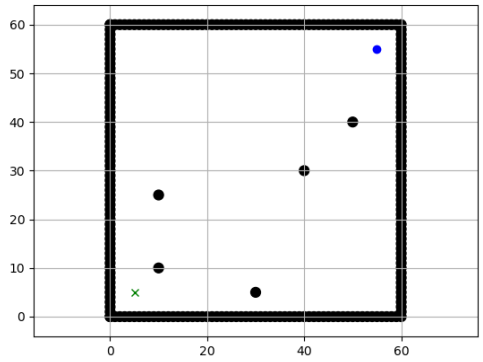

Fig. 4.5: Mundo 01

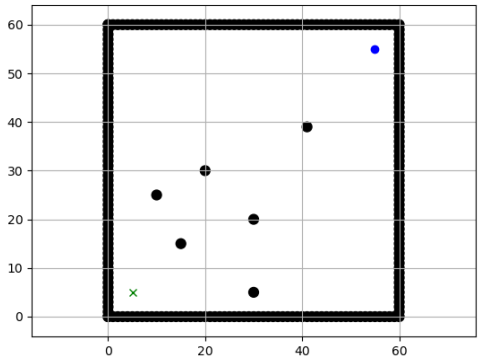

Fig. 4.6: Mundo 02

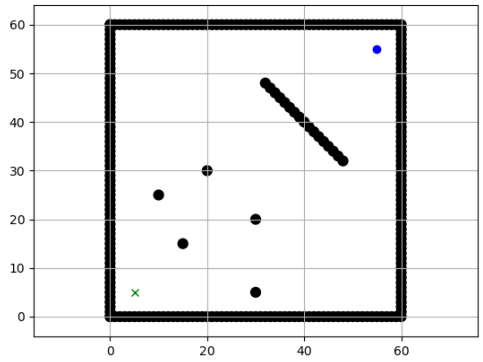

Fig. 4.7: Mundo 03

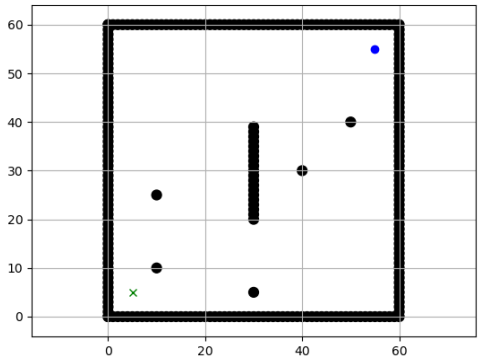

Fig. 4.8: Mundo 04

Estos cuatro mundos (figuras 4.5, 4.6, 4.7 y 4.8) representan los más sencillos y ayudaron en las pruebas que permitieron elegir el algoritmo reactivo a utilizar. Una vez terminadas esas pruebas y en parte por la elección de APF para la capa reactiva se creó un segundo conjunto de mundos que permitieran llevar el algoritmo a sus límites.

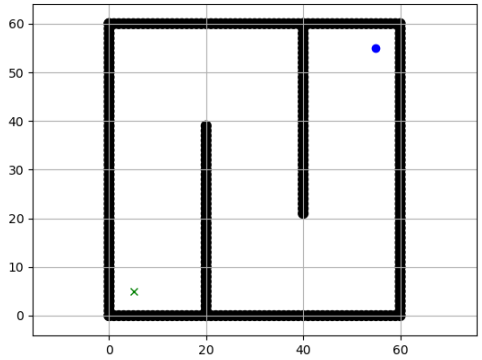

Fig. 4.9: Mundo 11

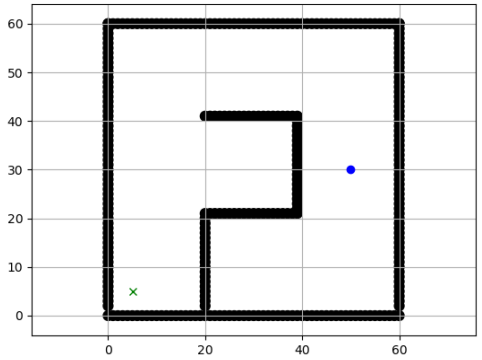

Fig. 4.10: Mundo 12

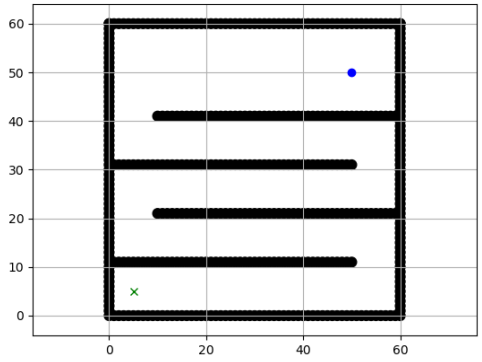

Fig. 4.11: Mundo 13

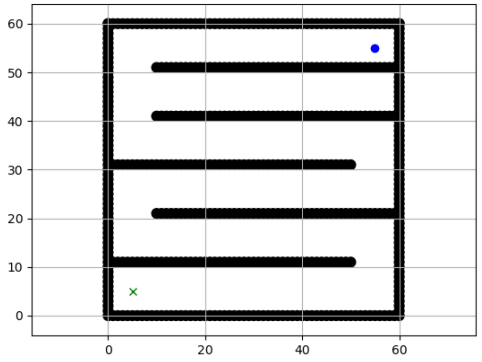

Fig. 4.12: Mundo 22

Los mundos de la figuras 4.9 y 4.10 tuvieron como propósito probar que el algoritmo que trabajaba con APF pueda evitar las trampas ya caracterizadas en la sección 4.1 figuras 4.1 y 4.2. Ya fuera APF o no, el algoritmo a desarrollar debe superar exitosamente estos dos mundos.

Los mundos de las figuras 4.11 y 4.12 permiten probar cómo responde el algoritmo a ambientes del tipo laberinto. Nuevamente el algoritmo a desarrollar debe poder resolver estos mundos que van teniendo una complejidad creciente.

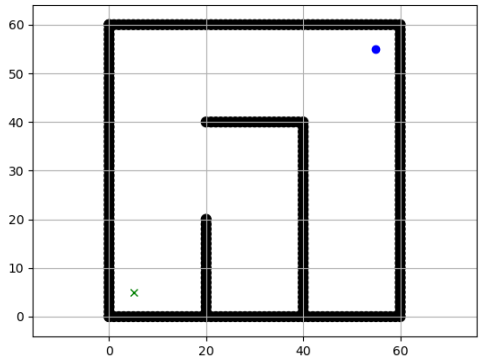

Fig. 4.13: Mundo 21

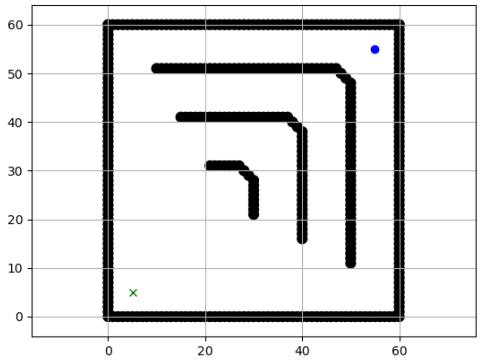

Fig. 4.14: Mundo 24

Si bien los mundos van aumentando en complejidad, existen ciertas situaciones que dentro de ambientes similares presentan problemas aún más complejos de resolver. Por ejemplo, el mundo de la figura 4.13 tiene cierta similitud con el mundo de la figura 4.10, pero en las experimentaciones veremos que presenta nuevos problemas. Otro mundo interesante es el de la figura 4.14 que si bien parece sencillo, presenta una complejidad alta a la hora de resolverlo sin información previa del ambiente.

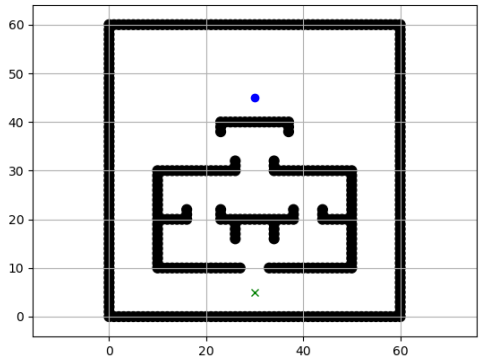

Fig. 4.15: Mundo 31

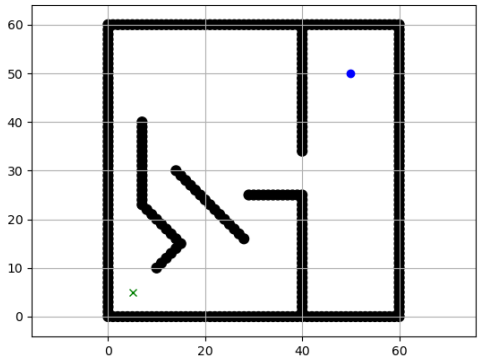

Fig. 4.16: Mundo 32

Los mundos de las figuras 4.15 y 4.16 intentan representar ambientes complejos, laberínticos y con algún paralelo a ambientes reales. Solo al agregar el algoritmo Astar el sistema propuesto logró resolverlos.

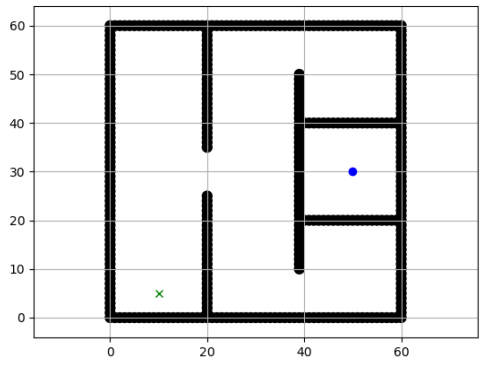

Fig. 4.17: Mundo 99

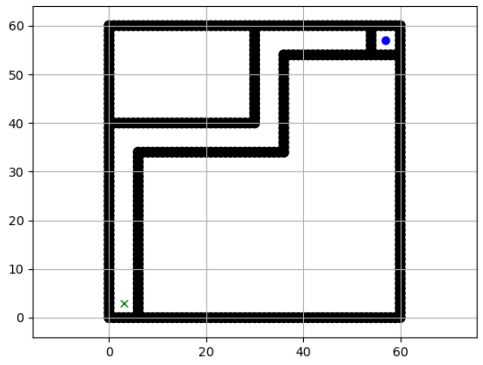

Fig. 4.18: Mundo 96

Finalmente, se debían presentar ambientes sin solución como los de las figuras 4.17 y 4.18. Aquí el objetivo es que el sistema detecte que el mundo no tiene solución.

4.3. La Simulación Del Sensado Del Ambiente

Este trabajo utilizó originalmente los desarrollos compartidos en PythonRobotics (Sakai, 2017), pero ni bien se comenzó a trabajar sobre los algoritmos se debió expandir la arquitectura/simulación de la percepción del mundo. El trabajo original simplemente detectaba los objetos, pero en el presente trabajo se generó una abstracción como sensor LIDAR que permite determinar los grados de “visión” del robot y cómo se perciben los obstáculos cuando aumentamos los sensores del robot.

En parte, esta línea de trabajo fue inicialmente necesaria para poder simular el algoritmo tangentBug (Kamon et al., 2002). Pero a medida que se avanzó con el algoritmo, se mantuvo esta simulación del sensado del ambiente ya que permite realizar distintos

experimentos con respecto a la información que necesita el robot para poder tomar decisiones.

Si bien el ambiente para el robot es desconocido, el sistema de simulación tiene que poder acceder a él. Esto fuerza al sistema a tener por un lado la información del mundo a recorrer y por el otro lo que ve el robot de este mundo a medida que avanza.

Como se adelantó en la sección 3.1, para modelizar la información del mundo se decidió utilizar una matriz homogénea de ocupación o grilla con celdas de igual tamaño (similar a lo que vemos en la figura 2.19). Cada celda tendrá un estado de ocupada o no. A medida que el robot avanza, va accediendo a la información dentro de su rango sensorio. Por ejemplo, el sistema puede modelar un sensor que solo “vea” en 4, 8 o más direcciones. Además, se puede ajustar cuánto puede ver. Si ve más lejos, podría recolectar información más rápidamente y así va a poder tomar mejores decisiones.

Otro problema fue cómo guardar esta información. Desde el punto de vista de la información del mundo, se guardó el punto de origen, el punto de destino y una grilla con las celdas libres y las ocupadas. Luego a medida que el robot avanza va accediendo a la información del mundo en base a los parámetros pre-establecidos y solo puede aprender lo que los sensores le permitan.

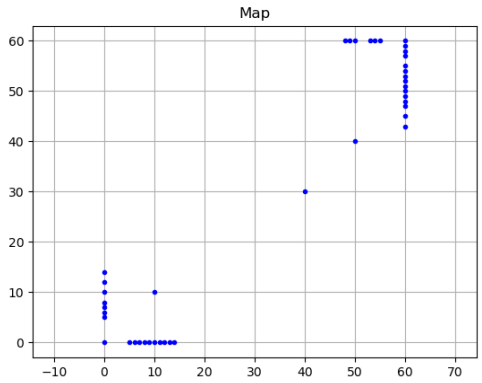

Fig. 4.19: Sensado del Mundo 01

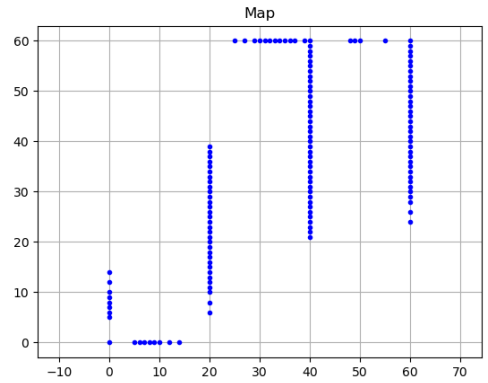

Fig. 4.20: Sensado del Mundo 11

Por ejemplo del sensado del mundo 01 (figura 4.5) el robot en toda su navegación solo llegó a aprender lo que se puede observar en la figura 4.19, el resto del mundo siguió siendo desconocido para él ya que no llegó a "verlo" con sus sensores. En la figura 4.20 podemos también ver lo que el robot pudo percibir del mundo 11 (figura 4.9). Como se puede observar solo aprendió lo necesario para poder navegarlo exitosamente.

4.4. La Capa Deliberativa

El desarrollo de la capa deliberativa de navegación fue orgánico. Desde el comienzo se tuvo un modelo reactivo preponderante, y la capa deliberativa evalúa si debe continuar navegando el modelo reactivo actual, o si llegó a un punto de falla en el cuál se debe cambiar el modelo de navegación utilizado. Una vez elegido el algoritmo de APF para la capa reactiva (ver sección 4.1), en la capa deliberativa se trabajó para evitar sus mínimos locales (ver subsección 2.2.3 y figura 2.18).

En una primera etapa los modelos reactivos utilizados eran APF y FollowPath. Como fue descrito en la sección 4.1, FollowPath es un algoritmo que sigue un camino prefijado por la capa deliberativa hasta conseguir su objetivo o encontrar un obstáculo. Si bien carece de inteligencia, suele ser útil para evitar los mínimos locales que presenta APF y se trabajó en la capa deliberativa para que pudiera establecer objetivos locales que permitieran resolver los mínimos locales.

Cada tipo de navegación reactiva (tanto APF como FollowPath) tiene un método denominado decide_status() que permite evaluar el estado de esa navegación, con esta información se decide la invocación del método propose_new_aproach() de la capa deliberativa.

Si bien desde un punto de vista estricto APF siempre es reactivo, en la descripción de la navegación se distingue entre APF en modo reactivo y deliberativo. Cuando se busca

alcanzar el objetivo global, se lo denominará APF reactivo. Cuando avanza hacia un objetivo local impuesto por la capa deliberativa, se lo denominará APF deliberativo.

Existen 2 métodos de la capa deliberativa que son importantes a la hora de entender cómo se decide la navegación:

- propose_new_aproach(): Este método evalúa toda la información recibida hasta el momento y decide cuál es el mejor camino a seguir. Las posibilidades son: cambiar la navegación, establecer un objetivo local o buscar un nuevo camino no explorado.
- get_next_unknown_goal(): Cuando durante la navegación se llega a un punto en el cuál el camino actual prueba ser infructuoso, se debe buscar más información del mundo. Este método propone un objetivo local que nos permite aprender más sobre el mundo que estamos navegando.

Al usar una navegación primariamente reactiva, se prioriza el alcanzar el objetivo de navegación global. Normalmente el obtener información para tomar mejores decisiones es un objetivo secundario. Pero cuando llegamos a un punto en el que la información acumulada no nos permite alcanzar el objetivo, lo más importante pasa a ser el conseguir nueva información. De esta manera en el método get_next_unknown_goal() se recurre a al algoritmo de búsqueda exhaustiva Astar (ver subsección 2.2.1) con una leve modificación, el objetivo de búsqueda no es el global, si no que pasa a ser el punto menos conocido y más cercano al objetivo de navegación global. Esto permite establecer un objetivo local que tomará el algoritmo FollowPath y una vez alcanzado tendremos más información del mundo para poder buscar un camino al objetivo global.

Si bien el algoritmo APF tiene una función decide_status() que nos permite saber si está enfrentando un mínimo local, no puede evaluar que tipo de mínimo local está enfrentando. Para poder establecer un objetivo local exitoso, en el transcurso de este trabajo se encontró que era importante establecer el tipo de mínimo local. Como se vio en la

sección 4.1 se detectaron 2 tipos (figura 4.1 y 4.2). La capa reactiva tiene la inteligencia para detectar entre estos 2 tipos y proponer un objetivo local que permita superarlos.

Finalmente, la capa deliberativa tiene otra responsabilidad y es la de acumular la información del mundo. Los detalles de esta responsabilidad se van a desarrollar en la siguiente sección.

4.5. Cómo Se Construye La Información Del Mundo

Si bien el robot no tiene un conocimiento del mundo previo, para poder tomar decisiones informadas (en la capa deliberativa) es importante acumular el conocimiento que se gana del ambiente que se recorre. Con la información acumulada del mundo se genera un mapa del mismo (Roßmann et al., 2009) y se decidió agregar además la información de navegación disponible a cada paso.

Se utilizaron técnicas generales y comunes al área de conocimiento (ver subsección 2.2.3) decidiendo utilizar un modelo del mundo discretizado con una matriz homogénea de ocupación o grilla con celdas de igual tamaño (similar a lo que vemos en la figura 2.19). Esta información sólo permitía conocer el mundo y sus obstáculos.

Cuánto ve el robot del mundo que va navegando depende del alcance de los sensores. Si el sensor puede ver a mayor distancia, o con mayor cantidad de grados, tendremos más información sobre el mundo más rápido. Esto permitirá detectar caminos sin salida o situaciones de mínimos locales antes y evitarnos navegar más para poder descubrirlo.

El recorrer el mundo con un sentido particular, nos provee además de información valiosa cuando se deben evaluar (o reevaluar) las decisiones pasadas a la vista de los hallazgos presentes. En particular, cuando se avanza en una dirección que nos permita alcanzar el objetivo, pero en su lugar nos encontramos con un camino sin salida. En estos

casos nos sirve poder evaluar nuevamente las decisiones ya tomadas. De esta manera es útil no solo poseer la información de ocupación, si no también la de navegación.

Si bien el robot no conoce nada del ambiente al arrancar la navegación, sí se define el tamaño del área de navegación a recorrer. Esto es importante porque al comenzar la navegación el mundo se presenta como 100% desconocido, solo a medida que lo recorramos vamos a ir estableciendo partes del mundo no solo como recorridas, si no también como conocidas.

Para poder evaluar caminos que nos permitan evitar obstáculos, se decidió agregar a la información del mundo los potenciales brushfire de cada celda. De esta manera no solo sabemos si está libre una celda, si no también la distancia que tiene a los obstáculos (ver subsección 2.2.4).

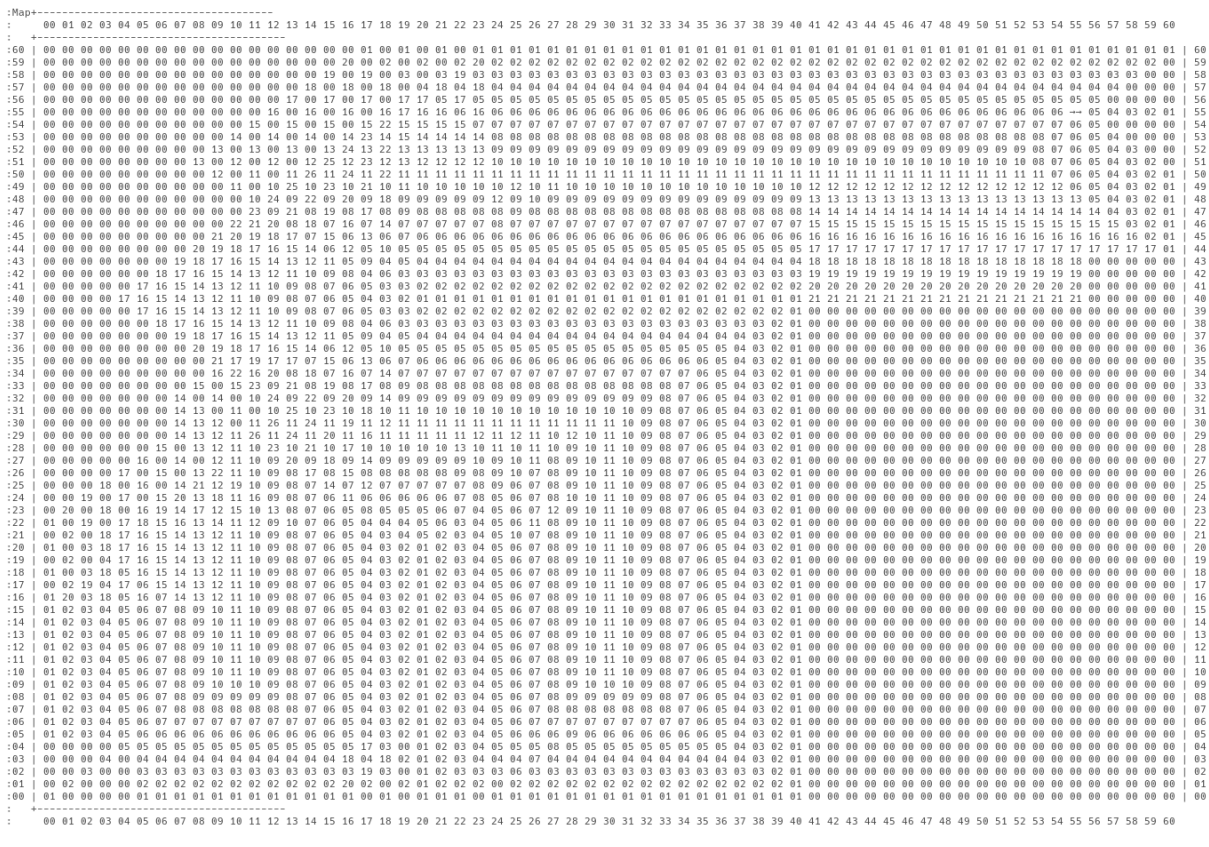

Fig. 4.21: Potenciales Brushfire luego de terminar la navegación del mundo de la fig. 4.13.

En la figura 4.21 se puede observar la información Brush fire acumulada durante la navegación del mundo 21 de la figura 4.13. Las celdas con un valor 00 son desconocidas, las celdas con un valor 01 tienen un obstáculo, todo el espacio libre conocido tiene valores desde 02 en adelante. Cada uno de estos valores expresan la distancia al obstáculo conocido más cercano.

5. CASOS DE EXPERIMENTACiÓN

En este capítulo se trabaja con la solución planteada para demostrar su funcionamiento. En la **introducción** (sección 5.1) se ve como se comenzó a construir una solución, en **ambientes de prueba** (sección 5.2) los ambientes usados para desarrollar las simulaciones. Luego en **simulaciones** (sección 5.3) se ve la simulación de cada mundo, sus particularidades, desarrollo y resultados. Finalmente, en **discusión de resultados** (sección 5.4) se comparan los resultados del algoritmo propuesto con otro de referencia.

5.1 Introducción

Se desarrolla la navegación para distintos ambientes de manera de utilizar una configuración de los sensores y navegación que permita resolver los ambientes de la mejor manera posible. Los ambientes se plantean de manera de demostrar los puntos fuertes y falencias de la combinación de algoritmos planteada como así también su funcionamiento complementario.

Si bien la inspiración inicial fue el trabajo de PythonRobotics (Sakai, 2017), faltaban muchas herramientas. Lo más útil del trabajo fue entender cómo trabajaban distintos módulos de Python y la demostración de navegación APF. Pero para poder realizar este trabajo se debió trabajar en otros algoritmos y modificar también el Astar.

Un área de desarrollo fue crear herramientas que permitieran armar mundos distintos. Algunos de estos mundos estuvieron basados en el trabajo realizado y otros en inspirados en otros papers o la búsqueda de ambientes sin salida.

Para poder crear ambientes se utilizó una combinación de hoja de cálculos, CSV e importación en Python de manera de tener origen, destino y obstáculos en el mismo archivo de datos.

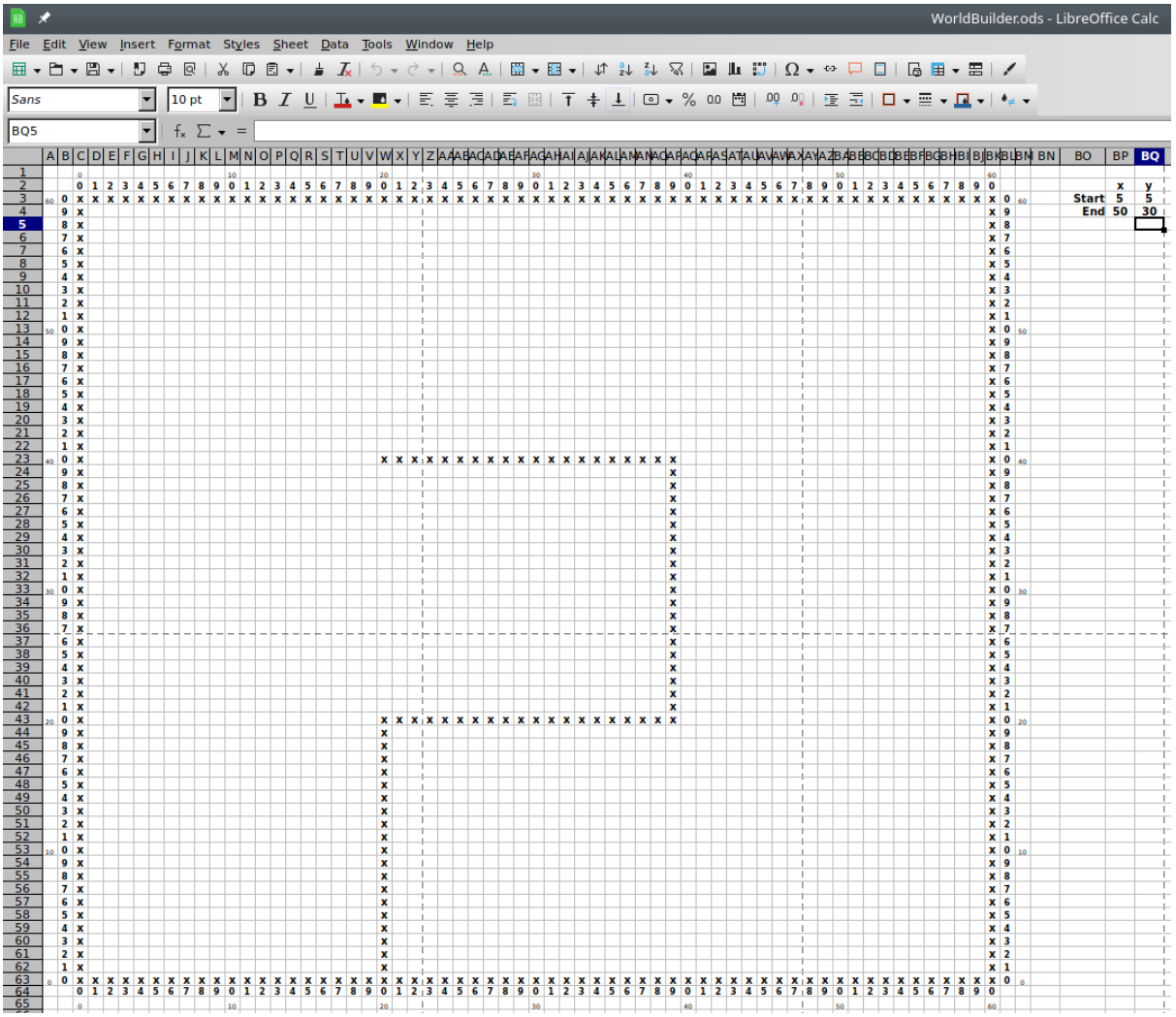

Fig. 5.1: Mundo diseñado en una hoja de cálculo listo para ser exportado

5.2 Ambientes De Prueba

Como se mencionó en la sección 4.3 con los ambientes propuestos se busca en primera instancia demostrar los detalles de implementación del algoritmo realizado y que permita encontrar soluciones completas a cualquier tipo de ambiente (sección 1.2). Si bien no se pueden simular todas las posibilidades, una buena elección debe tener un grado de confianza alto en que podría resolver cualquier tipo de mundo.

Para caracterizar los ambientes y su simulación se tendrán en cuenta los obstáculos, su ubicación, forma y cantidad. Esto permite entender cuál es la complejidad de resolución que va a presentar.

Los ambientes de prueba se plantearon todos con un límite común de 60 celdas de ancho por 60 celdas de alto. Esto permite hacer muchas corridas con configuraciones similares y poder comparar resultados. Se crearon distintos tipos de ambientes, solo el primer grupo es de ambientes abiertos. Luego se utilizarán distintos tipos de ambientes cerrados hasta llegar a un grupo de ambientes irresolubles.

5.2.1 Ambientes Abiertos

En el primer grupo de ambientes vemos obstáculos que primero simplemente deben ser evitados (Mundo 01 figura 5.2 y Mundo 02 figura 5.3), para luego evolucionar en obstáculos que presentan una barrera (Mundo 03 figura 5.4 y Mundo 04 figura 5.5). La diferencia que existe en la dirección del obstáculo presentada en el Mundo 03 y el Mundo 04 fue agregada para ver si alguna podía presentar un mínimo local para el algoritmo APF.

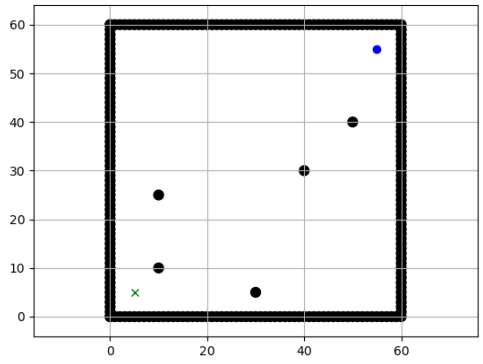

Fig. 5.2: Mundo 01

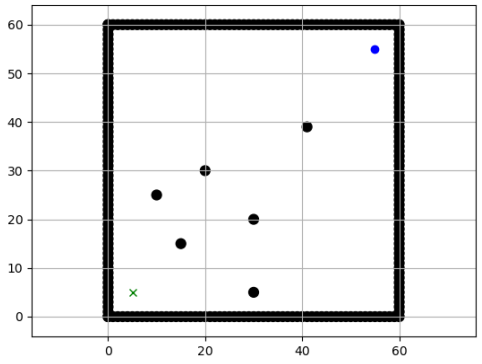

Fig. 5.3: Mundo 02

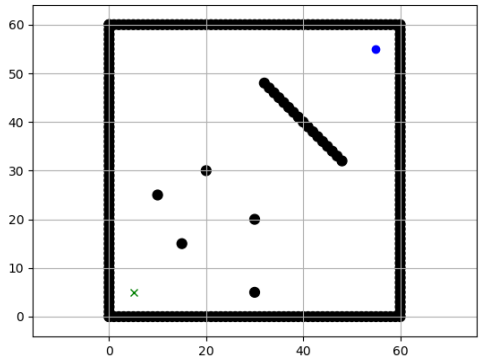

Fig. 5.4: Mundo 03

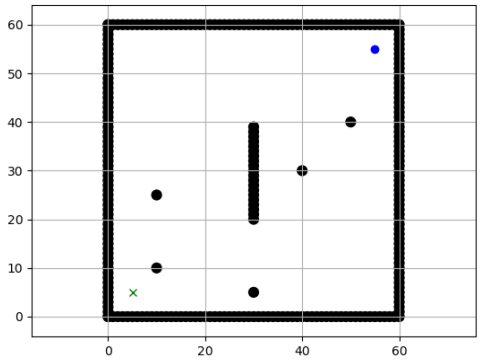

Fig. 5.5: Mundo 04

5.2.2 Ambientes Cerrados

En el segundo grupo se comienza a probar la navegación en ambientes cerrados. El Mundo 11 (figura 5.6) tiene un ejemplo de un mínimo local que afecta la navegación APF (visto en la subsección 2.2.3 figura 2.18). A este mínimo local se lo caracterizó en la sección 4.1 (figura 4.1) como una trampa de tipo 1. En el Mundo 12 (figura 5.7) podemos ver un ejemplo de trampa del tipo 2 (vista en la sección 4.1 figura 4.2). Si bien ambos mundos presentan trampas, cada uno presenta un desafío distinto y requiere una estrategia de solución distinta. Pero ambos mundos requieren algún tipo de capa deliberativa y un segundo algoritmo para poder evitar los problemas de APF.

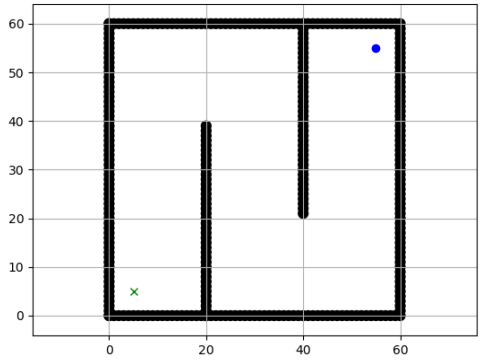

Fig. 5.6: Mundo 11

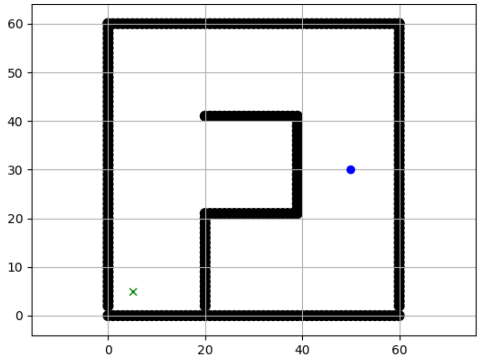

Fig. 5.7: Mundo 12

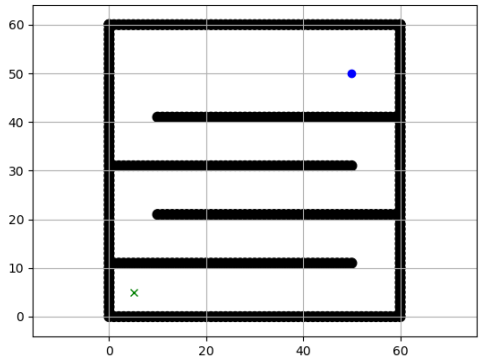

Fig. 5.8: Mundo 13

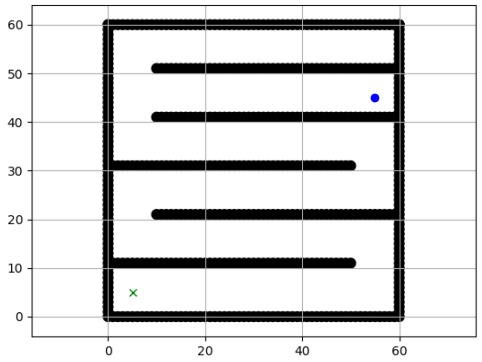

Fig. 5.9: Mundo 14

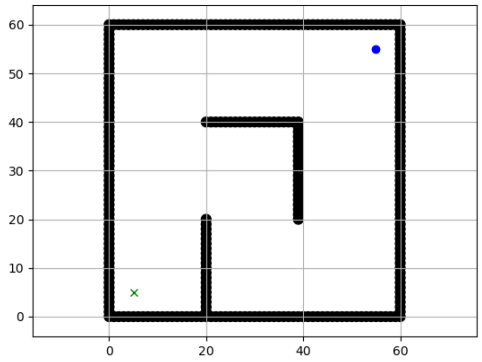

Fig. 5.10: Mundo 15

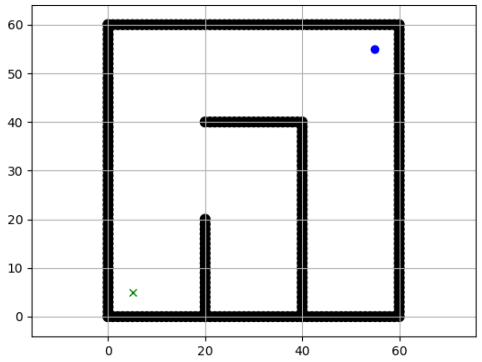

Fig. 5.11: Mundo 21

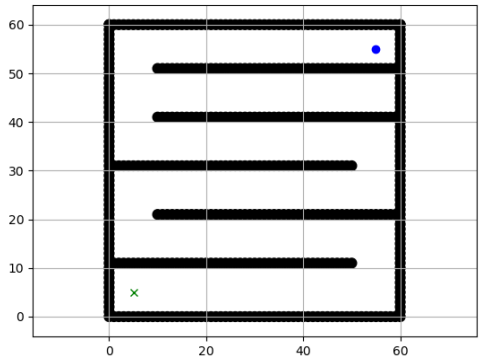

Fig. 5.12: Mundo 22

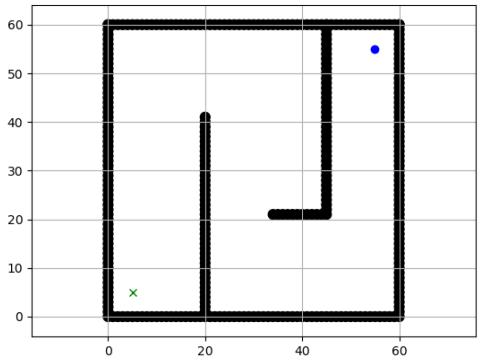

Fig. 5.13: Mundo 23

Los mundos 13, 14 y 22 (figura 5.8, 5.9 y 5.12) tienen una dificultad distinta. No solo presentan mínimos locales, si no que deben usar algún algoritmo que permita alejarse del

objetivo para poder recorrer el terreno, juntar información adicional y permitir avanzar. Son mundos del tipo laberinto y requieren estrategias distintas para poder navegarlos.

Los mundos 15, 21 y 23 (figura 5.10, 5.11 y 5.13) tienen otro tipo de dificultad. Si bien son similares a los utilizados para demostrar las falencias de APF y cómo evitarlas, permiten demostrar los problemas de no conocer el mundo de antemano. Dependiendo de la cantidad de información que se obtenga de la navegación (gracias a sensores con mayores capacidades) podremos tomar mejores decisiones o llevarán muchísimo más esfuerzo por la falta de información y necesidad de recorrerlos exhaustivamente para poder tomar decisiones.

Luego tenemos un grupo de mundos deliberadamente complicados, pero sin ningún objetivo particular más que demostrar que la combinación de algoritmos pueden navegar mundos complejos. Los mundos 24, 31 y 32 (figuras 5.14, 5.15 y 5.16) no tienen otra particularidad que su complejidad. Son situaciones que conociendo los algoritmos involucrados podrían presentar problemas.

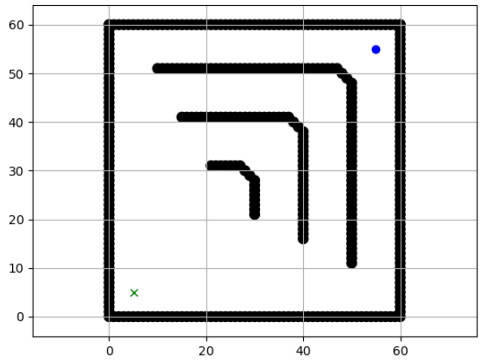

Fig. 5.14: Mundo 24

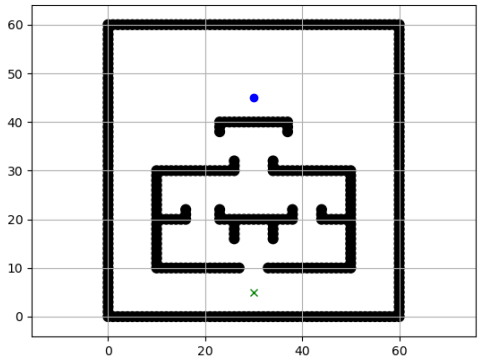

Fig. 5.15: Mundo 31

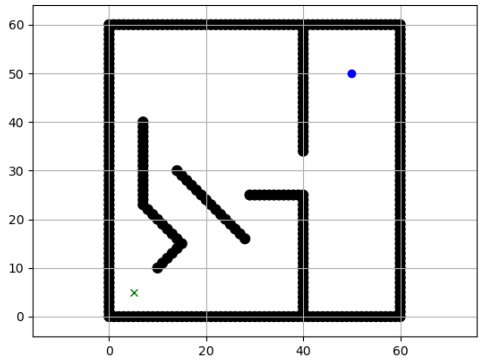

Fig. 5.16: Mundo 32

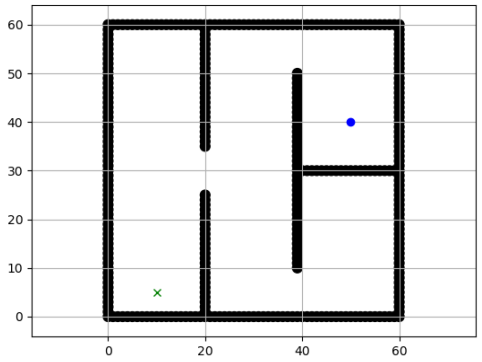

Fig. 5.17: Mundo 33

5.2.3 Ambientes Sin Solución

El mundo 33 (figura 5.17) tiene otro propósito. Si bien presenta una complejidad laberíntica ya explorada, es la base para comenzar con mundos sin solución. Este primer mundo tiene solución, pero el mundo 99 (figura 5.18) es muy similar y no tiene solución.

Otro mundo sin solución es el 97 (figura 5.19). Es mucho más sencillo y rápido llegar a la conclusión de que este mundo no tiene solución, fundamentalmente gracias a la facilidad de mapearlo (exhaustivamente) a medida que se lo recorre.

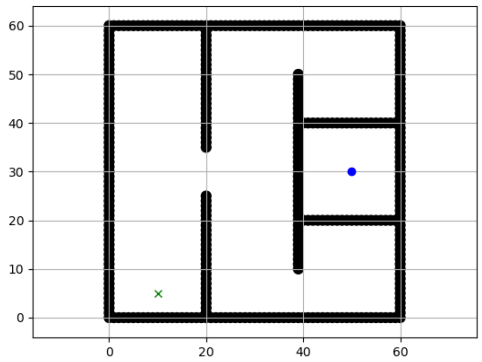

Fig. 5.18: Mundo 99

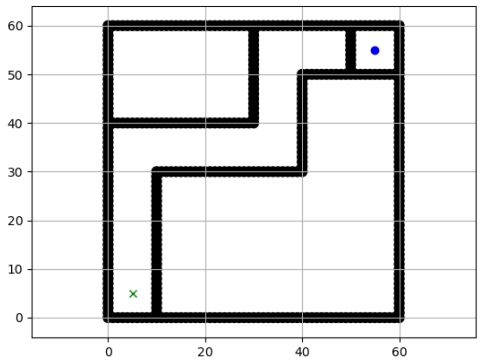

Fig. 5.19: Mundo 97

5.3 Simulaciones

Se evalúa el comportamiento del nuevo algoritmo para cada mundo propuesto con distintos parámetros de los sensores. Se varía el límite de visión y cantidad de pasos del sensor LIDAR. Esto permite en el capítulo de conclusiones hacer un análisis de la mejor configuración posible. Se muestra con la mejor configuración posible el camino realizado. Adicionalmente se compara en cada mundo el resultado del nuevo algoritmo, frente al resultado que tendría Astar. Si bien el algoritmo propuesto no tiene conocimiento del mundo, Astar tendrá toda la información del mundo de antemano. A pesar de esta diferencia, la comparación nos da una idea de eficiencia del nuevo algoritmo en los distintos ambientes.

Se aclaran los siguientes conceptos que se registran para cada simulación:

- Si se pudo alcanzar el **objetivo** propuesto.
- La cantidad de **objetos del mundo**. Esto es importante porque sirve para determinar cuán eficiente es el sensor en relevarlos durante la navegación.
- Los **pasos** que debió realizar el algoritmo para poder llegar al objetivo. Esto nos permitirá comparar su eficiencia con respecto a otro algoritmo.
- Los **pasos Astar** para resolver el mismo ambiente. Esto nos dará una indicación de cuán eficiente fue el algoritmo.
- Para cada simulación se realizan distintas corridas con distintos parámetros del sensor LIDAR y rango de visión. Los **pasos del LIDAR** y **límite de visión** elegidos son los que demostraron una mejor performance. Siempre se elige el menor valor posible que demuestra buenos resultados.

Estos parámetros que se resumen intentan caracterizar cada simulación y ayudan en el siguiente capítulo de conclusiones.

5.3.1 Mundo 1

- Objetivo: Alcanzado
- Objetos del mundo: 247
- Pasos: 54
- Pasos Astar: 53
- Pasos del LIDAR elegidos: 32
- Límite de visión elegido: 30

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
54	42	8	10
54	55	16	10
54	65	32	10
54	67	64	10
54	67	128	10
54	14	8	5
54	42	8	10
54	93	8	20
54	137	8	30

Tabla 1: Corridas del mundo 1

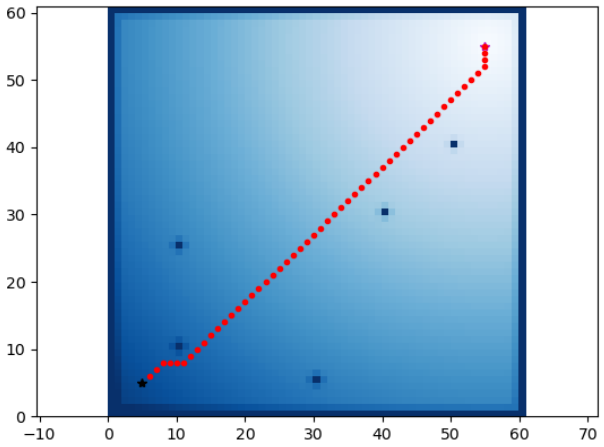

Fig. 5.1: Recorriendo el mundo 1

En este ambiente el algoritmo propuesto se desplaza utilizando navegación APF de manera reactiva durante todo el recorrido.

5.3.2 Mundo 2

- Objetivo: Alcanzado
- Objetos del mundo: 248
- Pasos: 56
- Pasos Astar: 53
- Pasos del LIDAR elegidos: 32
- Límite de visión elegido: 30

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
56	42	8	10
56	58	16	10
56	66	32	10
56	67	64	10
56	67	128	10
56	17	8	5
56	42	8	10
56	98	8	20
56	142	8	30

Tabla 2: Corridas del mundo 2

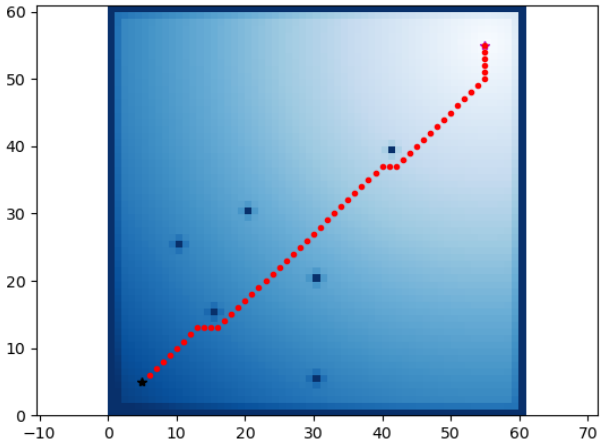

Fig. 5.2: Recorriendo el mundo 2

En este ambiente el algoritmo propuesto se desplaza utilizando navegación APF de manera reactiva durante todo el recorrido.

5.3.3 Mundo 3

- Objetivo: Alcanzado
- Objetos del mundo: 264
- Pasos: 74
- Pasos Astar: 69
- Pasos del LIDAR elegidos: 32
- Límite de visión elegido: 30

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
73	70	8	10
73	87	16	10
73	95	32	10
73	97	64	10
73	97	128	10
73	43	8	5
73	70	8	10
73	120	8	20
73	155	8	30

Tabla 3: Corridas del mundo 3

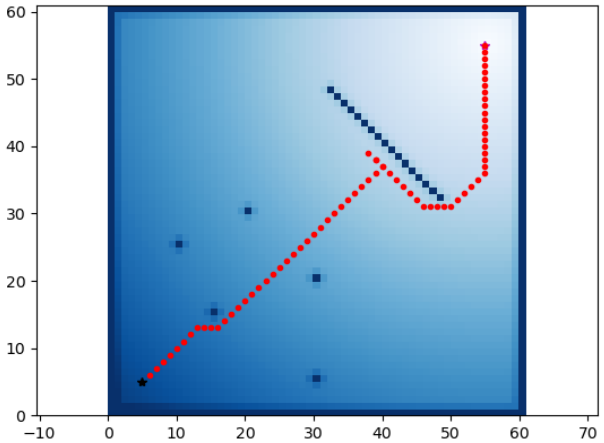

Fig. 5.3: Recorriendo el mundo 3

En este ambiente el algoritmo propuesto se desplaza utilizando navegación APF de manera reactiva hasta alcanzar la posición 39,38. En ese punto el algoritmo deliberativo detecta dos caminos APF posibles con la misma probabilidad. Como en estos casos APF no puede resolver, decide establecer el objetivo secundario 68,9 y comenzar a recorrer utilizando el algoritmo FollowPath. A medida que avanza y recolecta más información la capa deliberativa modifica el objetivo secundario por 50,31 y modifica la dirección de avance. Al lograr evitar el obstáculo en la posición 50,31 la capa deliberativa decide retornar el control a APF en modo reactivo. Continuará navegando de esta manera hasta alcanzar el objetivo planteado.

5.3.4 Mundo 4

- Objetivo: Alcanzado
- Objetos del mundo: 267
- Pasos: 65
- Pasos Astar: 62
- Pasos del LIDAR elegidos: 32
- Límite de visión elegido: 30

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
65	67	8	10
65	81	16	10
65	91	32	10
65	93	64	10
65	93	128	10
65	41	8	5
65	67	8	10
65	120	8	20
65	164	8	30

Tabla 4: Corridas del mundo 4

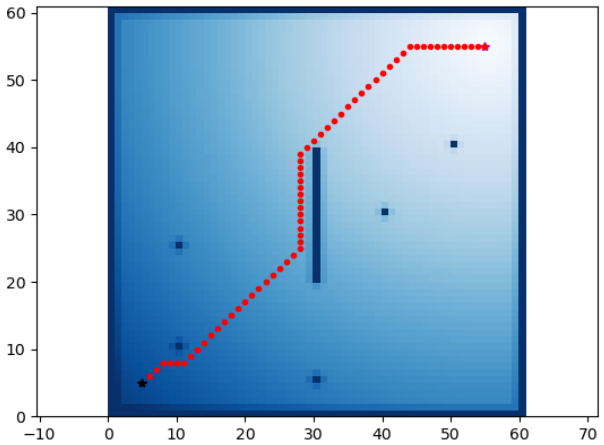

Fig. 5.4: Recorriendo el mundo 4

En este ambiente el algoritmo propuesto se desplaza utilizando navegación APF de manera reactiva durante todo el recorrido.

5.3.5 Mundo 11

- Objetivo: Alcanzado
- Objetos del mundo: 322
- Pasos: 123
- Pasos Astar: 95
- Pasos del LIDAR elegidos: 32
- Límite de visión elegido: 30

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
126	140	8	10
125	159	16	10
125	168	32	10
125	169	64	10
125	169	128	10
129	103	8	5
126	140	8	10
125	214	8	20
123	266	8	30

Tabla 5: Corridas del mundo 11

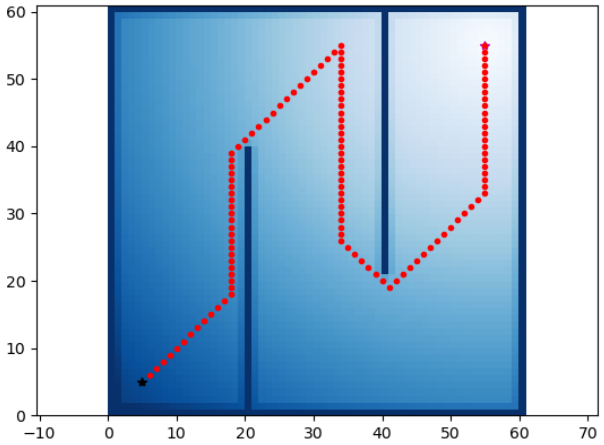

Fig. 5.5: Recorriendo el mundo 11

En este ambiente el algoritmo propuesto se desplaza utilizando navegación APF de manera reactiva hasta la posición 34,55. En ese punto se encuentra con un punto de falla APF (trampa tipo 1) y la capa deliberativa decide utilizar el algoritmo FollowPath utilizando como objetivo local el punto 34,25. Durante el avance en modo FollowPath el objetivo local se va modificando para seguir en línea recta hasta llegar al punto 34,26 donde al detectar el fin del obstáculo se establece el nuevo objetivo local 41,19. Una vez alcanzado este objetivo, la capa deliberativa le devuelve el control al algoritmo APF reactivo, que continuará navegando hasta alcanzar el objetivo global.

5.3.6 Mundo 12

- Objetivo: Alcanzado
- Objetos del mundo: 318
- Pasos: 86
- Pasos Astar: 71
- Pasos del LIDAR elegidos: 8
- Límite de visión elegido: 20

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
120	92	8	10
99	103	16	10
92	110	32	10
92	112	64	10
92	112	128	10
101	57	8	5
120	92	8	10
86	168	8	20
86	239	8	30

Tabla 6: Corridas del mundo 12

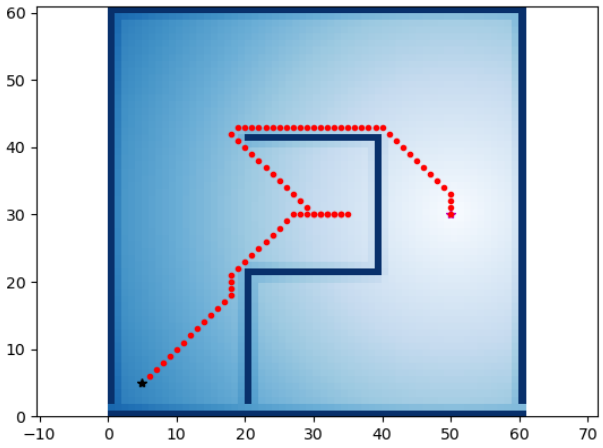

Fig. 5.6: Recorriendo el mundo 12

En este ambiente el algoritmo propuesto se desplaza utilizando navegación APF de manera reactiva hasta la posición 35,30. En este punto la capa deliberativa detecta una trampa del tipo 2 y cambia la navegación por el algoritmo FollowPath. Sin embargo al llegar al punto 30,30 la capa deliberativa se da cuenta de que conoce todo el ambiente que tiene alrededor y no ha logrado avanzar hacía el objetivo, por lo tanto recurre al algoritmo Astar para obtener un objetivo local desconocido. Astar define el nuevo objetivo local como 23,44 y FollowPath avanza hasta alcanzarlo. Una vez alcanzado este objetivo la capa deliberativa detecta situaciones que no podrá resolver APF por sí solo, por lo tanto busca un nuevo objetivo desconocido con Astar: 44,38. Se continúa navegando usando APF pero con un objetivo local establecido por la capa deliberativa (APF deliberativo). Al alcanzar el objetivo local, la capa deliberativa le devuelve el control a APF reactivo que termina la navegación alcanzando el objetivo.

5.3.7 Mundo 15

- Objetivo: Alcanzado
- Objetos del mundo: 300
- Pasos: 87
- Pasos Astar: 73
- Pasos del LIDAR elegidos: 32
- Límite de visión elegido: 30

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
89	102	8	10
89	123	16	10
89	131	32	10
89	132	64	10
89	132	128	10
93	70	8	5
89	102	8	10
89	160	8	20
89	217	8	30

Tabla 7: Corridas del mundo 15

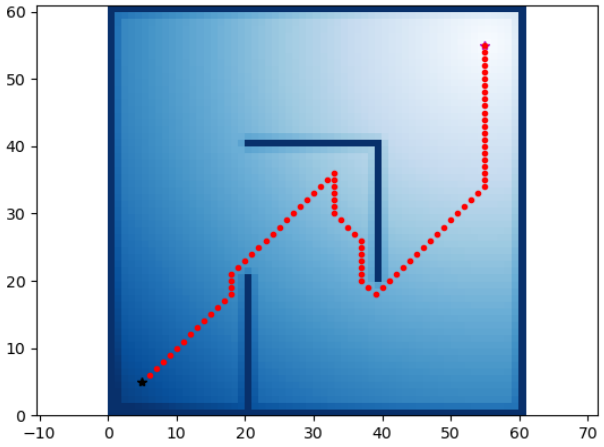

Fig. 5.7: Recorriendo el mundo 15

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 33,35. En este punto encuentra una trampa del tipo 1 y decide cambiar a navegación FollowPath con el objetivo local 33,6. Al llegar al punto 33,30 la capa deliberativa decide con la información disponible buscar el siguiente punto desconocido (42,21) y desplazarse hacía a él usando FollowPath. Al llegar al objetivo local la

capa deliberativa decide pasar nuevamente el control al algoritmo APF reactivo con el objetivo global y navega de esta manera hasta alcanzar el objetivo.

5.3.8 Mundo 21

- Objetivo: Alcanzado
- Objetos del mundo: 324
- Pasos: 93
- Pasos Astar: 73
- Pasos del LIDAR elegidos: 32
- Límite de visión elegido: 30

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
176	156	8	10
167	173	16	10
167	183	32	10
167	185	64	10
167	185	128	10
160	103	8	5
176	156	8	10
146	199	8	20
93	221	8	30

Tabla 8: Corridas del mundo 21

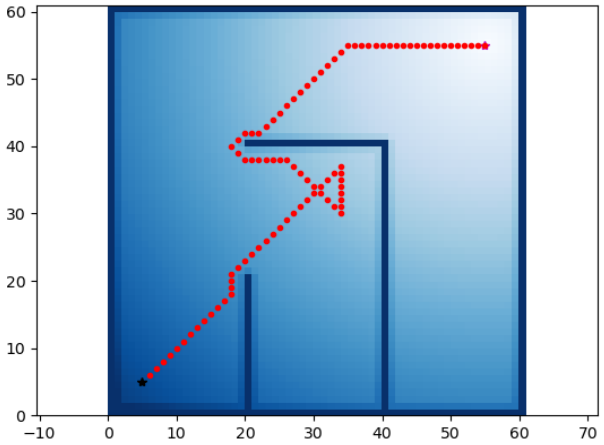

Fig. 5.8: Recorriendo el mundo 21

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 34,37. En este punto encuentra una trampa del tipo 1 y decide cambiar a navegación FollowPath con el objetivo local 34,7. Al llegar al punto 34,30 la capa deliberativa decide con la información disponible buscar el siguiente punto desconocido (23,43) y desplazarse hacía a él usando FollowPath. Al llegar al objetivo local la capa deliberativa decide pasar nuevamente el control al algoritmo APF reactivo con el objetivo global y navega de esta manera hasta alcanzar el objetivo.

5.3.9 Mundo 23

- Objetivo: Alcanzado
- Objetos del mundo: 334
- Pasos: 151
- Pasos Astar: 110
- Pasos del LIDAR elegidos: 32
- Límite de visión elegido: 30

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
170	165	8	10
171	183	16	10
158	192	32	10
158	193	64	10
158	193	128	10
158	127	8	5
155	165	8	10
153	246	8	20
152	297	8	30

Tabla 9: Corridas del mundo 23

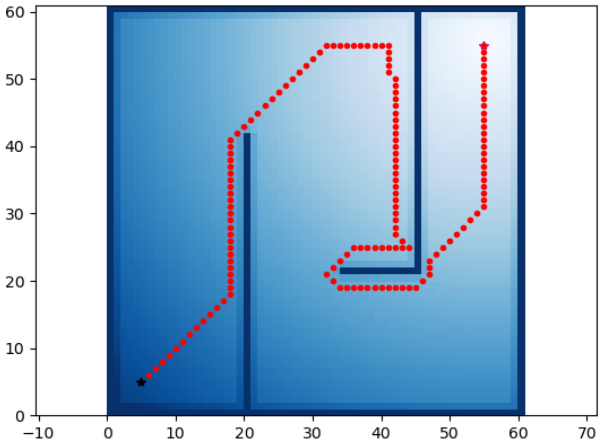

Fig. 5.9: Recorriendo el mundo 23

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 41,55. En este punto encuentra una trampa del tipo 1 y decide cambiar a navegación FollowPath con el objetivo local 41,25. Al llegar al punto 41,22 la capa deliberativa decide con la información disponible buscar el siguiente punto desconocido (45,24) y desplazarse hacía a él usando FollowPath. Al llegar al objetivo local la capa deliberativa decide buscar nuevamente un punto desconocido con Astar y tomar el

nuevo objetivo local 35,16. Comienza a navegar con el algoritmo FollowPath, pero al llegar al punto 36,25 la capa deliberativa decide buscar un nuevo punto desconocido: 48,22. Usando FollowPath alcanza ese objetivo y decide pasar nuevamente el control al algoritmo APF reactivo con el objetivo global. Finalmente navega de esta manera hasta alcanzar el objetivo.

5.3.10 Mundo 13

- Objetivo: Alcanzado
- Objetos del mundo: 442
- Pasos: 232
- Pasos Astar: 222
- Pasos del LIDAR elegidos: 8
- Límite de visión elegido: 10

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
232	311	8	10
230	338	16	10
230	360	32	10
230	369	64	10
230	369	128	10
266	205	8	5
232	311	8	10
230	390	8	20
230	401	8	30

Tabla 10: Corridas del mundo 13

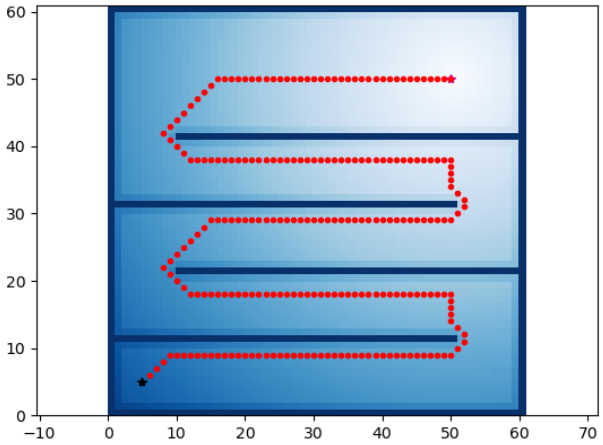

Fig. 5.10: Recorriendo el mundo 13

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 50,18. En este punto encuentra una trampa del tipo 1 y decide cambiar a navegación FollowPath con el objetivo local 40,18. Este objetivo se va extendiendo hasta llegar a la posición 8,22. Al llegar a este la capa deliberativa decide volver el control a la navegación APF reactiva que se desplaza hasta alcanzar el punto 50,38. Pero en este punto encuentra una nueva trampa del tipo 1 y vuelve a modo FollowPath con el objetivo 40,38. Este objetivo se extiende (ya que sigue a la pared que lo bloquea) y al llegar al punto 8,42 la capa deliberativa decide devolverle el control al algoritmo APF reactivo que continuará la navegación hasta alcanzar el objetivo global.

5.3.11 Mundo 14

- Objetivo: Alcanzado
- Objetos del mundo: 492
- Pasos: 245
- Pasos Astar: 227
- Pasos del LIDAR elegidos: 8
- Límite de visión elegido: 10

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
245	339	8	10
245	370	16	10
245	384	32	10
245	385	64	10
245	385	128	10
253	209	8	5
245	339	8	10
245	394	8	20
245	399	8	30

Tabla 11: Corridas del mundo 14

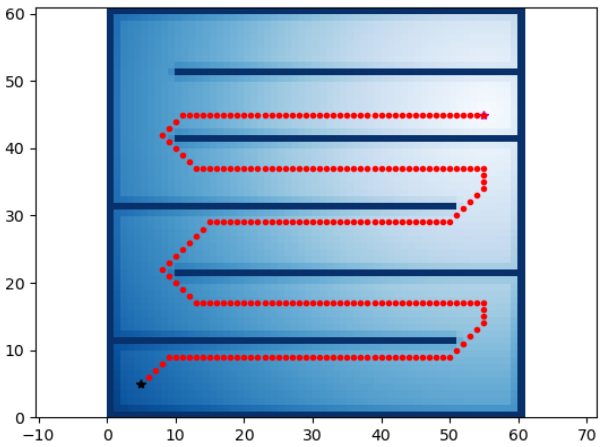

Fig. 5.11: Recorriendo el mundo 14

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 55,17. En este punto encuentra una trampa del tipo 1 y decide cambiar a navegación FollowPath con el objetivo local 45,17. Este objetivo se va extendiendo hasta llegar a la posición 8,22. Al llegar a este la capa deliberativa decide volver el control a la navegación APF reactiva que se desplaza hasta alcanzar el punto 55,37. Pero en este punto encuentra una nueva trampa del tipo 1 y vuelve a modo FollowPath con el

objetivo 45,37. Este objetivo se extiende (ya que sigue a la pared que lo bloquea) y al llegar al punto 8,42 la capa deliberativa decide devolverle el control al algoritmo APF reactivo que continuará la navegación hasta alcanzar el objetivo global.

5.3.12 Mundo 22

- Objetivo: Alcanzado
- Objetos del mundo: 442
- Pasos: 341
- Pasos Astar: 237
- Pasos del LIDAR elegidos: 8
- Límite de visión elegido: 10

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
341	404	8	10
341	443	16	10
341	460	32	10
341	462	64	10
341	462	128	10
349	306	8	5
341	404	8	10
341	460	8	20
341	468	8	30

Tabla 12: Corridas del mundo 22

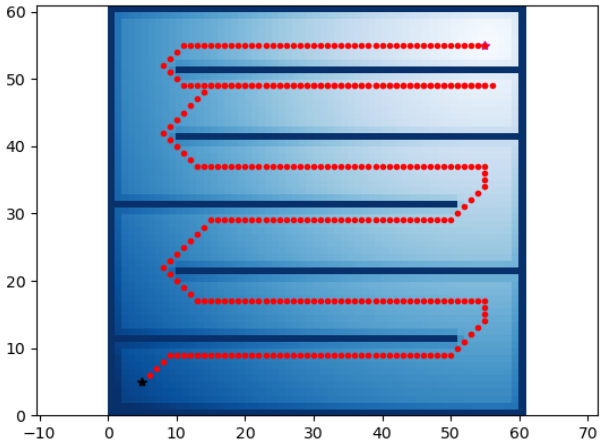

Fig. 5.12: Recorriendo el mundo 22

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 55,17. En este punto encuentra una trampa del tipo 1 y decide cambiar a navegación FollowPath con el objetivo local 45,17. Este objetivo se va extendiendo hasta llegar a la posición 8,22. Al llegar a este la capa deliberativa decide volver el control a la navegación APF reactiva que se desplaza hasta alcanzar el punto 55,37. Pero en este punto encuentra una nueva trampa del tipo 1 y vuelve a modo FollowPath con el objetivo 45,37. Este objetivo se extiende (ya que sigue a la pared que lo bloquea) y al llegar al punto 8,42 la capa deliberativa decide devolverle el control al algoritmo APF reactivo que continuará la navegación hasta alcanzar el punto 56,49. En este punto la capa deliberativa encuentra una nueva trampa del tipo 1 y decide utilizar FollowPath con el nuevo objetivo local 46,49. Continuará navegando y modificando el objetivo local hasta llegar al punto 8,52. En este punto la capa deliberativa decide devolverle el control al algoritmo APF reactivo que navegará hasta alcanzar el objetivo global.

Es interesante notar que el algoritmo FollowPath se comporta adecuadamente para seguir paredes, extendiendo el objetivo local hasta lograr encontrar un camino o en su defecto recurrir al algoritmo Astar. En este caso no fue necesario, pero otras situaciones que veremos más adelante sí es necesario.

5.3.13 Mundo 24

- Objetivo: Alcanzado
- Objetos del mundo: 385
- Pasos: 191
- Pasos Astar: 92
- Pasos del LIDAR elegidos: 8
- Límite de visión elegido: 10

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
191	225	8	10
191	246	16	10
191	256	32	10
191	258	64	10
191	258	128	10
195	183	8	5
191	225	8	10
191	285	8	20
191	307	8	30

Tabla 13: Corridas del mundo 24

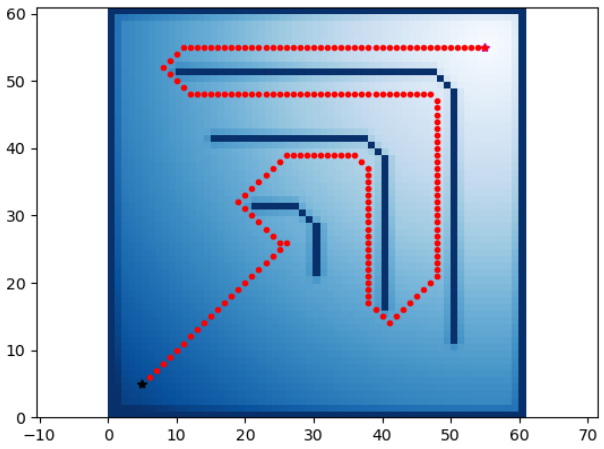

Fig. 5.13: Recorriendo el mundo 24

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 25,26. En este punto encuentra una trampa del tipo 1 y decide cambiar a navegación FollowPath con el objetivo local 16,26. Este objetivo local se va modificando hasta llegar a la posición 19,32. Al llegar a este la capa deliberativa decide volver el control a la navegación APF reactiva que se desplaza hasta alcanzar el punto 38,37. Al llegar a este punto el algoritmo APF se encuentra en un punto de indecisión, tiene 2 caminos posibles con igual probabilidad. La capa deliberativa decide pasar al modo FollowPath con el objetivo 38,27. Este objetivo se extiende (ya que sigue a la pared que lo bloquea) y al llegar al punto 41,14 la capa deliberativa decide devolverle el control al algoritmo APF reactivo que continuará la navegación hasta alcanzar el punto 47,48. Aquí APF tiene nuevamente un punto de indecisión entre dos caminos con igual probabilidad. La capa deliberativa decide utilizar FollowPath con el nuevo objetivo local 37,48. Continuará navegando y modificando el objetivo local hasta llegar al punto 8,52. En este punto la capa deliberativa decide devolverle el control al algoritmo APF reactivo que navegará hasta alcanzar el objetivo global.

Es interesante notar que el algoritmo FollowPath se comporta adecuadamente para seguir paredes, extendiendo el objetivo local hasta lograr encontrar un camino. En este caso el seguir las paredes es suficiente para encontrar un nuevo camino y no necesita recurrir al algoritmo Astar. Este comportamiento en ambientes desconocidos tiene precedentes en otros algoritmos como el tangentBug (ver sección 3.1), parte de la articulación entre APF y FollowPath está inspirada en el comportamiento de tangentBug para seguir paredes.

5.3.14 Mundo 31

- Objetivo: Alcanzado
- Objetos del mundo: 415
- Pasos: 57
- Pasos Astar: 46
- Pasos del LIDAR elegidos: 16
- Límite de visión elegido: 10

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
65	91	8	10
57	100	16	10
57	111	32	10
57	114	64	10
57	114	128	10
65	91	8	10
58	227	8	20
56	323	8	30

Tabla 14: Corridas del mundo 31

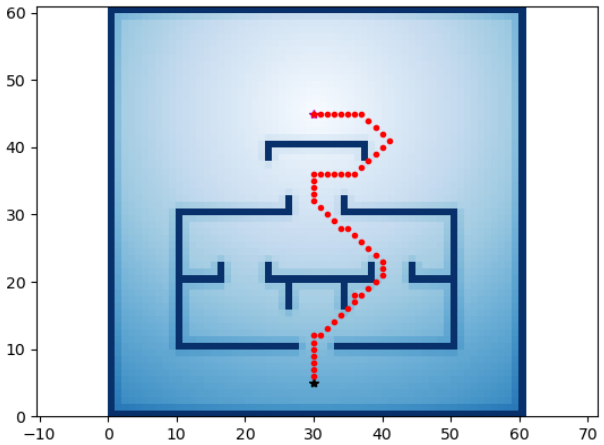

Fig. 5.14: Recorriendo el mundo 31

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 30,45. En este punto encuentra una trampa para APF y decide cambiar a navegación FollowPath con el objetivo local 36,17. Al alcanzar el objetivo local la capa deliberativa decide volver el control a la navegación APF reactiva, que con solo dar un paso (37,18) encuentra una nueva trampa. La capa deliberativa decide utilizar nuevamente FollowPath con el objetivo 40,21. Una vez alcanzado el objetivo local, la capa deliberativa decide devolverle el control al algoritmo APF reactivo que continuará la navegación hasta alcanzar el punto 30,36. Aquí APF encuentra una nueva trampa y la capa

deliberativa decide utilizar FollowPath, primero con el objetivo local 45,36 y una vez evitado el obstáculo seguirá todavía con FollowPath pero objetivo local 41,41. Una vez alcanzado el objetivo local la capa deliberativa decide devolverle el control al algoritmo APF reactivo que navegará hasta alcanzar el objetivo global.

Si bien se eligió mostrar esta corrida por que fue la que tuvo mayor eficiencia, existen otras situaciones en las que la capa deliberativa debió recurrir al algoritmo Astar para poder encontrar un punto desconocido y evitar trampas APF.

5.3.15 Mundo 32

- Objetivo: Alcanzado
- Objetos del mundo: 348
- Pasos: 91
- Pasos Astar: 66
- Pasos del LIDAR elegidos: 8
- Límite de visión elegido: 10

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
91	105	8	10
91	120	16	10
90	137	32	10
90	148	64	10
90	148	128	10
108	59	8	5
91	105	8	10
91	197	8	20
91	247	8	30

Tabla 15: Corridas del mundo 32

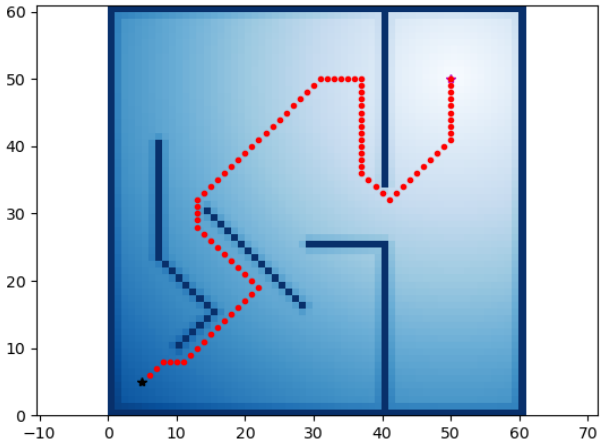

Fig. 5.15: Recorriendo el mundo 32

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 20,21. En este punto encuentra una trampa para APF y decide cambiar a navegación FollowPath con el objetivo local 10,31. En su avance en modo follow la capa deliberativa decide extender primero su objetivo local hasta 7,34 y luego cambiar de dirección (todavía en modo FollowPath) para alcanzar el objetivo local 13,32. Al alcanzarlo la capa deliberativa decide volver el control a la navegación APF reactiva, que avanza hasta la posición 37,50. En este punto encuentra una nueva trampa y la capa deliberativa decide utilizar nuevamente FollowPath con el objetivo 37,40. Todavía en modo FollowPath, en el punto 37,36 cambia de dirección hasta llegar al objetivo local 41,32. Una vez alcanzado el objetivo local, la capa deliberativa decide devolverle el control al algoritmo APF reactivo que continuará la navegación hasta alcanzar el objetivo global.

Si bien este ambiente presenta ciertas complejidades, su resolución puede alcanzarse mediante combinaciones (nuevamente) de APF y FollowPath.

5.3.16 Mundo 33

- Objetivo: Alcanzado
- Objetos del mundo: 351
- Pasos: 64
- Pasos Astar: 60
- Pasos del LIDAR elegidos: 16
- Límite de visión elegido: 10

Pasos	Objetos Encontrados	Pasos LIDAR	Límite Visión
64	64	8	10
62	80	16	10
62	94	32	10
62	105	64	10
62	105	128	10
64	42	8	5
64	64	8	10
64	163	8	20
64	212	8	30

Tabla 16: Corridas del mundo 33

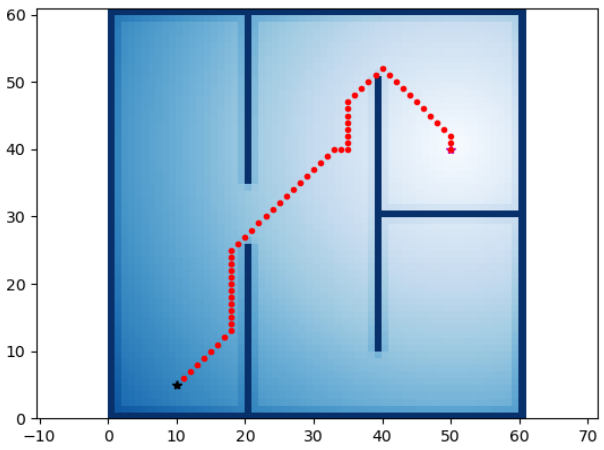

Fig. 5.16: Recorriendo el mundo 33

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 37,40. En este punto encuentra una trampa para APF y decide cambiar a navegación FollowPath. A diferencia de casos anteriores, para este ambiente la capa deliberativa decide utilizar APF para determinar un punto desconocido como objetivo local. El primer objetivo local a alcanzar con FollowPath es 41,46. Las búsquedas de nuevos objetivos locales se suceden hasta llegar al objetivo local 43,48. En este punto la capa deliberativa decide devolverle el control al algoritmo APF reactivo que continuará la navegación hasta alcanzar el objetivo global.

5.3.17 Mundo 99

Este es uno de los casos elegidos en el que el objetivo no se puede alcanzar. Se puede observar que el mundo es bastante similar al anterior caso de experimentación (mundo 33) pero con la diferencia de que el objetivo planteado no es alcanzable.

En este ambiente se intentaron dos corridas, la primera con 8 pasos del LIDAR y un límite de visión de 10. Si bien puede resolver, en este ambiente el tiempo/pasos necesarios para alcanzar la conclusión de que no lo puede resolver es alta (353) frente a una segunda configuración de 32 pasos del LIDAR y un límite de visión de 30. En esta segunda configuración (que se eligió en base al muestreo realizado en las distintas experimentaciones y se elabora más en las conclusiones) se logró concluir la no resolución del mundo en 199 pasos. Que si bien es alto, mirando el camino realizado es razonable.

A continuación se presenta y describe la navegación de 32 pasos del LIDAR con una visión límite de 30.

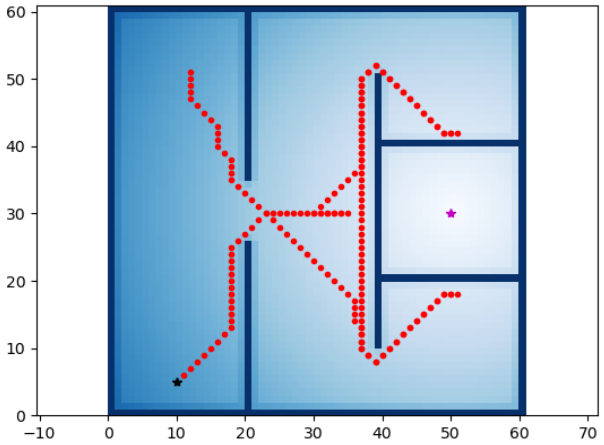

Fig. 5.17: Recorriendo el mundo 99

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 35,30. En este punto encuentra una trampa para APF y decide cambiar a navegación FollowPath con el objetivo local 01,30. Pero sin lograr alcanzarlo la capa deliberativa decide recurrir a Astar (debido a la falta de camino en el mundo conocido) estableciendo el objetivo local 41,50. En este punto la capa deliberativa decide devolverle el control al algoritmo APF reactivo que llegará al punto 51,42. Aquí nuevamente la capa deliberativa recurre a Astar y pasa a navegar utilizando FollowPath hasta llegar al punto 41,10. En este punto la capa deliberativa le pasa el control a APF reactivo que navega hasta llegar al punto 51,18. Al llegar aquí la capa deliberativa recurre nuevamente a Astar y navega utilizando FollowPath hasta el punto 15,44 donde decide pasarle el control a APF “deliberativo” con el objetivo local 12,51. Al alcanzar este objetivo la capa deliberativa concluye que no hay camino posible para alcanzar el objetivo y termina la navegación.

5.3.18 Mundo 97

Si bien este tampoco tiene solución, es un poco más sencillo llegar a esta conclusión. En este caso también se decide utilizar 32 pasos del LIDAR con una visión límite de 30 ya que se concluyó que es la más óptima en la mayoría de los casos.

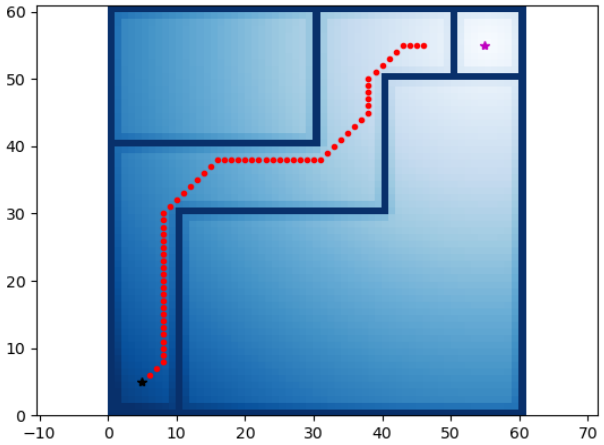

Fig. 5.18: Recorriendo el mundo 97

En este ambiente el algoritmo propuesto se desplaza mediante el algoritmo APF en forma reactiva hasta llegar a la posición 46,55. Al alcanzar este punto la capa deliberativa concluye que no hay camino posible para alcanzar el objetivo y termina la navegación.

Lo interesante de este caso es que debido a la configuración del mundo y la navegación “con propósito” de APF se logra mapear toda la información necesaria para que el algoritmo Astar pueda concluir que el mundo no tiene solución.

5.4. Discusión De Resultados

Dentro del área de navegación se encontró que el foco común de los trabajos es mostrar una mejora en la navegación gracias a una modificación del algoritmo principal planteado (más sobre esto en la sección A.2.1.1).

Si comparamos los resultados de la navegación del algoritmo propuesto sin conocimiento del mundo, con una navegación con conocimiento previo del mundo (como Astar) veremos que los resultados en la mayoría de los casos son similares.

	Pasos		Dif.	%
	Híbrido	A*		
Mundo1	54	53	1	2%
Mundo2	56	53	3	6%
Mundo3	74	69	5	7%
Mundo4	65	62	3	5%
Mundo11	123	95	28	29%
Mundo12	86	71	15	21%
Mundo15	87	73	14	19%
Mundo21	93	73	20	27%
Mundo23	151	110	41	37%
Mundo13	232	222	10	5%
Mundo14	245	227	18	8%
Mundo22	341	237	104	44%
Mundo24	191	92	99	108%
Mundo31	57	46	11	24%
Mundo32	91	66	25	38%
Mundo33	64	60	4	7%

Tabla 17: Comparación de resultados

Solo en 2 mundos (22 y 24) la diferencia es grande. En ambos casos la única forma de reducir el camino sería conocer el mundo en forma previa. Ambos presentan “trampas” que solo pueden descartarse al haberse recorrido. Pero fuera de estos 2 casos particulares, el desconocimiento previo del mundo no tiene un impacto grande en la cantidad de pasos necesarios para encontrar un camino. Esto se debe a que se logra una arquitectura que tiene una detección temprana de problemas y que se complementan bien sus partes.

Si bien la navegación en los mundos 22 (sección 5.3.9), 23 (sección 5.3.12) y 32 (sección 5.3.15) son buenos, la navegación APF informada tiene una mejor performance. Viendo la navegación quizás se podría hacer algo con respecto a la visión o decisión de la capa deliberativa para mejorar la performance del algoritmo presentado en esos casos.

La gran mayoría de los mundos no pueden ser resueltos con APF. Solo el mundo 1 y el 2 pueden navegar de manera 100% reactiva. Todo el resto de los ambientes recurren a la capa deliberativa, probando la necesidad de una arquitectura híbrida y el éxito de la arquitectura planteada. Los ambientes más sencillos pueden ser navegados con un subconjunto de los algoritmos incluidos, pero a medida que la complejidad del ambiente aumenta es evidente que la combinación de todos es necesaria para poder resolver cualquier tipo de ambiente. El uso de una estrategia mixta (reactiva+deliberativa) es fundamental a la hora de poder navegar por cualquier tipo de ambiente desconocido.

Por otro lado, si bien el algoritmo Astar podría resolver todos los mundos, la imposición de navegar por un ambiente desconocido hace su elección como único algoritmo inviable. Pero en los resultados se observa que la información recolectada tanto por la navegación APF como FollowPath permiten generar la información que Astar necesita para complementar la solución.

Otra cosa que queda demostrada en los resultados es la correcta articulación entre los algoritmos. La forma en que se complementan permiten al algoritmo propuesto lograr una navegación autónoma desconociendo a priori el ambiente en todos los ambientes planteados. Todos los caminos planteados son convergentes y permiten alcanzar el objetivo planteado. Ninguno de los algoritmos individualmente pueden alcanzar una resolución de todos los casos.

Es importante destacar que muchos de los algoritmos o propuestas de algoritmos revisadas no podrían resolver el tipo de mundos que se plantearon en este trabajo:

1. En el estudio detallado sobre los algoritmos tipo bug (Ng, J. 2005) todos los mundos a resolver son del tipo abiertos y con resolución. Estos algoritmos carecen de la capacidad de declarar un mundo sin solución.
2. El trabajo de Lee, et. al. (2018) tiene problemas similares, y dentro del trabajo de tesis presente tampoco se pudieron reproducir los resultados del paper. De alguna

manera el uso de APF y FollowPath intentó reproducir los resultados de NP-APF pero a medida que se complicaron los mundos esa mejora dejó de dar resultados.

3. Un trabajo más cercano en su propuesta al presente es Zhang et al. (2007). Si bien este trabajo ofrece una solución completa, lo hace con un conocimiento completo del mundo (necesario para aplicar descomposición de celdas aproximadas). El presente trabajo construye la información del mundo a medida que lo explora.
4. Si bien el trabajo de descomposición de celdas aproximadas para ambientes desconocidos (Dugarjav et al., 2013) tiene objetivos similares, la falta de un algoritmo reactivo que le permita recoger información del mundo a medida que avanza lo hace menos eficiente.

Todos estos resultados y comparaciones demuestran que la combinación propuesta es una mejora superadora de los trabajos revisados por el presente tesista dentro del área planteada de investigación y trabajo.

Finalmente, como se puede observar en las tablas de cada simulación, se exploró la relación entre el radio de visión del robot, la cantidad de pasos del LIDAR y la eficiencia del algoritmo actual. Se encontró que en algunos casos cuanto más amplia es la visión y pasos, mejores resultados se obtienen. Pero esto no es una máxima, depende mucho del tipo de mundo. Y si bien en algunos casos se observó una mejora, en la mayoría de los casos no tuvo una gran influencia.

Para poder determinar que los mundos no tienen solución, primero debieron ser recorridos completamente. Solo al llegar a un punto donde Astar nos informa el conocimiento pleno del mundo podemos concluir que no hay camino posible al punto de destino.

6. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

En este capítulo se presentan los **resultados finales y las conclusiones** del trabajo (sección 6.1), y las **futuras líneas de investigación** (sección 6.2) qué de este trabajo se desprenden.

6.1. Conclusiones

En la sección 1.2 se plantearon objetivos específicos de la tesis, revisando estos objetivos a vistas del trabajo realizado:

1. Presentar el área de conocimiento, su contexto y particularidades: Esto se realizó en las secciones 1 y 2 de la tesis. Si bien el área es amplia, se buscó hacer foco en los temas que eran necesarios para entender el trabajo.
2. Encontrar algoritmos candidatos para formar una arquitectura híbrida: Se buscaron y evaluaron algoritmos candidatos para formar la arquitectura híbrida. Estos algoritmos fueron revisados en la sección 3 separadamente y en la sección 4 ya en conjunto. En particular la sección 4 intenta transmitir la arquitectura y el por qué de la solución propuesta.
3. Identificar distintos tipos de ambientes que permitan validar la arquitectura: En la sección 4.2 se explica la necesidad de identificar ambiente para validar la navegación completa y el área de vacancia dentro del área en este sentido. En la sección 5.2 se presentan estos mundos proponiendo una clasificación que ayudó a dirigir el trabajo de tesis desde una complejidad baja a una cada vez mayor.
4. Ensayar distintas combinaciones de algoritmos para lograr el objetivo general: Si bien se ensayaron diferentes combinaciones detalladas en la sección 4.1, la búsqueda del presente trabajo es de una solución. Por lo tanto la sección 5.3 muestra los resultados obtenidos con la arquitectura elegida.

5. Comparar los resultados de la arquitectura planteada con otros algoritmos: En la sección 5.4 se comparan los resultados obtenidos con un algoritmo que tiene un conocimiento previo del mundo (Astar). Demostrando que la arquitectura presentada tiene un performance similar, en la mayoría de los casos, a la que tiene conocimiento del mundo.

Con respecto al objetivo general planteado en 1.1, se puede decir que se logró crear un algoritmo novedoso que logra navegar de forma autónoma cualquier tipo de ambientes desconocidos e invariantes en el tiempo. Adicionalmente, y como demuestran las pruebas 5.3.17 y 5.3.18 si no existe un camino el sistema declara al ambiente sin solución. Esto lo hace en una cantidad de pasos limitados, no debe hacer una examinación exhaustiva del mundo para concluirlo.

En el proceso de desarrollo de la tesis trajo consigo aprendizajes sobre las necesidades de la capa reactiva, la necesidad de uso de Astar, cómo articular entre los algoritmos y la necesidad de recolectar información del mundo, . Esto fue desarrollado más exhaustivamente en la sección 4.1, pero a modo de resumen:

- La elección del mejor algoritmo reactivo posible en calidad de planeamiento y recolección de información es importante a la hora de conseguir buenos resultados. En este caso APF demostró cumplir con este propósito y al tener falencias conocidas es sencillo para la capa deliberativa saber cuándo cambiar el algoritmo que controla el avance.
- Si bien el sistema resultante usa Astar en ciertas situaciones, la mayoría de los mundos (todos excepto el mundo 12 y 23 con solución) fueron resueltos sin la ayuda de Astar. En la siguiente sección (futuras líneas de investigación) se exploran las excepciones.
- La utilización de un proceso exhaustivo (como Astar) es necesario a la hora de decidir si el objetivo es inalcanzable. Cualquier arquitectura propuesta va a necesitar un algoritmo exhaustivo para poder determinar la no resolución de manera efectiva.

- Dentro de este trabajo la decisión de pasar de un proceso reactivo al deliberativo se realiza teniendo en cuenta la información disponible, las debilidades del algoritmo deliberativo elegido y la cantidad de información presente al momento de tomar la decisión. Dependiendo de la cantidad de información el algoritmo deliberativo puede elegir buscar más información del ambiente que permita una decisión más informada.
- Es importante la información acumulada de la navegación para poder tomar buenas decisiones. Si bien con un conocimiento del mundo completo alcanza con saber si en una posición existe un obstáculo o no, en el caso de ir descubriendo el mundo a medida que se navega, es importante también saber cómo se obtuvo esa información.

6.2. Futuras Líneas De Investigación

El trabajo realizado permitió llegar a alcanzar los objetivos propuestos, pero hay mucho más para explorar en esta área. En esta sección se presentan posibles líneas de mejora sobre el trabajo actual.

Un área a explorar sería mejorar la capa reactiva. El modelo propuesto (sección 4.1.7) utiliza un algoritmo reactivo que se podría combinar con otro del tipo exploratorio. La solución propuesta hoy utilizando el algoritmo FollowPath (sección 4.1.4) funciona, pero se podrían explorar posibilidades de mejoras. Este algoritmo exploratorio debería contar con una capa de lógica difusa cuyo objetivo será recolectar la información necesaria para tomar la decisión de pasar a un algoritmo exhaustivo. La búsqueda de esta capa debe tener un balance entre la información a recolectar y el costo de navegar para conseguirla. En ciertas situaciones puede ser mejor pasar a la acción y en el trayecto, seguir recolectando información.

Si bien en el presente trabajo se realiza un intento de clasificar los mundos, hay mucho trabajo por recorrer en esta área. En la sección 4.2 se revisaron ciertos trabajos

incipientes que proponen algunos conjuntos de mundos de prueba. Sería bueno crear un catálogo de mundos, clasificarlos y relacionarlos con los distintos algoritmos. El tener una base estandarizada para la evaluación de algoritmos de navegación puede ayudar a encontrar mejores combinaciones y enriquecer el área de estudio. La propuesta de clasificación y ambientes de prueba (sección 5.2) no fueron el foco, sino una necesidad.

En este trabajo se decidió utilizar APF por diversas razones (ver sección 4.1.2), pero se podría explorar el uso del algoritmo TangentBug también para la capa reactiva. Si bien es menos eficiente en ciertas circunstancias (ver sección 3.2, 4.1.1, 4.1.4 y 5.1.13), el algoritmo TangentBug no tiene el problema de los mínimos locales. Por lo tanto, en ciertas situaciones podría representar una ventaja al presente uso de FollowPath (sección 4.1.4).

Cabe destacar la importancia de investigar el funcionamiento de la combinación propuesta en este trabajo para que articule con un algoritmo de visión artificial. La simulación de un LIDAR dentro del trabajo (sección 4.3) es una forma de acotar el área de investigación. Pero en la discusión de resultados (sección 5.4) se revisa el uso de LIDAR y la necesidad de explorar más la influencia de la información recolectada en las decisiones que toma el robot. Ya sea mediante el uso de cámaras como sensores, como así también con algoritmos de reconocimiento de patrones que permitan detectar formas se podría adquirir información más rica y mejorar las decisiones.

Como se mencionó en la sección 5.4 se podría ver si mejoras en la capa deliberativa y un análisis más exhaustivo de la información recolectada no podría ayudar a mejorar la performance de la solución presentada cuando se compara con los resultados de Astar informado. En la arquitectura actual parece haber un sesgo de decisión que a pesar de mejorar la visión, no permite mejorar la navegación. Si bien los resultados son buenos, intuitivamente parecen mejorables. Un futuro trabajo podría enfocarse en mejorar los resultados de la navegación en los mundos 22, 23, 24 y 32. Cuando el alcance (límite de visión) y la precisión (pasos del LIDAR) aumentan se debería poder tomar mejores

decisiones. O por otro lado con una mejor evaluación de la información recolectada quizás se puedan mejorar esos resultados.

7. REFERENCIAS

Abafogi, M., Durdu, A., & Akdemir, B. (2018). A new approach to mobile robot navigation in unknown environments. 2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 1–5.

Ackerman, E. (2018, August 06). Boston Dynamics Is Getting Ready to Produce Lots of SpotMinis. Retrieved from https://spectrum.ieee.org/automaton/robotics/industrial-robots/boston-dynamics-spotminis

Agirrebeitia, J., Avilés, R., de Bustos, I. F., & Ajuria, G. (2005). A new APF strategy for path planning in environments with obstacles. Mechanism and Machine Theory, 40(6), 645–658. https://doi.org/10.1016/j.mechmachtheory.2005.01.006

Ajeil, F. H., Ibraheem, I. K., Sahib, M. A., & Humaidi, A. J. (2020). Multi-objective path planning of an autonomous mobile robot using Hybrid PSO-MFB optimization algorithm. Applied Soft Computing, 89, 106076. doi:10.1016/j.asoc.2020.106076

Akka, K., & Khaber, F. (2018). Mobile robot path planning using an improved ant colony optimization. International Journal of Advanced Robotic Systems, 15(3), 172988141877467. https://doi.org/10.1177/1729881418774673

Algabri, M., Mathkour, H., Ramdane, H., & Alsulaiman, M. (2015). Comparative study of soft computing techniques for mobile robot navigation in an unknown environment. Computers in Human Behavior, 50, 42–56. https://doi.org/10.1016/j.chb.2015.03.062

An, H., Hu, J., & Lou, P. (2021). Obstacle avoidance path planning based on improved APF and RRT. 2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), 1028–1032.

Anwar, A., & Raychowdhury, A. (2020). Autonomous Navigation via Deep Reinforcement Learning for Resource Constraint Edge Nodes Using Transfer Learning. IEEE Access, 8, 26549-26560. doi:10.1109/access.2020.2971172

Arkin, R. C., & MacKenzie, D. C. (1994). Planning to behave: A hybrid deliberative/reactive robot control architecture for mobile manipulation. https://smartech.gatech.edu/handle/1853/22224

Basili, V. R. (1993). The experimental paradigm in software engineering. In Experimental Software Engineering Issues: Critical Assessment and Future Directions (pp. 1–12). Springer Berlin Heidelberg.

Bhattacharya, P. & Gavrilova, M. L. (2008). Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path. IEEE Robotics Automation Magazine, 15(2), 58-66. doi:10.1109/MRA.2008.921540

Bounini, F., Gingras, D., Pollart, H., & Gruyer, D. (2017). Modified artificial potential field method for online path planning applications. 2017 IEEE Intelligent Vehicles Symposium (IV), 180–185.

Casner, S. M., Hutchins, E. L., & Norman, D. (2016). The challenges of partially automated driving. Communications of the ACM, 59(5), 70–77.

Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., & Thrun, S. (2005). Principles of robot motion: Theory, algorithms, and implementation. Cambridge, MA: MIT Press.

Creswell, J. W. (2015). Educational research: Planning, conducting, and evaluating quantitative and qualitative research (6th edition).

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271. https://doi.org/10.1007/bf01386390

Dugarjav, B., Lee, S.-G., Kim, D., Kim, J. H., & Chong, N. Y. (2013). Scan matching online cell decomposition for coverage path planning in an unknown environment. International Journal of Precision Engineering and Manufacturing, 14(9), 1551–1558. https://doi.org/10.1007/s12541-013-0209-5

Feiner, L. (2021, June 13). Amazon details new warehouse robots, “Ernie” and “Bert.” CNBC. https://www.cnbc.com/2021/06/13/amazon-details-new-warehouse-robots-ernie-and-bert.html

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4(1), 23–33. https://doi.org/10.1109/100.580977

Franceschini, F., Galetto, M., Maisano, D., Mastrogiacomo, L., & Pralio, B. (2011). Indoor GPS (iGPSTM). In Distributed Large-Scale Dimensional Metrology (pp. 23–35). Springer London.

Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (2015). Path planning and trajectory planning algorithms: A general overview. In Motion and Operation Planning of Robotic Systems (pp. 3–27). Springer International Publishing.

Guzewich, Lemmon, Smith, Martínez, Vicente-Retortillo, Newman, and Mier (2019). Mars Science Laboratory Observations of the 2018/Mars Year 34 Global Dust Storm. Geophysical Research Letters, 46(1), 71-79. doi:10.1029/2018gl080839

Hart, P., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107. https://doi.org/10.1109/tssc.1968.300136

Heiden, E., Palmieri, L., Bruns, L., Arras, K. O., Sukhatme, G. S., & Koenig, S. (2021). Bench-MR: A motion planning benchmark for wheeled mobile robots. IEEE Robotics and Automation Letters, 6(3), 4536–4543. https://doi.org/10.1109/lra.2021.3068913

Hong, J., Tang, K. & Chen, C. (2017). Obstacle avoidance of hexapod robots using fuzzy Q-learning. En 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1-6). doi:10.1109/SSCI.2017.8280907

Hornyak, T. (2018, October 30). The world's first humanless warehouse is run only by robots and is a model for the future. CNBC. https://www.cnbc.com/2018/10/30/the-worlds-first-humanless-warehouse-is-run-only-by-robots.html

Hossen, J., Sayeed, S. & Iqbal, A. P. (2015). A Modified Hybrid Fuzzy Controller for Real-Time Mobile Robot Navigation. Procedia Computer Science, 76, 449-454. 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015). doi:10.1016/j.procs.2015.12.307

Hossian, R., Martinez, G., Olivera, A., & VERONICA. (2014). 9789871871247: Robótica de los navegadores : un enfoque desde las tecnologías inteligentes - AbeBooks - GARCIA MARTINEZ, RAMON / HOSSIAN, ALEJANDRO / OLIVERA, VERONICA: 9871871244. NUEVA LIBRERIA. https://www.abebooks.com/9789871871247/Rob%C3%B3tica-navegadores-enfoque-desde-tecnolog%C3%ADas-9871871244/plp

Hsiao, F. and Lee, P. (2017). Autonomous indoor passageway finding using 3d scene reconstruction with stereo vision. In 2017 Computing Conference, pages 279--285. doi:10.1109/SAI.2017.8252115

Iswanto, I., Wahyunggoro, O. & Cahyadi, A. I. (2016). Quadrotor Path Planning Based on Modified Fuzzy Cell Decomposition Algorithm. TELKOMNIKA (Telecommunication Computing Electronics and Control), 14(2), 655. doi:10.12928/telkomnika.v14i2.2989

Jin, J., Gokhale, V., Dundar, A., Krishnamurthy, B., Martini, B., and Culurciello, E. (2014). An efficient implementation of deep convolutional neural networks on a mobile coprocessor. In 2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS), pages 133--136. doi:10.1109/MWSCAS.2014.6908370

Jung, T., Lim, J., Bae, H., Lee, K. K., Joe, H., and Oh, J. (2018). Development of the humanoid disaster response platform drc-hubo+. IEEE Transactions on Robotics, 34(1):1--17. doi:10.1109/TRO.2017.2776287

Kalra, N., Ferguson, D., & Stentz, A. (2009). Incremental reconstruction of generalized Voronoi diagrams on grids. Robotics and Autonomous Systems, 57(2), 123–128. https://doi.org/10.1016/j.robot.2007.01.009

Kameyama, N., & Hidaka, K. (2017). A sensor-based exploration algorithm for autonomous map generation on mobile robot using kinect. 2017 11th Asian Control Conference (ASCC). doi:10.1109/ascc.2017.8287213

Kamon, I., Rimon, E., & Rivlin, E. (1998). TangentBug: A range-sensor-based navigation algorithm. The International Journal of Robotics Research, 17(9), 934–953. https://doi.org/10.1177/027836499801700903

Kamon, I., Rivlin, E., & Rimon, E. (2002). A new range-sensor based globally convergent navigation algorithm for mobile robots. Proceedings of IEEE International Conference on Robotics and Automation, 1, 429–435 vol.1.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research, 5(1), 90–98. https://doi.org/10.1177/027836498600500106

Kim, J., & Woo, S. H. (2018). Reference Test Maps for Path Planning Algorithm Test. International Journal of Control, Automation and Systems, 16(1), 397-401. doi:10.1007/s12555-017-0059-5

Krogh, B., & Thorpe, C. (2005). Integrated path planning and dynamic steering control for autonomous vehicles. Proceedings. 1986 IEEE International Conference on Robotics and Automation, 3, 1664–1669.

Kul, G., Özyer, T., & Tavli, B. (2014). IEEE 802.11 WLAN based real time indoor positioning: Literature survey and experimental investigations. Procedia Computer Science, 34, 157–164. https://doi.org/10.1016/j.procs.2014.07.078

Le, A. V., Prabakaran, V., Sivanantham, V., & Mohan, R. E. (2018). Modified A-star algorithm for efficient coverage path planning in Tetris inspired self-reconfigurable robot with integrated laser sensor. Sensors (Basel, Switzerland), 18(8), 2585. https://doi.org/10.3390/s18082585

Lee, D., Jeong, J., Kim, Y. H. & Park, J. B. (2017). An improved artificial potential field method with a new point of attractive force for a mobile robot. En 2017 2nd International Conference on Robotics and Automation Engineering (ICRAE) (pp. 63-67). doi:10.1109/ICRAE.2017.8291354

Lee, Y.-T., Chiu, C.-S., & Kuo, I.-T. (2017). Fuzzy wall-following control of a wheelchair. 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), 1–6.

Lei, X., Zhang, Z., & Dong, P. (2018). Dynamic path planning of unknown environment based on deep reinforcement learning. Journal of Robotics, 2018, 1-10. doi:10.1155/2018/5781591

Lueck, S. (2020, May 5). How to win AWS DeepRacer - Axel Springer Tech - Medium. Axel Springer Tech. https://medium.com/axel-springer-tech/how-to-win-aws-deepracer-ce15454f594a

Luo, C., & Yang, S. X. (2008). A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. IEEE

Transactions on Neural Networks, 19(7), 1279–1298. https://doi.org/10.1109/tnn.2008.2000394

Mautz, R. (2009). The challenges of indoor environments and specification on some alternative positioning systems. 2009 6th Workshop on Positioning, Navigation and Communication, 29–36.

Moore, S. K. (2019). 3 New Chips to Help Robots Find Their Way Around. IEEE Spectrum. Recuperado desde https://spectrum.ieee.org/automaton/semiconductors/processors/3-new-chips-to-help-robots-find-their-way-around

Morando, A., Gershon, P., Mehler, B., & Reimer, B. (2021). A model for naturalistic glance behavior around Tesla Autopilot disengagements. Accident; Analysis and Prevention, 161(106348), 106348. https://doi.org/10.1016/j.aap.2021.106348

Murphy, R. R. (2001). Introduction to AI robotics. The Industrial Robot, 28(3), 266–267. https://doi.org/10.1108/ir.2001.28.3.266.1

Na, Y.-K., & Oh, S.-Y. (2003). Autonomous Robots, 15(2), 193–206. https://doi.org/10.1023/a:1025597227189

Ng, J. (2005). A Practical Comparison of Robot Path Planning Algorithms given only Local Information [University of Western Australia]. https://robotics.ee.uwa.edu.au/theses/2005-Navigation-Ng.pdf

Nolfi, S. (2002). Power and the limits of reactive agents. Neurocomputing, 42(1–4), 119–145. https://doi.org/10.1016/s0925-2312(01)00598-7

Ollero Baturone, A. (2001). Robótica : manipuladores y robots móviles. Marcombo / Alfaomega.

Paliwal, S. S., & Kala, R. (2018). Maximum clearance rapid motion planning algorithm. Robotica, 36(6), 882–903. https://doi.org/10.1017/s0263574718000127

Pallas, J. M. A., & Jimenez Villa, J. (2019). Metodos de Investigacion Clinica Y Epidemiologica (5th ed.). Elsevier. https://www.elsevier.com/books/metodos-de-investigacion-clinica-y-epidemiologica/argimon-pallas/978-84-9113-007-9

Papadimitriou, C. H. (1977). The Euclidean travelling salesman problem is NP-complete. Theoretical Computer Science, 4(3), 237-244. doi:10.1016/0304-3975(77)90012-3

Patle, B., Babu L, G., Pandey, A., Parhi, D., & Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. Defence Technology, 15(4), 582-606. doi:10.1016/j.dt.2019.04.011

Poorva, A., Gautam, R., & Kala, R. (2018). Motion Planning for a Chain of Mobile Robots Using A* and Potential Field. Robotics, 7(2), 20. doi:10.3390/robotics7020020

Roßmann, J., Krahwinkler, P., & Bücken, A. (2009). Mapping and navigation of mobile robots in natural environments. In Advances in Robotics Research (pp. 43–52). Springer Berlin Heidelberg.

Rostami, S. M. H., Sangaiah, A. K., Wang, J., & Liu, X. (2019). Obstacle avoidance of mobile robots using modified artificial potential field algorithm. EURASIP Journal on Wireless Communications and Networking, 2019(1). https://doi.org/10.1186/s13638-019-1396-2

Sakai, A. (2017-2021). AtsushiSakai/PythonRobotics. Retrieved April 11, 2021, from https://github.com/AtsushiSakai/PythonRobotics

Sedighi, S., Nguyen, D., & Kuhnert, K. (2019). Guided hybrid a-star path planning algorithm for valet parking applications. 2019 5th International Conference on Control, Automation and Robotics (ICCAR). doi:10.1109/iccar.2019.8813752

Sharma, K., & Doriya, R. (2020). Path planning for robots: an elucidating draft. International Journal of Intelligent Robotics and Applications, 4(3), 294–307. https://doi.org/10.1007/s41315-020-00129-0

Shitsukane, A., Cheriuyot, W., Otieno, C., & Mvurya, M. (2018). A Survey on Obstacles Avoidance Mobile Robot in Static Unknown Environment. International Journal of Computer (IJC), 28(1), 160-173. Recuperado de https://www.ijcjournal.org/index.php/InternationalJournalOfComputer/article/view/1161

Shitsukane, A., Cheriuyot, W., Otieno, C. & Mvurya, M. (2018). A Survey on Obstacles Avoidance Mobile Robot in Static Unknown Environment. International Journal of Computer (IJC), 28(1), 160-173. Recuperado desde http://ijcjournal.org/index.php/InternationalJournalOfComputer/article/view/1161

Singh, N. H., Devi, S. S., & Thongam, K. (2020). Modified artificial potential field approaches for mobile robot navigation in unknown environments. In Advances in Intelligent Systems and Computing (pp. 319–328). Springer Singapore.

Smisek, J., Jancosek, M., & Pajdla, T. (2013). 3D with Kinect. In Consumer Depth Cameras for Computer Vision (pp. 3–25). Springer London.

Song, J., & Gupta, S., ε* An Online Coverage Path Planning Algorithm. IEEE Transactions on Robotics, vol. 34, no. 2, pp. 526-533, April 2018, doi: 10.1109/TRO.2017.2780259.

Spong, M. W. (2006). Robot modeling and control. The Industrial Robot, 33(5), 403–403. https://doi.org/10.1108/ir.2006.33.5.403.1

Teli, T. A., & Wani, M. A. (2021). A fuzzy based local minima avoidance path planning in autonomous robots. International Journal of Information Technology, 13(1), 33–40. https://doi.org/10.1007/s41870-020-00547-0

Tschirschnitz, M. V., Ghosh, D., Narayanan, K., Ram, A., Wagner, M. & Honkote, V. (2019). Systems, apparatus, and methods for robot swarm coordination.

Wahabi, A. E., Baraka, I. H., Hamdoune, S., & Mokhtari, K. E. (2020). Design of a Mini Robot for the Automation of 3D Winding Machines Axes and Self-correction by Artificial Vision Using Deep Learning. Advances in Intelligent Systems and Computing Advanced Intelligent Systems for Sustainable Development (AI2SD'2019), 4, 210-223. doi:10.1007/978-3-030-36674-2_23

Walter, W. G. (1950). An Imitation of Life. 182(5), 42-45. Reports experiments with two autonomous, electronic "turtles" — Elmer & Elsie — equipped with various sensors and simple control circuitry which wander around their environment and interact with one another. doi:10.1038/scientificamerican0550-42

Wang, N., Gao, Y., Zheng, Z., Zhao, H., & Yin, J. (2018). A Hybrid Path-Planning Scheme for an Unmanned Surface Vehicle. 2018 Eighth International Conference on Information Science and Technology (ICIST). doi:10.1109/icist.2018.8426161

Wang, S., Wu, Z., & Zhang, W. (2019). An overview of SLAM. In Proceedings of 2018 Chinese Intelligent Systems Conference (pp. 673–681). Springer Singapore.

Wen, J., Zhang, X., Bi, Q., Pan, Z., Feng, Y., Yuan, J., & Fang, Y. (2021). MRPB 1.0: A unified benchmark for the evaluation of mobile robot local planning approaches. 2021 IEEE International Conference on Robotics and Automation (ICRA), 8238–8244.

Wong, C., Yang, E., Yan, X., & Gu, D. (2017). Adaptive and intelligent navigation of autonomous planetary rovers — A survey. 2017 NASA/ESA Conference on Adaptive Hardware and Systems (AHS). doi:10.1109/ahs.2017.8046384

Wu, P., Cao, Y., He, Y., & Li, D. (2017). Vision-based robot path planning with deep learning. In Lecture Notes in Computer Science (pp. 101–111). Springer International Publishing.

Xi, W., Ou, Y., Peng, J. & Yu, G. (2017). A new method for indoor low-cost mobile robot SLAM. En 2017 IEEE International Conference on Information and Automation (ICIA) (pp. 1012-1017). doi:10.1109/ICInfA.2017.8079050

Yan, B., Chen, T., Zhu, X., Yue, Y., Xu, B., & Shi, K. (2020). A Comprehensive Survey and Analysis on Path Planning Algorithms and Heuristic Functions. Advances in Intelligent Systems and Computing Intelligent Computing, 581-598. doi:10.1007/978-3-030-52249-0_39

Zhang, H., Zhang, C., Yang, W., and Chen, C. (2015). Localization and navigation using qr code for mobile robot in indoor environment. In 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 2501--2506. doi:10.1109/ROBIO.2015.7419715

Zhang, L., Kim, Y. J., & Manocha, D. (2007). A hybrid approach for complete motion planning. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 7–14.

APÉNDICE

Esta sección tiene una primera parte (A.1) que hace una revisión del código generado en el desarrollo de la tesis. La segunda parte (A.2) es una revisión sistemática que se agregó posteriormente al desarrollo de la tesis.

A.1. Revisión del código generado

Debido a lo extenso que es y la cantidad de archivos que hay, todo el código creado para las simulaciones, corridas y herramientas se encuentra en un repositorio público de github (https://github.com/ibonelli/pathplanning). Este tesista entiende que es una mejor forma de presentarlo. En este apéndice se mencionan algunas características generales de organización, partes y generalidades del código.

A.1.1. Generalidades

Para el desarrollo de las simulaciones se buscó algún trabajo que tuviera los algoritmos de navegación y búsqueda ya desarrollados. Se encontraron diversas fuentes, pero el trabajo PythonRobotics de Sakai, 2017 realizado en Python fue el que pareció la mejor fuente tanto de investigación (para ver el comportamiento real de los diferentes algoritmos), como también un punto de comienzo para realizar las simulaciones.

El código realizado se encuentra separado en herramientas, clases que implementan la navegación y el código para simulaciones. Adicionalmente, el código incluye los mundos elegidos y distintos scripts para correr esas simulaciones de forma orquestada.

A.1.2. Herramientas

- WorldBuilder.ods : Planilla de cálculos que permite crear mundos.
- show_world.py : Traduce los mundos de WorldBuilder a formato CSV
- Las simulaciones guardan resultados en archivos JSON. Para poder mostrar esta información de manera gráfica se creó:
    - show_limits.py : Permite ver los límites del sensor LIDAR
    - show_map.py : Muestra el mapa que aprendió el robot durante su navegación.
    - show_path.py : Muestra el camino recorrido por el robot.

A.1.3. Simulaciones

- Scripts para corridas en batch: run*.sh
- Mundos creados: world*.csv
- main.py : Programa principal para correr los mundos. Crea los objetos de navegación y va registrando que sucede.
- main_astar.py : Programa de navegación Astar. Fue utilizado para comparar los resultados del algoritmo propuesto, frente a un algoritmo con conocimiento del mundo.

A.1.4. Navegación

- navApfNavigation.py : Implementación del algoritmo de navegación APF
- navFollowPath.py : Implementación del algoritmo FollowPath
- navBrushfire.py : Generación del mapa Brushfire
- navDeliverative.py : Implementación de la capa deliberativa
- navLidar*.py : Procesamiento LIDAR del mundo
- navData.py : Clase que maneja el almacenamiento de los datos

A.1.5. Estadísticas sobre el código generado

A continuación se presentan algunos indicadores sobre el código generado para poder correr las simulaciones. A la derecha tenemos las líneas de código de cada archivo:

Archivo	En blanco	Comentarios	Líneas de código
./navDeliverative.py	91	128	688
./navBrushfire.py	37	62	385
./main.py	31	29	227
./navLidarLimit.py	26	38	173
./navAstar.py	46	53	168
./navLidar.py	15	40	161
./navApfNavigation.py	34	23	152
./navMap.py	23	3	118
./navData.py	13	7	101
./navTrapNavigation.py	11	33	99
./main_astar.py	15	7	73
./navFollowPath.py	16	15	61
./navGraf.py	9	2	61
./navLidarPoint.py	8	1	47
./config.py	3	9	32
./tools/show_world.py	15	24	40
./tools/show_world_fromCSV.py	8	9	33
./tools/plt_polar.py	6	4	19
./tools/calc_win_from_limits.py	8	7	18
./tools/show_limits.py	8	7	16
./tools/show_map.py	8	7	15
./tools/show_path.py	9	7	15

Archivo	En blanco	Comentarios	Líneas de código
./tools/spyral_iteration.py	1	1	12
./run.sh	4	1	19
./run_worlds.sh	6	4	49
./run_nopath.sh	7	1	34
./cleanup.sh	0	0	5

A.2. Revisión sistemática de la literatura

Esta revisión incluye trabajos realizados al principio y durante el desarrollo de la tesis. Pero como la revisión no formó parte del plan original de tesis y no se incluyó en la primera versión. Adicionalmente al momento de la escritura de este anexo hay información que ya no está disponible. Por la falta de material a esta altura del trabajo no se puede documentar una revisión sistemática formal, pero sí se puede entender el camino de desarrollo que tuvo la tesis y los trabajos más relevantes que la orientaron.

En una planilla de cálculo online se disponibiliza los datos recopilados durante el trabajo:

docs.google.com/spreadsheets/d/1bRAeyXV2pvOPabGOExEZ6USdEk02bnvj2SWzypUGf_4/

Esta planilla de cálculo sólo incluye los papers que pasaron un primer filtro. Se revisaron muchos otros trabajos que no se incluyen en esa planilla de cálculo. Si (una vez revisado) el trabajo no aportaba a la línea de investigación en curso, no se incluyó en la recopilación. Lo que si se incluye aquí es el camino de investigación, y en la planilla de cálculo todos los trabajos que pasaron el primer filtro.

A.2.1. Distintos objetivos de investigación durante la revisión

En el proceso de revisión sistemática hubo diferentes etapas. Cada etapa tuvo objetivos de investigación alineados con el objetivo general, pero con diferente foco. A continuación se presenta una lista de las etapas y sus objetivos:

1. **Caracterización del área de estudio:** Entender el área de estudio, el estado del arte, las líneas de investigación, el uso de sensores y la obtención de información del ambiente.
2. **La necesidad de una navegación híbrida**: Se buscaron trabajos previos enfocados en la navegación híbrida que demostraran tanto la posibilidad de navegar un ambiente de manera completa como también un área de vacancia.
3. **Selección de algoritmos**: Que pudieran interactuar en conjunto.
4. **Buscar un marco de prueba**: La búsqueda de un marco de prueba en el área que fuera aceptado y se pudiera utilizar era algo deseado.
5. **Alternativas de navegación**: Para superar los obstáculos encontrados durante el desarrollo de la solución y alcanzar una navegación completa.

A.2.1.1. Caracterización del área de estudio

El objetivo inicial fue entender el área de estudio, el estado del arte y las líneas de investigación. Si bien se revisaron más papers, los que formaron la base del estudio inicial se presentan a continuación:

Preguntas de Investigación	¿Qué tipos de trabajos existen de navegación?
Protocolo de Búsqueda	robot navigation "unknown environment"
Criterios de Selección	Que tuvieran trazabilidad y permitieran caracterizar el área de estudio
Trabajos	● Quadrotor Path Planning Based on Modified Fuzzy Cell Decomposition Algorithm ● An improved artificial potential field method with a new point of attractive force for a mobile robot ● A sensor-based exploration algorithm for autonomous map generation on mobile robot using Kinect ● Obstacle avoidance of hexapod robots using fuzzy Q-learning ● A Survey on Obstacles Avoidance Mobile Robot in Static Unknown Environment
Conclusiones	● El área de trabajo era válida

	● Los trabajos normalmente se centran en un algoritmo de navegación particular o una mejora sobre él. ● Existen muy pocos trabajos en español sobre el tema. ● No se encontraron comparaciones sistemáticas o exhaustivas en el área. ● Existen distintos enfoques dentro del área de navegación como la visión artificial, evitar obstáculos, uso de redes neuronales y fuzzy logic.

Luego de esta primera investigación se continuó expandiendo el área de investigación buscando poder caracterizar el área de estudio. El trabajo PythonRobotics (Sakai, 2017) ya mencionado ayudó a caracterizar el área y entender las posibilidades. No solo por el código si no por su clasificación de los algoritmos:

- Localización
- Mapeo
- Localización y mapeo simultáneo (SLAM)
- Planeo de caminos
- Seguimiento de caminos

Un área importante omitida en esta lista es la visión artificial. Si bien existen estudios centrados en este tema, dentro del área de planeamiento de caminos se suelen encuadrar dentro de SLAM. Las técnicas de visión artificial se usan para identificar la ubicación del robot dentro del mapa y para evitar obstáculos. La visión artificial es parte del área de estudio, pero como se puede encuadrar dentro de las herramientas de sensado y ubicación espacial (sección 2.1.2) no se realizó una investigación separada.

Otros trabajos importantes en esta primer etapa fueron las tesis de James Ng:

- A Practical Comparison of Robot Path Planning Algorithms given only Local Information (2005)
- An Analysis of Mobile Robot Navigation Algorithms in Unknown Environments (2010)

Si bien estos trabajos tenían un foco muy cerrado en los algoritmos del tipo “bug”, tenían una revisión clara de resultados y comparaciones numéricas. La gran mayoría de los trabajos del área carecen de una caracterización numérica fuerte y limitan sus comparaciones a algoritmos de mismo tipo.

La falencia fundamental de estos algoritmos es el fuerte sesgo reactivo que tienen y la difícil articulación con otros algoritmos. Si bien tienen ventajas, a priori el algoritmo APF se vió como un mejor candidato y los resultados fueron buenos.

Otro hallazgo importante de esta etapa fue la forma de demostrar resultados del área de estudio del planeamiento de camino. Si bien se encontraron algunas excepciones, la gran mayoría de los trabajos revisados muestran el aporte al campo estableciendo un algoritmo base con falencias y una evolución/mejora del mismo dentro de un marco de pruebas que demuestre tanto las falencias como el aporte del algoritmo modificado. Hay muchos ejemplos dentro de la investigación documental, pero solo para mencionar algunos: Akka & Khaber (2018), Zhang et al. (2007) y Luo & Yang (2008)

A.2.1.2. La necesidad de una navegación híbrida

En la primera etapa se realizaron pruebas de concepto con el algoritmo APF (sección 2.2.3) que intentaban buscar lograr una navegación completa reactiva (buscando resultados similares al trabajo de James Ng). Esto llevó a un entendimiento más amplio del área de estudio y acompañado por el director de tesis una búsqueda más amplia de trabajos previos enfocados en la navegación híbrida (sección 2.3).

Preguntas de Investigación	¿Cuán frecuente es el uso de una arquitectura híbrida? ¿Qué tipos de combinaciones conviene usar? ¿Hay clasificaciones de arquitecturas que se puedan usar?
Protocolo de Búsqueda	Hybrid navigation
Criterios de Selección	Que tuvieran trazabilidad y aportaran datos para poder responder las preguntas de investigación

Trabajos más relevantes de esta etapa	● Desarrollo de una Técnica de Navegación Híbrida para Robots Móviles en Ambientes con Obstáculos Predefinidos ● A hybrid approach for complete motion planning ● A Hybrid Genetic Algorithm for Mobile Robot Shortest Path Problem ● Guided Hybrid A-star Path Planning Algorithm for Valet Parking Applications ● A Hybrid Path-Planning Scheme for an Unmanned Surface Vehicle (Wang et al., 2018) ● Motion Planning for a Chain of Mobile Robots Using A* and Potential Field (Poorva et al., 2018)
Conclusiones	● Enfocarse en soluciones completas y recurrir a soluciones híbridas no es lo más común dentro del área. Se encontró una mayor cantidad de trabajos con foco en hacer mejoras (medibles) a algoritmos particulares. ● Incluso los algoritmos más básicos (como A-Star) son considerados para arquitecturas híbridas. ● Si bien existen arquitecturas híbridas sugeridas en los libros sobre el área, no hay un patrón definido en los trabajos. ● Todavía no existe dentro del área un marco que permita comparar los algoritmos entre ellos. Habitualmente las comparaciones son dentro de familias de algoritmos muy acotadas (sección 6.2)

Esta revisión y el trabajo previo demostraron la necesidad de una arquitectura híbrida. Si bien no sé encontraron combinaciones de arquitecturas híbridas candidatas, si se logró verificar el uso de estrategías híbridas para poder llegar a resultados de navegación completa. Los resultados de esta etapa definieron la validez de una búsqueda de arquitectura candidata híbrida que resolviera mundos de manera completa.

A.2.1.3. Selección de algoritmos base reactivo

Existen muchos algoritmos a considerar y las combinaciones pueden ser muchas. Pero no todas las combinaciones son eficientes. Todos los algoritmos tienen falencias, lo

importante es poder detectarlas y encontrar otros algoritmos que puedan ayudar a superarlas.

Preguntas de Investigación	¿Qué algoritmos de navegación existen? ¿Cuáles son sus falencias? ¿Cómo se puede articular entre los distintos algoritmos?
Protocolo de Búsqueda	APF and local minima detection, SLAM, unknown environment navigation, bug algorithms, robot navigation algorithms
Criterios de Selección	Que presentaran casos de navegación que permitieran entender las falencias, fortalezas y posibilidades de articulación de los algoritmos.
Trabajos más relevantes de esta etapa	● Principles Of Robot Motion (MIT 2005) Choset ● Robótica: Manipuladores y robots móviles (2005) BATURONE ● Introduction to Autonomous Mobile Robots (MIT 2004) SIEGWART & NOURBAKHSH ● Robotics - Modelling Planning and control (Springer 2009) Siciliano, Sciavicco, Villani & Oriolo
Conclusiones	● La lista de algoritmos es muy amplia ● Todos los algoritmos tienen falencias, no siempre son complementarias, además tienen foco y características que no permiten siempre su combinación ● La mejor forma de estructurar el mundo e interactuar entre algoritmos es la descomposición en celdas.

Si bien se revisaron papers y trabajos, la mejor fuente para esta etapa fueron los libros de robótica con capítulos específicos sobre navegación. Hay mucho trabajo realizado sobre las fortalezas y debilidades de los algoritmos individualmente. Los algoritmos elegidos han sido investigados ampliamente lo que ayuda a su caracterización, el foco del trabajo fue cómo combinarlos de manera exitosa en una arquitectura híbrida (presentando una alternativa a los trabajos encontrados en la etapa anterior). Sus fortalezas y falencias son conocidas y esto ayudó a encontrar la mejor manera de combinar los algoritmos. Los

trabajos revisados aportaron muchos ejemplos de usos de estos algoritmos y combinaciones posibles.

El presente trabajo hace una revisión de algoritmos, pero el objetivo del trabajo es encontrar un nuevo modelo de navegación completa. Luego de esta etapa de investigación se decidió utilizar combinaciones de algoritmos probadas, conocidas y compatibles. Tanto APF como Astar fueron elegidas por su popularidad, bondades y extensos casos de estudio. Si bien se consideraron otros algoritmos, no todos eran compatibles y el objetivo fue encontrar un modelo funcional que superará sus partes constitutivas.

A.2.1.4. Buscar un marco de prueba

Es importante para este tipo de trabajos un marco de pruebas que permite demostrar que los objetivos planteados son alcanzados. En la mayoría de los trabajos revisados el objetivo suele ser mejorar un poco un algoritmo ya existente, por lo tanto el marco se restringe a las fortalezas y debilidades de ese algoritmo. La búsqueda de una solución completa (sección 1.2) planteó la búsqueda de un marco de pruebas general.

Preguntas de Investigación	¿Existen formas de comparar los resultados? ¿Hay sets de validación como los disponibles en trabajos de reconocimiento de imágenes?
Protocolo de Búsqueda	Motion Planning Benchmark
Criterios de Selección	Que no fueran solo propuestas, que tuvieran casos de uso
Trabajos más relevantes de esta etapa	● Bench-MR: A Motion Planning Benchmark for Wheeled Mobile Robots ● MRPB 1.0: A Unified Benchmark for the Evaluation of Mobile Robot Local Planning Approaches
Conclusiones	Todavía no existe dentro del área un marco que permita comparar los algoritmos entre ellos. Habitualmente las comparaciones son dentro de familias de algoritmos muy acotadas (sección 6.2)

Este vacío es lo que llevó al desarrollo de ambientes de prueba propios (sección 5.2) que se fueron construyendo en base a los ejemplos encontrados en otros trabajos, propios o simplemente para desafiar a la solución propuesta y tener mayor certidumbre de la validez del modelo.

A.2.1.5. Alternativas de navegación

Durante el trabajo fue necesario buscar algoritmos adicionales para complementar la solución y poder superar los obstáculos encontrados durante el desarrollo de la solución y los diferentes ambientes propuestos. En la sección 4 se detalla el camino y obstáculos encontrados en esta etapa.

A.2.2. Fuentes de información más utilizadas

El trabajo de búsqueda se realizó utilizando distintos buscadores como el de IEEE o Google Scholar. Pero como el área es todavía joven, el trabajo incluye muchas fuentes distintas. Si bien la búsqueda fue amplia, sólo se incluyeron fuentes académicas, libros, trabajos presentados en revistas científicas o de la especialidad.

Los sitios/repositorios con más artículos y generalmente de mayor calidad fueron:

- IEEExlore: https://ieeexplore.ieee.org/
- Elsevier: https://www.sciencedirect.com/
- ACM digital library: https://dl.acm.org/
- Springer: https://link.springer.com/
- Sitios de universidades con trabajos dentro del área

A.2.3. Papers revisados y su frecuencia temporal