uc3m | Universidad Carlos III de Madrid

**Grado en Ingeniería electrónica industrial y automática**

**2018-2019**

**Trabajo Fin de Grado**

# "APLICACIONES DE FAST MARCHING A LA NAVEGACIÓN"

---

**Moisés Calderón Arribas**

**Tutor**

**Santiago Garrido Bullón**

**9 de junio de 2019**

## ABSTRACT

Durante muchos años el cálculo de trayectorias ha sido un campo ampliamente estudiado, y constituye el bloque fundamental para lograr el desplazamiento autónomo de un robot.

Con el objetivo en el horizonte de alcanzar una trayectoria óptima, ese extenso estudio ha originado la existencia de multitud de métodos que combinan velocidad, seguridad y coste en diferentes dosis. Por ello a la hora de seleccionar el algoritmo más adecuado para el proyecto, deberá elegirse en función del balance de esos tres parámetros que más convengan.

Los diferentes métodos tienen en común la meta que persiguen, que es encontrar solución al problema que representa buscar la trayectoria óptima entre dos puntos a tiempo real y esquivando los obstáculos que puedan surgir en el recorrido.

Uno de los algoritmos más extendidos es el de Fast Marching y su derivado Fast Marching Cuadrado ($FM^2$) que será desarrollado en el presente documento donde podrá conocerse el funcionamiento del mismo en su totalidad. Para cerrar podrá contemplarse la aplicación de esta metodología en un entorno real para facilitar su comprensión.



Palabras clave: Fast Marching, Fast Marching Cuadrado, Path Planning, Eikonal

# CONTENIDO

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# Capitulo 1. INTRODUCCIÓN

Durante los últimos años se ha visto incrementada de manera considerable la demanda de sistemas que sean capaces de trazar la ruta más corta entre dos puntos, ya sea para un desplazamiento dirigido (GPS, Google Maps) o para un desplazamiento autónomo (navegación inteligente).

Con el crecimiento de políticas empresariales de reducción de costes, la expansión de la utilización de robots, y la búsqueda de mejorar la experiencia del usuario, se ha puesto de manifiesto el valor del desplazamiento autónomo e inteligente en los sistemas de transporte. Por este motivo surgen nuevos algoritmos y metodologías que buscan una optimización de los antiguos sistemas.

Se trata de un campo que ha experimentado cambios constantemente para tratar de superarse y seguir creciendo, como consecuencia de la productividad que ofrece este nicho de mercado. Y es por ello que existe una gran variedad de técnicas para llevar a cabo el cálculo de trayectorias entre dos puntos, pero uno de los que mejores resultados arroja es el método de Fast Marching en torno al cual gira este documento.

Aplicado a la navegación marina, el citado método permite completar un recorrido de manera autónoma esquivando los obstáculos del camino. Es el resultado de la combinación de cuatro parámetros que son la velocidad, seguridad, coste computacional y tiempo real para obtener los resultados más óptimos posibles que se adecúen a las situaciones cambiantes del entorno.

Partiendo de una misma base, el Fast Marching desarrolla multitud de variantes- las cuales también serán expuestas- que combinan de distinta manera los cuatro parámetros anteriores, lo que añadirá diferentes cualidades a la hora de encontrar una solución.

Durante el desarrollo del proyecto podrá conocerse desde la base matemática que sustenta al Fast Marching hasta una aplicación en un entorno real, pasando por la implementación en software y dando una visión global de conjunto al documento para ser capaz de comprender el algoritmo completo con la máxima perfección posible.

De la misma manera el último apartado engloba las cuestiones socio-económicas relacionadas con la aplicación del proyecto y un pequeño estudio legislativo del mismo.

## 1.1. Contenido

La aplicación consiste en un sistema de navegación autónomo de un barco en un embalse sin colisionar con obstáculos ni bordes. La embarcación ha de recorrer la trayectoria marcada por el usuario con los puntos de inicio y final de manera óptima, incluyendo la necesidad de evitar determinados obstáculos móviles como pueden ser otros barcos que puedieran interferir en su trayectoria. Por ello debe ser un sistema que en tiempo real analice todas las constantes que sus sensores le proporcionan para lograr el objetivo.

Este desarrollo a tiempo real puede convertirse en uno de los principales enemigos a los que ha de enfrentarse, siendo la capacidad y velocidad de computación del sistema factores determinantes, de la misma manera tiene un papel relevante la precisión de la solución. Conviene destacar que cuanto mayor sea la proximidad de la embarcación a los bordes u obstáculos su velocidad será menor, por lo tanto de entre la multitud de métodos existentes deberá seleccionarse el que más se adapte a los objetivos requeridos, estudiando así el ratio Precisión/Velocidad de computación.

El documento comienza presentando el método matemático que hace posible el Fast Marching -su base algebraica como es la ecuación de Eikonal y el gradiente- para después explicar las diferentes variantes del algoritmo. Lo siguiente es analizar el código de implementación para entender cómo aplicar esos conocimientos matemáticos en software, y por último se analizará un caso práctico aplicando el algoritmo en un entorno real que será el embalse del Atazar.

Durante el desarrollo de esta memoria es necesario tener varias puntualizaciones en cuenta. Una de ellas es que va a analizarse en profundidad el bloque de evasión de obstáculos bajo el título “Choques” en los sucesivos apartados del escrito, al igual que el bloque de “Coeficiente de seguridad”. Esto responde a que al ser campos muy importantes en el resultado final es conveniente que sea explicado en las diferentes fases del proyecto para mostrar cómo trabaja este bloque.

Otra puntualización gira en torno a la utilización del término robot. Un robot puede representar desde un ovni multicolor de gigantes dimensiones hasta un insecto de color negro por lo que durante el desarrollo de la memoria se citará el objeto de estudio como

robot y nunca de barco. Será en el apartado 5 donde se desarrolle un caso práctico aplicado en un entorno real cuando pueda emplearse la definición de barco.

El programa está desarrollado sobre Matlab porque el entorno de desarrollo lo hace idóneo para las tareas matemáticas y geométricas, así como para la función de representación gráfica.

# Capitulo 2. PLANIFICACIÓN DE TRAYECTORIAS

[1] [2] El cálculo de trayectorias o "Path Planning" representa una disciplina que brinda la característica más representativa de un robot y materializa su concepto: la autonomía. Dotar de inteligencia a un objeto requiere de la interconexión de multitud de disciplinas, y a pesar de que puede parecer un campo relativamente reciente, sus comienzos datan de los años 70; si bien es cierto el cálculo de trayectorias ha sufrido un crecimiento exponencial en la última década como resultado de la mejora de los sistemas computacionales que permiten velocidades de procesamiento, capacidad de memoria y tiempos de computación inalcanzables hace unos años.

Se abre así un campo muy complejo a la par que amplio donde se encuentran inmersas una gran cantidad de fórmulas algebraicas, métodos matemáticos y relaciones geométricas que quedan ocultas a vista del usuario final. Para comprender este gran bloque y todas sus subdivisiones es necesario explicar todo lo que concierne al mundo del cálculo de trayectorias, comenzando por los bloques que describen el entorno como son el modelado geométrico y el espacio de configuración.

Antes de comenzar a explicar cualquier apartado es conveniente poner en conocimiento un par de conceptos expresados en [3] :

- "Un camino o 'path' es óptimo si la suma del coste del desplazamiento es la mínima entre todos los caminos posibles desde un punto inicial al punto objetivo."
- "Un path es completo si encuentra siempre un camino posible en un tiempo finito, en caso de existir. Si por el contrario no existe ninguno que cumpla, el path comunicará la inexistencia de una solución en un tiempo también finito."

## 2.1. Modelado geométrico y C-Space

[4] [5] La resolución de cálculos de trayectoria requiere una visión global del problema. No basta con entender cómo un objeto llega de un punto a otro, pues eso siguiendo la dirección de máximo gradiente es suficiente (como más adelante se explicará), sino que es obligatorio conocer el entorno y los métodos matemáticos en los que se basa.

Para conocer este entorno salen a la luz dos conceptos básicos. El primero de ellos es el modelado geométrico de los cuerpos y el segundo representa el espacio de configuración o "Configuration Space".

El modelado geométrico conlleva conocer la distribución y forma de esos objetos en el espacio para poder así desarrollar el resto de actividades en función de ellos. De cara al movimiento no es lo mismo que un cuerpo sea por ejemplo circular y pequeño a que sea un pentágono de grandes dimensiones, pues la trayectoria final a seguir puede modificarse de manera considerable.

De esta manera en el modelado geométrico se diferenciarán 2 entidades de diferente naturaleza: el robot y los obstáculos.

Por lo que al desarrollo de esta memoria respecta se tomará un mundo W donde,

$$W = R^2 \qquad (2.1)$$

que representa un entorno de 2 dimensiones identificado por sus tres parámetros $q = (x, y, \theta)$.

[6] Un robot tiene un ratio amplio de movimientos y un gran número de grados de libertad. Por ello se define la configuración de un robot como aquello que representa todas las posiciones probables del mismo respecto a un sistema de coordenadas. El conjunto de todas las configuraciones del robot constituye el espacio de configuración o "C-Space", es decir el lugar continuo donde el algoritmo deberá buscar una solución.

Para entenderlo mejor el C-Space representa al conjunto de transformaciones que se pueden aplicar a un robot, en otros términos es la estructura que identifica y conoce el mundo que lo rodea.

El mundo W de dos dimensiones que contenía dos entidades del modelado geométrico (los obstáculos y el robot) engloba al espacio de configuración. Así de la misma manera el C-Space se divide en dos subcampos que representan los obstáculos y el espacio libre como se observa en la figura 2.1. Por un lado el $C_{free}$ engloba los puntos del C-Space por donde el robot tiene libertad para moverse, mientras que el $C_{obs}$ es un C-Space incontable

e infinito donde se modelan los obstáculos como una colección de subespacios de *m*-conexiones.

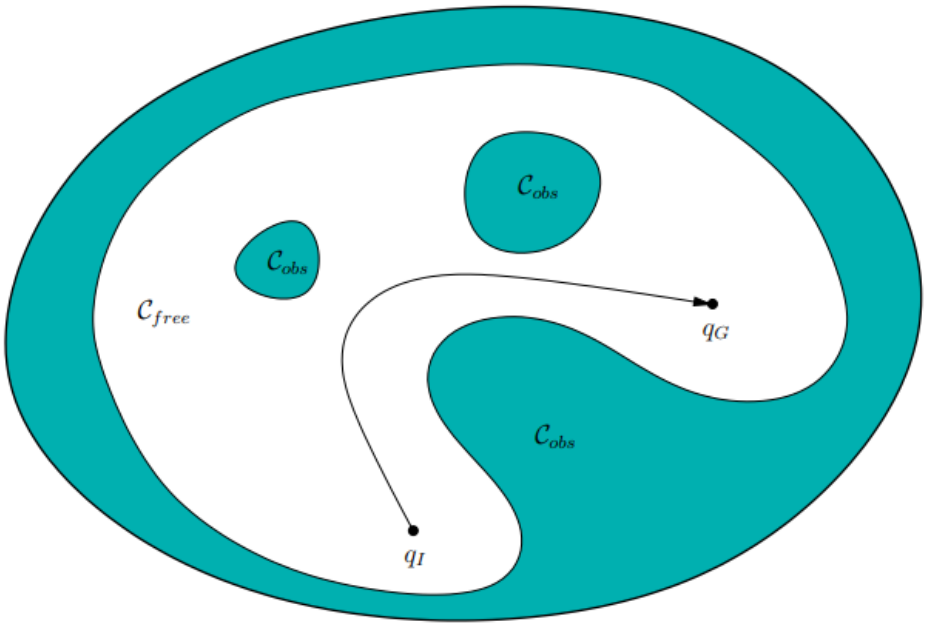



Fig. 2.1: Espacio de Configuración (C-Space) formado por el $C_{free}$ de color blanco y el $C_{obs}$ de color oscuro con $q_1$como punto de inicio y $q_G$ como punto final. *[2]*

[7] [8] Una vez comprendidos el modelado geométrico y el espacio de configuración se presentan los métodos más habituales que llevan a cabo el cálculo de trayectorias. Estos pueden ser de dos tipos, discretos o continuos en función de cómo trabajen sus espacios correspondientes. Así de esta manera un espacio continuo buscará las soluciones en un espacio infinito, mientras que la ventaja de los métodos que discretizan el espacio reside en disminuir el espacio del problema a uno con un número finito de dimensiones.

### 2.1.1. Aproximaciones basadas en muestreo

Los “Sampling-Based methods” o aproximaciones basadas en muestreo tienen como función muestrear el espacio continuo y llevar a cabo búsquedas discretas empleando un módulo detector de obstáculos.

Estos algoritmos son los más extendidos en los últimos años. Rastrean el C-Space con un esquema de muestreo, y para ello emplea el módulo de detección de colisiones llamada caja negra o ”Black box” en lugar de construir el $C_{obs}$ –como si harán las aproximaciones combinatorias. En la Figura 2.2 puede comprobarse que estos métodos buscan la solución con la Black box de manera iterativa y secuencial y, como su propio nombre indica, muestreando el espacio o entorno. El orden en el que se realicen dichos muestreos es muy importante, pues dará lugar a técnicas muy diferentes.

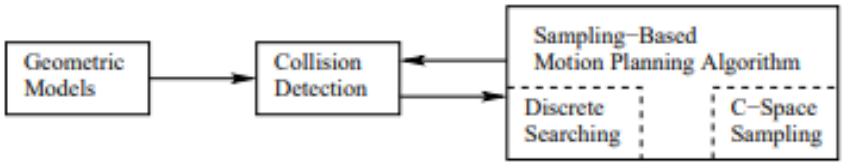



Fig. 2.2: Esquema de las aproximaciones basadas en muestreo que aísla el problema de evasión de obstáculos en un módulo independiente, el conocido como “Black box”. Se puede observar separación de este bloque del modelado geométrico y del algoritmo de muestreo en sí. *[2]*

Hay dos técnicas importantes que encarnan a la perfección la descripción del método:

- Probabilistic RoadMaps (PRM): muestrean de manera aleatoria el C-Space generando vértices con cada muestreo. Acto seguido comprueba si el campo de visión está libre de colisiones para conectar los vértices más cercanos.

  Es un método completo probabilísticamente pero no es óptimo, pues en entornos estrechos es complicado establecer el grado de “campo de visión libre” que el entorno pueda tener.

- Rapidly Expansion Random Trees (RRT): partiendo de un punto inicial genera ramas aleatorias por el espacio como si de un árbol en crecimiento se tratara. Los choques de las ramas con los obstáculos van almacenándose para lograr conocer el camino de ramas que conduce al punto final sin choques.

Es un método agresivo y con un grado de convergencia desconocido (Figura 2.3), ya que experimenta un crecimiento exponencial de soluciones en cada iteración. A su favor tiene la sencillez de implementación.

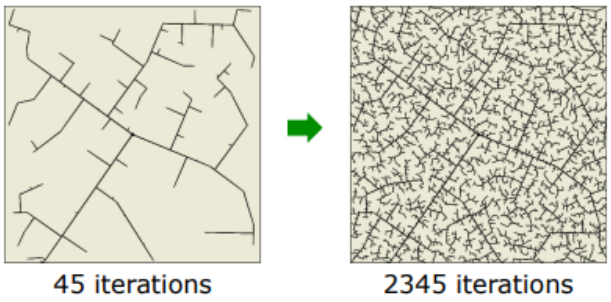



Fig. 2.3: Comparación después de 45 iteraciones del método Rapidly Expansion Random Trees (RRT) y después de 2345 iteraciones. *[7]*

### 2.1.2. Aproximaciones combinatorias

A partir de un espacio continuo que reciben, los métodos combinatorios o "Combinatorial planning" generan una representación exacta del mismo espacio de configuración pero en discreto.

Las diferentes técnicas combinatorias tienen en común dos características. La primera es que pertenecen al grupo de métodos discretos en los que el espacio es dividido en partes o trozos atendiendo al mecanismo empleado; mientras que la segunda es el resultado que producirán, que son los "Roadmaps" u hojas de rutas. Los Roadmaps son grafos en el $C_{free}$ en los que cada vértice representa una configuración y cada borde constituye un camino posible. Así de esta manera se almacena la conectividad del $C_{free}$ en los grafos donde buscará la solución.

Trabaja generando una representación exacta y discreta del problema original sin perder información, por lo que completan siempre el cálculo. Esto permite clasificar los algoritmos como los más seguros ya que proporcionan respuestas fiables en todo momento tanto si existe solución como si no.

[9] Existen técnicas diferentes que llevan a cabo esto: las más comunes son el diagrama de Voronoi, descomposición en celdas exactas, descomposición en celdas aproximadas y grafos de visibilidad.

El método de descomposición en celdas atiende a una división vertical del espacio por los vértices que presenta el espacio de configuración y su interior (Figura 2.4). Cuando se detecta un vértice en el espacio se traza una línea vertical para después generar un nodo en el punto medio de esa recta. Uniendo los nodos se obtiene un camino.

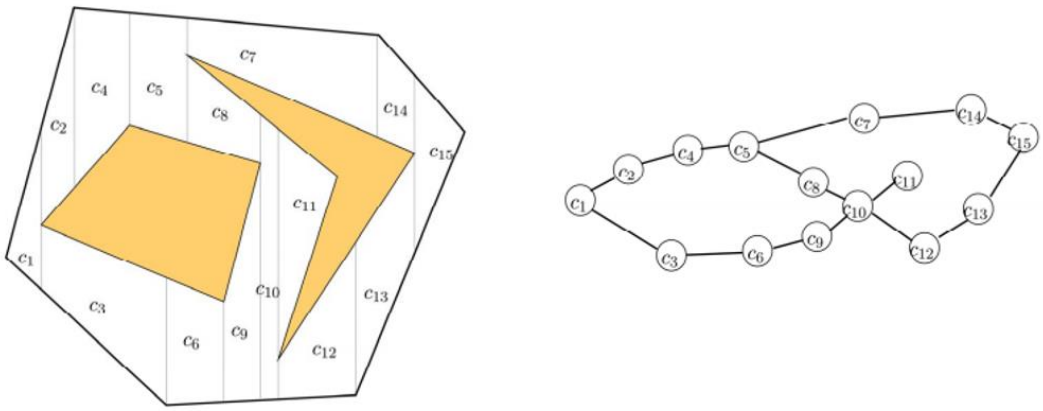



Fig. 2.4: Representación en dos pasos del método de Descomposición en celdas *[9]*

Atendiendo a una clasificación de LaValle [2] las aproximaciones basadas en muestreo y las combinatorias constituyen los dos principales grupos de división en cuanto a cómo discretizan su espacio, y sus diferencias son más que reseñables.

Si bien los métodos combinatorios proporcionan una respuesta completa y fiable, si el C-Space es un entorno amplio emplea un tiempo muy elevado en el proceso. A este inconveniente se le añade además la dificultad de implementación lo que en muchas ocasiones le desplaza del mercado industrial en beneficio de los métodos basados en muestreo.

Ahora bien, éstos últimos no son tan completos como los primeros por lo que deberán buscarse los parámetros que más se adecuen al caso de estudio que se realice. En algunas situaciones es más necesaria la seguridad de obtener una respuesta fiable, y en otras se busca emplear el mínimo tiempo necesario para obtener una solución, por ello es vital una evaluación previa de las necesidades del sistema a desarrollar.

### 2.1.3. Campos potenciales

[10] [11] [12] En estos métodos el robot puede asemejarse a una partícula sometida a campos potenciales de atracción y repulsión.

El espacio es discretizado como una malla y los obstáculos modelados como fuentes productoras de campo repulsivo, de manera opuesta al punto objetivo que constituirá una fuente de atracción. Las primeras emiten un potencial de repulsión no negativo, continuo y con tendencia a infinito para que en ningún caso se le acerque la partícula móvil. Pero ahora bien, la fuerza de este campo potencial debe ser limitada y controlada dentro de unos valores, pues si la fuerza se extiende mucho por el entorno, haría impracticable el movimiento de la partícula por el mapa.

Transcribiendo esta información en una fórmula,

$$U_{art}(x) = U_{xd}(x) + U_o(x) \qquad (2.2)$$

se obtiene la potencia artificial $\mathrm{U_{art}}$ en el punto x a la que está sometida la partícula. Este valor es resultado de la suma del potencial generado por los obstáculos, $\mathrm{U_o}$, y el potencial de atracción del punto final $x_d$, $\mathrm{U_{xd}}$.

Utilizando Lagrange en la ecuación 2.2 el vector que se obtiene al aplicar el campo potencial artificial $\mathrm{U_{art}}$ es $F^*_{art}$:

$$F^*_{art} = F^*_{xd} + F^*_o \qquad (2.3)$$

Donde:

$$F^*_{xd} = -\nabla[U_{xd}(x)] \qquad (2.4)$$

$$F^*_o = -\nabla[U_o(x)] \qquad (2.5)$$

[13] [14] [15] Así $F^*_{xd}$ representa la Fuerza de atracción que resulta de aplicar el gradiente descendente sobre la función potencial de atracción $U_{xd}$, al igual que $F^*_o$ es la Fuerza de

repulsión al computar el gradiente descendente sobre el potencial de repulsión $U_o$ en el punto $x$.

El gradiente descendente es el vector que encuentra los valores de la función que minimiza el coste de la misma, es decir optimiza la función.

El problema de estos métodos es que el robot puede caer atrapado en un mínimo local. Esto sucede cuando un punto vecino a la fuente de atracción o repulsión tiene un valor de potencial inferior o superior respectivamente al de la propia fuente, ocasionando una incongruencia. Para ello se crea un campo potencial artificial/una función de navegación libre de local mínima y se aplica el gradiente sobre ella.

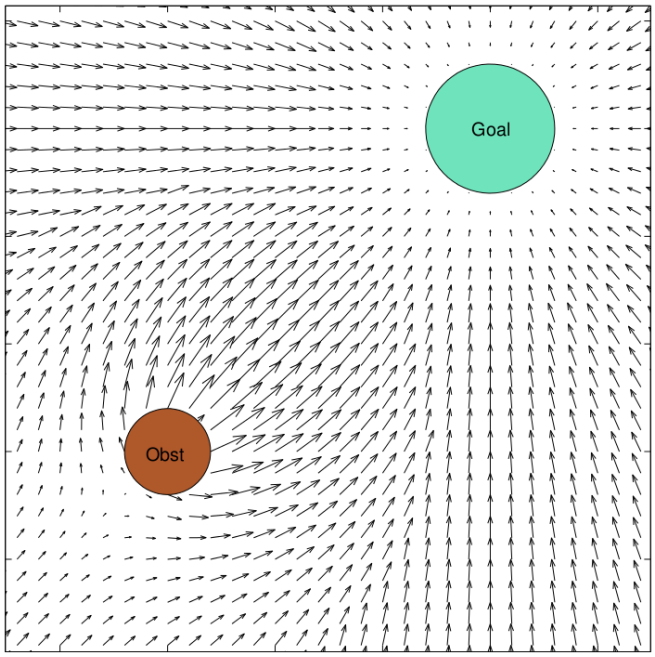



Fig. 2.5 Campos potenciales de atracción y repulsión que modelan las fuerzas que sufre una partícula mientras avanza por el espacio. El obstáculo emite fuerzas repulsivas mientras que la meta o "goal" produce un campo de atracción. *[16]*

### 2.1.4. Algoritmos de búsqueda

[17] [18] [19] [20] Estos métodos buscan una secuencia de pasos que les guíen hasta el punto objetivo. Van chequeando el estado de los puntos del entorno y encadenando nodos para alcanzarlo.

Una clasificación primaria podría dividir en 2 conjuntos los métodos basándose en la información que reciben:

- Métodos no informados: Se trata de métodos ciegos que no usan la información proporcionada por el entorno para reducir la búsqueda, sino que directamente examina todos los nodos del espacio. Los dos más representativos son Breadth-first Search (BFS) y Deepth-first Search (DFS).

- Métodos informados: También conocidos como heurísticos, emplean una función $f(n)$ que permite hacer una expansión entre nodos inteligente. Esto quiere decir que a partir de los nodos ya generados, se escogen aquellos hacia los que cueste menos desplazarse. Algunos exponentes importantes de este método son A* y D* basados en el método Dijkstra.

El método Dijkstra toma forma con una modelación discreta del espacio en forma de nodos y ramas que los conectan. Consiste en ir saltando de nodo en nodo desde el punto inicial siguiendo la rama más corta sucesivamente hasta alcanzar el punto final. Diferencia tres estados para la clase rama y dos para la clase nodo que informan (de ahí el nombre) si el elemento pertenece o no al recorrido que se está siguiendo y los posibles candidatos para seguir avanzando.

De esta manera la función heurística $h(n)$ es la parte clave de la función $f(n)$ que selecciona los nodos, siendo $h(n)$ la que represente el camino más económico del nodo $n$ al objetivo:

$$f(n) = h(n) \tag{2.6}$$

Una forma de mejorar la Ecuación 2.6 sería no solo tener en cuenta la distancia al nodo más próximo $h(n)$, sino además preocuparse también del coste $g(n)$ para llegar a dicho nodo:

$$f(n) = h(n) + g(n) \tag{2.7}$$

Así, con la ecuación 2.7, se constituye el método de búsqueda A* que clasifica en dos columnas el estado de cada nodo en función de si es un punto aún libre, o uno ya alcanzado. Va computando los nodos con menor coste asociado para alcanzar el objetivo como muestra la Figura 2.6 con los números entre paréntesis. Este método además es capaz de detectar vías ciegas que finalizarían su recorrido y tomar caminos alternativos, así como explorar otras rutas por si no se ha seguido la trayectoria más corta.

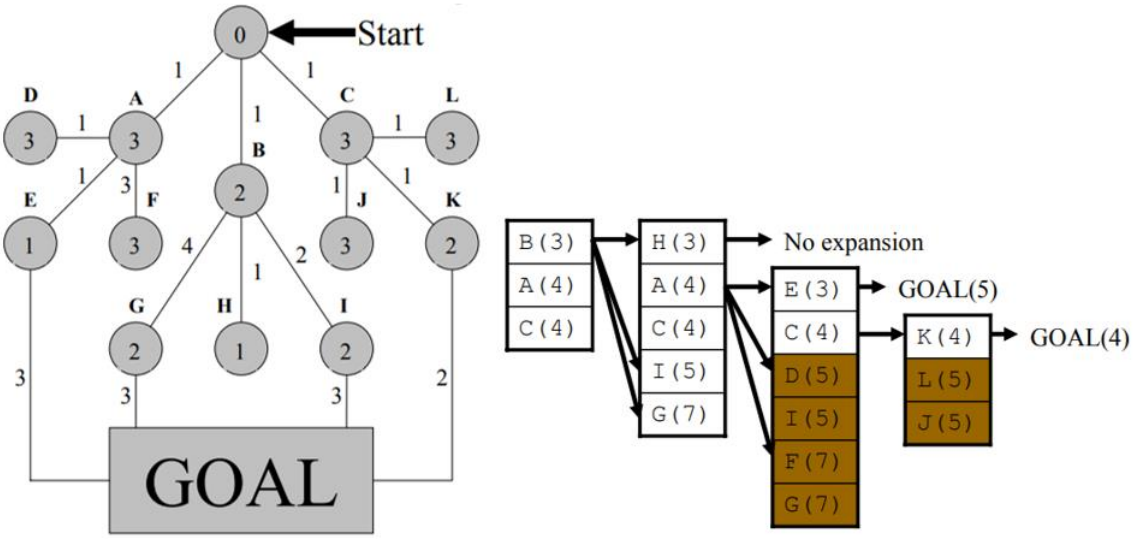



Fig. 2.6 Metodología que emplea el algoritmo de búsqueda mediante nodos para llegar del nodo 0 inicial al punto objetivo. A la izquierda la representación esquemática del método y a la derecha sucesión de concatenaciones para lograr el objetivo. *[17]*

Para solventar problemas de ineficiencia en espacios grandes surge el método D* que emplea la misma fórmula y sistemática que el A*, pero añade la funcionalidad de ser capaz de analizar la función heurística de manera independiente. De esta manera escoge entre la $h(n)$ anterior y la $h(n)$ nueva comprobando cual es más eficaz y desechando la nueva si es preciso.

Se ha puesto especial interés en este apartado porque dentro de los algoritmos de búsqueda se incluye el método a desarrollar en esta memoria que es el Fast Marching. El algoritmo establece un tiempo de llegada $T$ a cada nodo mediante la expansión de una onda y discretizando la malla ortogonalmente.

Este apartado deja patente el continuo deseo de mejora de las técnicas de planificación de trayectorias

## 2.2. Choques

[10] El principal atributo de un robot autónomo es su independencia, que quiere decir la capacidad de realizar tareas sin necesidad de una intervención de terceros.

Un campo de interés, quizá el que despierte mayor dificultad, dentro de esta disciplina es la evasión de obstáculos. Siempre ha sido un rompe cabezas para los investigadores, pues este alto grado de dificultad debe comprimirse en los niveles más bajos de los sistemas de control para no saturar al robot, y permitirle así el desarrollo de su trabajo en tiempo real. Esto le permitirá al sistema una mejor distribución de tareas que optimizará su capacidad de computación.

En la mayoría de algoritmos donde más tiempo se emplea es precisamente en esta función de detección de obstáculos. Existe una gran variedad de metodologías que trabajan la tarea y algunas de ellas se han citado anteriormente de manera directa o indirecta.

Una de las formas consiste en utilizar un módulo de detección de obstáculos independiente que va comprobando si existe contacto entre dos objetos, otra forma va implícita en el método pues expande un potencial alrededor de los obstáculos para que nunca colisionen. [21] Otra manera es chocando directamente con el obstáculo y siguiendo el perfil del mismo hasta que acabe como hace Bug algorithm o “Algoritmo de insecto” y RRT.

# Capitulo 3. METODOLOGÍA FAST MARCHING Y DERIVADOS

[22] En la naturaleza, las abejas para escapar de una habitación a oscuras buscan la luz que les guíe hasta la salida, y esto responde a que el haz de luz siempre sigue el camino más corto, la ruta más directa. El principio de Fermat fundamenta esa teoría y en consecuencia la ecuación Eikonal.

El Fast Marching es un algoritmo matemático para el cálculo de trayectorias que encaja dentro del grupo de métodos de búsqueda heurísticos. Concebido por Oshen y Setian en 1996 se ha convertido en un algoritmo muy utilizado debido a las soluciones tan fiables que produce en un tiempo no demasiado elevado con respecto a otros métodos.

En la década de los 80 cuando comenzó a explotarse este ámbito de manera más destacada, los expertos buscaban solamente la consecución de un recorrido completo entre un punto inicial y un punto final. Pero en estos años de constante cambio y evolución se busca alcanzar no solo ese objetivo, sino también la seguridad, fiabilidad y solidez de la trayectoria entre otros. Es precisamente la búsqueda de ello lo que ha generado multitud de alternativas al Fast Marching tratando de optimizar cada vez más el resultado final del "path" o recorrido.

A pesar de la cantidad de ramas derivadas del método inicial todas ellas gozan de la misma base algorítmica: una o varias fuentes emiten un frente de onda que se expandirá por un medio hasta alcanzar un punto objetivo, siendo el camino que recorren el más corto en tiempo. Algunos de estas variaciones del Fast Marching inicial son: FM sobre diagrama de Voronoi, $FM^2$, FM Saturado o $FM^{2*}$.

## 3.1. Método Fast Marching

[23] [24] La mecánica de funcionamiento es semejante a la planteada por Dijkstra, y puede asemejarse a las ondas acuáticas generadas sobre la superficie tras el lanzamiento de una piedra sobre un lago. De esta manera, la fuente de las ondas será el punto donde la piedra cae expandiendo así olas circulares concéntricas que van propagándose a lo largo de toda la superficie. La expansión sobre el medio no homogéneo se produce de nodo en nodo empleando la ecuación Eikonal.

### 3.1.1. Ecuación Eikonal

[25] [26] El método Fast Marching calcula el tiempo que el frente de onda tarda en alcanzar cada punto del espacio, empleando para ello la ecuación de Eikonal. Se trata de una ecuación basada en derivadas parciales con no linealidad que computa el movimiento de una onda plana con una frecuencia $\omega$ cuyo vector sigue una dirección perpendicular al frente de onda.

Un punto de la onda se mueve por el espacio siguiendo la ecuación de movimiento rectilíneo uniforme donde la distancia $\theta$ recorrida por dicha partícula es proporcional a la velocidad $F$ durante un tiempo $T$:

$$\theta = F * T \tag{3.1}$$

Al aplicar la derivada espacial sobre la ecuación de movimiento se obtiene

$$\frac{\theta}{d\theta} = F * \frac{\partial T}{\partial \theta} \tag{3.2}$$

la cual en una dimensión define la ecuación de Eikonal

$$1 = F(x)\,|\nabla T(x)| \tag{3.3}$$

De esta manera el tiempo, denotado como $T$, que una onda tarda en alcanzar un punto $x$ cualquiera y la velocidad de la misma, llamada $F$ componen la ecuación Eikonal sobre una malla rectangular ortogonal.

Reagrupando términos se puede comprobar que el gradiente es inversamente proporcional a la velocidad $F$:

$$\frac{1}{F} = |\nabla T(x)| \quad (3.4)$$

A pesar de que se ha considerado la ecuación como unidimensional, se puede aplicar de la misma manera en varias dimensiones pues el gradiente es ortogonal a un conjunto de niveles de la función $T(x)$.

Esta ecuación constituye el principio en el que se basa la metodología Fast Marching.

### 3.1.2. Mecánica de funcionamiento

Para aplicar la metodología Fast Marching debe considerarse el sistema como no estacionario pues la velocidad $F > 0$ no cambia de signo. Esto significa que la onda solo cruza cada punto una vez: avanza hacia adelante sin interferencias ni reflexión de las ondas, lo que simplifica bastante el cálculo [27] y a su vez lo diferencia de la metodología Level Set. [28] Así mismo el medio de expansión se considera como no homogéneo, por lo que la onda se expande a diferentes velocidades a lo largo del tiempo.

A modo de inicio, debe prepararse el espacio estableciendo los obstáculos con velocidad $F$ igual a 0 (color negro), y los espacios libres con velocidad de expansión 1 (color blanco). Este método basa su funcionamiento en la división del mapa en celdas para su etiquetado, así de esta manera se diferencian 3 tipos de puntos:

La fuente generadora de la onda que tendrá el valor de tiempo $T = 0$ se cataloga como punto *Congelado* y constituye un mínimo global, nunca local. Esta diferencia es importante ya que si la fuente se considerase mínimo local cualquier punto del mapa podría tener un tiempo de llegada $T$ menor que el que tiene la fuente o cualquiera de sus vecinos, cosa que es imposible a toda costa.

Las celdas vecinas a la fuente pertenecen a la llamada *Narrow Band* que son puntos que aún no forman parte a la trayectoria de desplazamiento pero que deben ser considerados como potencialmente probables de serlo debido a su proximidad. Por último, la tercera clasificación la conforman celdas de espacio libre que aún no pertenecen a nada y son etiquetadas como *Desconocidas*.

Entonces se pone a trabajar la maquinaria Fast Marching y en cada iteración se resuelve la ecuación de Eikonal que va propagando la onda concéntrica generada por la fuente hacia las celdas con menor tiempo de llegada $T$. De esta manera las celdas catalogadas como *Narrow Band* se convierten en celdas *Congeladas* al igual que ya lo era la fuente. Esto provoca que los puntos *Desconocidos* de alrededor de estas celdas ya congeladas se conviertan en *Narrow Band* como puede observarse iteración tras iteración en la Figura 3.1. El algoritmo se repite hasta que todas las celdas del espacio se encuentran *Congeladas* o se alcance el punto final donde la partícula desea llegar.

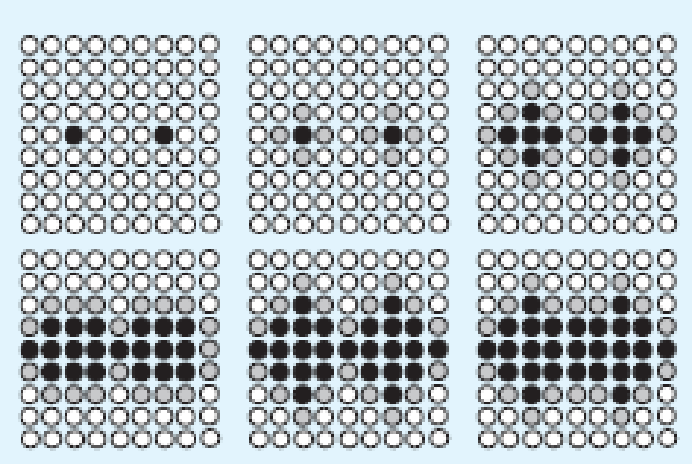


Fig. 3.1: Propagación de la onda de Fast Marching desde dos fuentes. Las celdas en blanco representan el estado Desconocido, las de color gris pertenecen a la Narrow Band, mientras que las celdas pintadas de negro son celdas ya Congeladas. *[23]*

[29] Entonces a modo de conclusión, el método Fast Marching computa la distancia de un punto inicial al resto de posiciones del mapa usando la ecuación Eikonal. Expande una onda por un medio no homogéneo en el cual la velocidad de propagación $F$ dependerá del punto del mapa en el que se encuentre.

Todos los mecanismos derivados del Fast Marching emplean la misma metodología como luego se verá. Primero se adecúa el mapa mediante técnicas particulares de cada método, para después aplicar en ellas el Fast Marching explicado en este apartado.

### 3.1.3. Gradiente

La expansión de la onda permite computar el tiempo de llegada del punto inicial al resto de puntos y generar así un mapa de distancias, el cual será empleado por el gradiente para encontrar el camino que se busca.

El gradiente consiste en un vector de derivadas parciales que devuelve dos parámetros: la dirección del máximo incremento de la función, y el módulo de la cantidad de variación en esa dirección.

$$\nabla U(q) = DU(q)^T = \left[\frac{\partial U}{\partial q_1}(q), \ldots\ldots, \frac{\partial U}{\partial q_m}(q)\right]^T \quad (3.5)$$

Es una herramienta matemática que maximiza la pendiente siguiendo la dirección del vector de máxima dirección. Ahora bien, el gradiente que se computa para obtener el recorrido final del robot es el gradiente descendente. Este se diferencia por optimizar las funciones sobre las que se aplica, pues busca el mínimo de la función con el menor coste posible.

De esta manera se modela el punto de origen como un mínimo local mientras que la función de expansión se realiza desde el punto objetivo. Así se asegura que, si es posible, siempre se generará una trayectoria que busque el mínimo local, y por lo tanto el camino o "path" completo.

### 3.1.4. Choques

De acuerdo con la Real Academia Española una colisión se define como el choque de dos cuerpos, pero, es necesario insertar esta definición en el sistema y para ello se recurre al software. Por lo tanto, en este apartado va a explicarse cómo a nivel interno el programa entiende que dos objetos van a chocar.

El bloque de detección de obstáculos funciona de la siguiente manera:

Se toma $O \subset W$ como el obstáculo $O$ perteneciente al mundo $W$ , $A \subset W$ como el sólido rígido que representa al robot y el vector $q \subset C$ ($C$ es la configuración del sólido $A$) como el conjunto de configuraciones en las que $A(q)$ intersecta con el obstáculo.

Para comprobar si un objeto golpea con un obstáculo se recurre a simples condiciones lógicas de intersección entre los puntos del obstáculo y el vector de posibles configuraciones espaciales del robot.

Si $A$ toca el obstáculo $O$:

$$int(O) \cap int[A(q)] = 0 \quad y\, O \cap A(q) \neq 0 \qquad (3.6)$$

, por lo que dependiendo de lo que devuelva la función podrá conocerse el estado en el que se encuentra el robot.

También se apunta que el $C_{obs}$ es una colección de subespacios con $m$-conexiones unidos por el vector $q$. Este vector $q \subset C$ ( $C$ es la configuración del sólido $A$)

$$C_{obs} = \{q \in C \mid A(q) \cap O \neq 0\} \qquad (3.7)$$

Ahora bien, el método Fast Marching calcula el camino más corto entre dos puntos sin tener en cuenta ningún aspecto de seguridad de trayectoria nada más que el de evitar que el robot colisione con los obstáculos. No considera la situación de que cuando el robot se mueve cercano a ellos debe reducir su velocidad, por lo que este método es el más rápido pero no el más óptimo.

Para solventar este problema surgió el Fast Marching$^2$ que busca el equilibrio entre los aspectos de velocidad y seguridad que se explicará en el siguiente apartado.

## 3.2. Método Fast Marching$^2$

[28] [22] El método de FM$^2$, desarrollado por Santiago Garrido y Luis E. Moreno, como su nombre deja entrever, se forma mediante la implementación del método Fast Marching de manera duplicada. Para ello resuelve la ecuación Eikonal realizando una interpolación

intrínseca, o lo que es lo mismo, convierte la función que toma valores discretos en una función continua ya que la propagación de la onda se realizará de manera continua.

Consiste en la emisión de ondas en pasos diferentes que generen dos campos potenciales distintos que se complementarán para alcanzar el "path" final.

La primera de ellas tiene como objetivo la generación de un campo de repulsión propagando ondas desde los obstáculos y paredes, lo que dará como resultado un índice de refracción o campo potencial de lentitud ($1^{er}$ potencial) proporcional a la inversa de la velocidad $F$. La cantidad que los obstáculos van a expandirse es medida por el coeficiente de seguridad que más adelante se explicará en un apartado bajo el mismo título.

Este mapa llamado "FM Grid Map" (Figura 3.2) o mapa de velocidades representará la velocidad relativa de cada punto del espacio, la cual es proporcional a la distancia de dicho punto al obstáculo más próximo; por lo que a mayor distancia del obstáculo el punto podrá desarrollar una velocidad mayor. De esta manera se genera un mapa en escala de grises donde cada punto del mapa presentará un tono de gris en función de la velocidad que desarrolle. Los valores entre los que está comprendido son 0 y 1, siendo el blanco velocidad máxima (1) y el color negro velocidad igual a cero (0).

Por este motivo, como el robot mantiene una velocidad directamente proporcional al tiempo $T$ de cada punto del espacio, nunca colisionará ya que cuando se acerque mucho a un obstáculo que tiene tiempo $T$=0, la velocidad sufrirá la misma reducción.

Una vez generado el primer campo potencial se procede a ejecutar el Fast Marching de la misma manera que se ha explicado en el anterior apartado, pero con una modificación. La velocidad de expansión de esta segunda onda es impuesta por el índice de refracción obtenido del primer potencial (el cual es inversamente proporcional a la velocidad de propagación), así generará un mapa donde el robot utilice la máxima velocidad segura en cada punto.

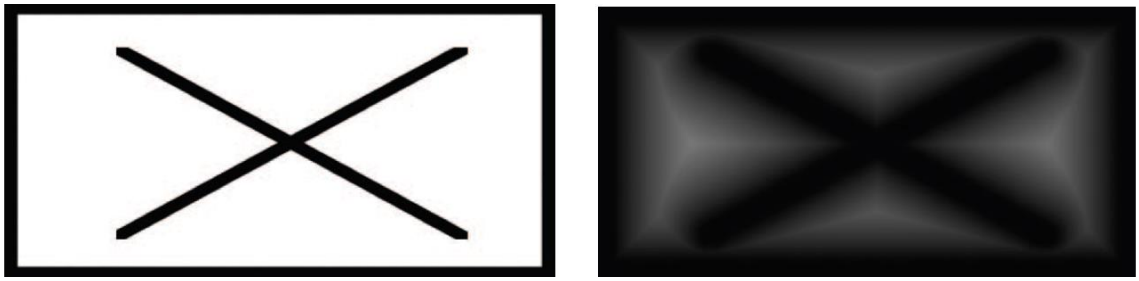


Fig. 3.2. En la parte izquierda se tiene un mapa en blanco y negro de un lugar donde las zonas negras representan obstáculos y las zonas blancas espacios libres. La imagen de la derecha corresponde a aplicar el primer potencial sobre la imagen izquierda y expandir los obstáculos obteniendo el mapa de lentitud en escala de grises. *[23]*

Se recuerda que este desarrollo sucede en dos dimensiones por lo que al añadir una variable extra, tiempo $T$, éste 2º potencial genera la superficie de Lyapunov que es una representación formada por ondas isócronas donde todos los puntos del espacio que tienen el mismo tiempo $T$ desde el origen están comprendidos en la misma circunferencia.

El mapa de distancias conserva el tiempo $T$ de llegada de la onda a cada punto del mismo y proporciona una velocidad $F$ segura que permitirá en cada tramo del "FM Grid Map" tener la máxima velocidad segura posible.

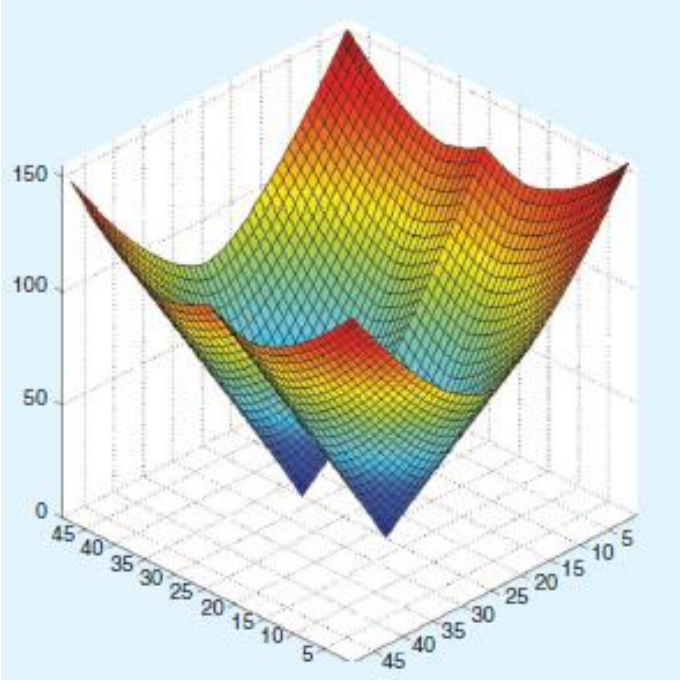



Fig. 3.3: Superficie de Lyapunov (3D) de la Fig. 3.1 como resultado de la expansión de ondas desde dos focos. La tercera dimensión de la figura viene dada por la introducción del eje 'z' que representa el tiempo T de llegada. *[23]*

En la figura 3.3 se puede comprobar la existencia del mínimo global de la función. En este caso se expande una onda desde la fuente que constituye un mínimo global porque es el mínimo absoluto. Si existiera un mínimo local (Figura 3.4) se vería que en otra parte de la superficie cónica existiera una pequeña caída donde el robot pudiera quedar atrapado.

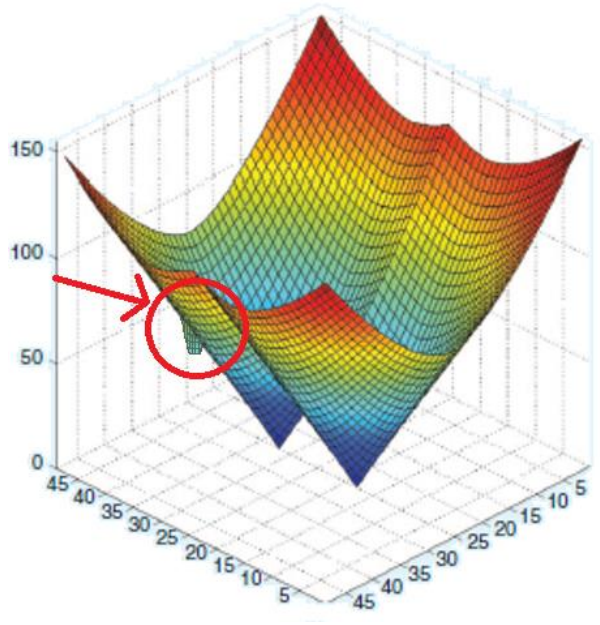



Fig. 3.4 Superficie de Lyapunov con un mínimo local señalado con un punto rojo. Figura de elaboración propia.

Posteriormente se aplica el gradiente descendente en el mapa de distancias desde el punto inicial al punto final, lo que generará la máxima pendiente para calcular el camino más rápido y seguro que recorrerá el robot.

Los beneficios del uso de esta ecuación son muchos pues asegura una respuesta rápida al ser un algoritmo sencillo, algo muy necesario para poder resolver situaciones críticas en tiempo real. También permite al robot mantener una trayectoria suave sin tirones ni movimientos bruscos gracias al coeficiente de seguridad, además de fiable, pues al haber expandido en tamaño en el mapa los obstáculos y paredes nunca habrá colisiones. Todas estas características dan como resultado un recorrido y velocidad óptimos en todo momento.

Por esta razón el algoritmo de Fast Marching$^2$ se incluye dentro de las aproximaciones potenciales, pues los obstáculos generan un campo potencial artificial de velocidad para que no pueda existir colisión.

### 3.2.1. Premisas Fast Marching$^2$

Para desarrollar correctamente el Fast Marching se siguen una serie de condiciones que ayudan a optimizar el método como son la seguridad, solidez de la trayectoria y el bajo coste computacional:

Un recorrido seguro implica evitar giros bruscos y que en ningún momento el robot colisione con los obstáculos, por lo tanto es necesario que se vaya actualizando continuamente convirtiendo al conjunto en un sistema a tiempo real.

Debido a esta necesidad de respuesta instantánea surge la obligación de desarrollar un sistema rápido que no emplee demasiado tiempo en llevar a cabo las operaciones, es decir, un sistema con una carga computacional baja.

Y por último el sistema debe generar una trayectoria suave sin cambios bruscos de dirección asegurando una trazabilidad óptima de la solución final. En la Figura 3.5 se comprueba los cambios bruscos del recorrido en los diferentes mapas.

Una de los motivos por los que este algoritmo es seguro y eficiente es porque siempre el tiempo $T>0$ , lo que se traduce en que el robot nunca colisionará con los obstáculos ya que tienen su tiempo establecido en $T=0$.

Por ello un método completo será aquel que satisfaga estas características, y es el objetivo que se ha buscado en los sucesivos años de investigación. El intento de una optimización del Fast Marching ha generado la multitud de variantes que existen en la actualidad, las cuales tratan de alcanzar una trayectoria que cumpla los requisitos pero es algo muy complicado. Así pues, la mayoría de métodos cumplen varios de ellos pero flaquean en otro aspecto; por ejemplo una variante que tenga una seguridad y fiabilidad óptima pero emplee demasiado tiempo en realizar cada operación nunca será una buena alternativa pues podría originar errores fatales si se aplicara al mundo real.

### 3.2.2. Coeficiente de seguridad

La cantidad de espacio que los obstáculos se expanden es llamada saturación del mapa y puede ser modificada para establecer criterios de seguridad.

Un recorrido será más seguro cuanto más saturado esté el mapa de velocidades, es decir expandirá los obstáculos en mayor medida aumentando el valor del campo artificial de repulsión del obstáculo. Pero esto tiene una contra, que es la rigurosidad de la trayectoria que deberá seguir el robot, lo que desembocará en un escasa o nula capacidad de reacción.

Por el contrario si un mapa apenas está saturado, es decir, apenas se ha expandido la onda desde los obstáculos, el margen de reacción ante imprevistos será mayor pero la seguridad de la trayectoria apenas será existente. Por lo tanto experimentará cambios bruscos en su trayectoria y un descenso relevante de velocidad cuando se acerque a los objetos. Esto puede ser observado en la Figura 3.5 donde a) tiene una alta saturación y d) una saturación prácticamente nula.

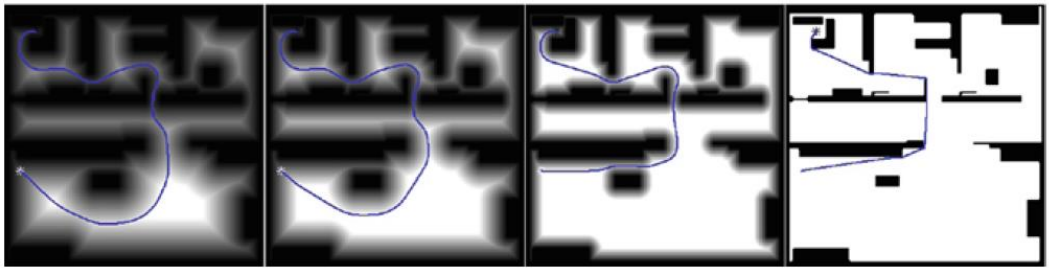


Fig. 3.5: Variación de la saturación en un mismo mapa utilizando diferentes niveles de seguridad. De derecha a izquierda se observan a), b), c), d) donde se comprueba un descenso de la seguridad de la trayectoria inversamente proporcional a la libertad de movimiento del robot. *[22]*

## 3.3. Método Fast Marching sobre diagrama de Voronoi

Basándose en el diagrama de Voronoi, el cual traza la mediatriz de un segmento de un conjunto de puntos del espacio, se divide el mapa en regiones provocando la disminución del tiempo de computación. Y así, tomando este mapa como primer potencial se aplica el método de Fast Marching colocando la fuente emisora de ondas en el punto final.

Este método se diferencia del $FM^2$ simplemente en la forma de obtener el primer potencial siendo más fácil de implementar el $FM^2$ que el de Voronoi.

### 3.4. Método Fast Marching$^2$ Saturado

En algunas ocasiones no es necesario alejar el robot mucho de los bordes y obstáculos para poder alcanzar el objetivo como hace el $FM^2$, sino que con una pequeña distancia de seguridad sirve y por ello nace esta variante. De esta manera se podrá optimizar la trayectoria que seguirá el robot.

Para llevarlo a cabo primero se escala el "FM Grid Map" y después se satura a la máxima velocidad permitida, dando como resultado un mapa con la máxima velocidad posible pero perdiendo un poco de seguridad de trayectoria.

### 3.5. Método Fast Marching$^{2*}$

La principal ventaja de este método es que expande muchas menos celdas para obtener una solución prácticamente idéntica, lo que disminuye el coste computacional hasta 3 veces. Para ello reduce la cola de puntos de la *Narrow Band* escogiendo solamente los que menor tiempo $T$ posean, así la función de la Narrow Band del FM es sustituida por otra llamada *Cost-to-go.*

Dicha función calcula el tiempo óptimo para llegar al objetivo sin la necesidad de expandir muchas más celdas (Figura 3.6).

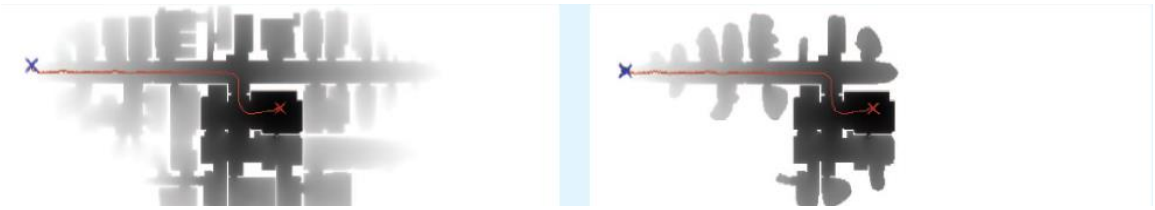


Fig. 3.6 Comparación de la cantidad de celdas expandidas empleando diferentes métodos. A la izquierda se aprecia un mapa en el que se ha aplicado Fast Marching$^{\mathbf{2}}$ mientras que a la derecha se observa el mismo mapa pero aplicando Fast Marching$^{\mathbf{2}*}$. Se puede comprobar que la expansión de las celdas es mucho menor utilizando el FM$^{2*}$ *que el* FM$^{2}$. *[23]*

# Capitulo 4. IMPLEMENTACIÓN

El código desarrollado apoya la secuencia de pasos que el sistema Fast Marching emplea. [30] Una gran parte de las funciones utilizadas fueron implementadas en la "toolbox" o barra de herramientas, de Fast Marching por Gabriel Peyre.

Fig. 4.1. Mapa que muestra una trayectoria calculada desde el punto superior al punto inferior derecho de la imagen. Los dos asteriscos verdes son los puntos de inicio y final del camino de un objeto móvil independiente y desconocido. Figura de elaboración propia.

## 4.1. Tratamiento de imagen

Para poder aplicar el método de Fast Marching se convierte en primordial el correcto trato previo del mapa a estudiar. El preprocesamiento de las imágenes es igual de importante que el propio algoritmo matemático en sí, pues sin este primer paso no es posible que el resto funcione; de esta manera debemos modificar la imagen mediante filtros y técnicas para adecuarla a nuestros intereses y sacarle el máximo rendimiento.

Los pasos a seguir son los siguientes:

- Binarizar la imagen para convertirla únicamente a 2 colores: Blanco (1) y negro (0) mediante la función `im2bw`. Siendo X un valor umbral, binarizar la imagen quiere decir convertir todos sus pixeles con valor por debajo de X a negro y todos los superiores a blanco, reduciendo así la cantidad de información muy considerablemente.
- Aplicación de una operación morfológica como es la dilatación de imagen que extiende las zonas con valores negros (obstáculos) en el entorno gracias a un elemento estructural `SE`. Este paso es el que genera el primer potencial del Fast Marching, expandiendo o dilatando los valores de la imagen para generar un margen de seguridad en el mapa.
- Cálculo de la distancia euclidea que representa la distancia de cada punto del mapa al punto distinto de 0 (obstáculo) más cercano, generando de esta manera el "FM Grid Map". Para ello se emplea la función `bwdist`.
- Por último, reescalando el "FM Grid Map" podemos obtener el mapa de lentitud o "Slowness map ".

```
bw = im2bw(imcolor,0.4);
Wo = bw;

SE=strel('disk',1);
bw3 = imdilate(~bw,SE);

bw4=~bw3;

W=bwdist(~bw4');

W = rescale( double(W) );
```

### 4.2. Puntos salida y llegada

Algo básico para el cálculo de trayectorias es comunicar al robot el punto de inicio y final del camino a realizar para que por sí solo busque el camino más adecuado. Esta tarea toma forma gracias a la función `pick_start_end_point` la cual almacena del mapa `Wo'`

las coordenadas de inicio y final del recorrido en 2 variables. Estas variables son `start_points` y `end_points` respectivamente y representan un vector de 2 enteros *x*, *y*.

Así solamente es necesario pinchar con el ratón en el lugar deseado del mapa y la función `ginput` incluida en `pick_start_end_point` se encargará de tomar las coordenadas.

```
[start_points,end_points] = pick_start_end_point(Wo');
```

### 4.3. Función Fast Marching

Es en este paso donde se coloca el peso del software con la llamada a ciertas funciones que pondrán a trabajar la maquinaria Fast Marching.

Una vez procesada la imagen y guardado las coordenadas de inicio y fin de la trayectoria a seguir por el robot, se procede a la expansión de las ondas por el entorno para posteriormente calcular el máximo gradiente que desvelará el camino a seguir, tal y como se explicó en el previo desarrollo del método.

- En primer lugar se recurre a la función desarrollada por Gabriel Peyre `perform_fast_marching_2d` que ejecuta el algoritmo de Fast Marching en el mapa de puntos `W3` mediante la llamada a la función `perform_front_propagation_2d`, la cual devuelve a la función madre 2 variables que serán muy importantes: D y S.

```
[D,S] = perform_fast_marching_2d(W3, start_points, options);
```

La primera de ellas 'D' almacena en un vector de 2 dimensiones la distancia que resta para alcanzar el punto objetivo, siendo la matriz que permite observar cómo ha sido la expansión de la onda del Fast Marching[2] hasta alcanzar el punto final. Como se puede

observar en la Figura 4.1 el resultado es un mapa que muestra los valores de distancia en escala de grises. Un tono oscuro corresponde a una cercanía de los puntos a la fuente, mientras que a mayor claridad de tonos existe mayor distancia.

Mientras que la segunda variable 'S' va computando el estado de cada punto. Y esto es, recordando los tres tipos de celda explicados se va analizando la información de los puntos y van clasificándose en: -1 representa un punto ya analizado *Congelada*, 0 una celda perteneciente a la *Narrow Band*, y 1 una celda *Desconocida.*
En la imagen se observa 'S' completamente negra y es porque la expansión de la onda ya ha alcanzado todos los puntos, dando por finalizado el algoritmo.

Fig. 4.2. De izquierda a derecha mapas 'D' y 'S' que contienen la expansión de la onda de la Figura 4.1 y sus distancias a la fuente. Figura de elaboración propia.

- Posteriormente tiene lugar el cálculo del gradiente descendiente que proporcionará el recorrido más corto, utilizando como argumento la matriz de distancia D obtenida de la anterior función `perform_fast_marching_2d`. Para ello se ejecuta la función `extract_path_2d` que devuelve un vector de 2 dimensiones con el "path".

```
path = extract_path_2d(D,end_points, options);
```

Puede resultar extraño que este conjunto de funciones que se encargan de calcular el "path" y expandir la onda del Fast Marching sean desarrolladas fuera del bucle principal, pero existe un motivo condicional para ello. Uno de los parámetros que arrojarán estás funciones previas será clave para el ciclo de vida del bucle que más adelante será explicado.

### 4.4. Generación del entorno

Una vez se tiene la imagen ya filtrada y procesada tiene lugar la generación del entorno visual que representará a tiempo real el desarrollo del sistema de navegación autónoma. En esta fase se establecen los mapas a visualizar, sus límites, los sensores… en resumidas cuentas el aspecto visual que percibe el usuario y que debe ser lo más agradable posible.

### 4.5. Bucle

El desplazamiento del robot se genera a través de un bucle que irá repitiendo las órdenes de movimiento hasta alcanzar el punto final '`end_points`' de la trayectoria. Esto implica la necesidad continua de una propagación de onda, un cálculo de la trayectoria restante, una actualización de los sensores y por ende el desplazamiento del objeto consecuente a las constantes que recibe.

El bucle comienza con una sentencia "while" estableciendo una doble y simultánea condición para poner fin a su ejecución. La primera es la repetición del "loop" o bucle hasta que alcance el punto final, y la segunda es que la longitud del recorrido sea mayor que 200 unidades, razón por la cual es necesaria la aplicación previa de la sintaxis del $FM^2$ que se ha comentado con anterioridad en el apartado 3.3.

```
while final_path==0 && length(path)>200,
```

En cada iteración, el método de Fast Marching$^2$ será aplicado utilizando las sentencias del apartado 4.3 para poder expandir la onda desde la fuente con el objetivo de llegar al punto final. De la misma manera también es necesario computar el recorrido restante para alcanzar el objetivo utilizando el gradiente.

Así se conseguirá ir actualizando la posición del robot y calcular en cada momento sus necesidades como la de modificación de trayectoria en caso de choque que será explicada en el sub apartado de este epígrafe.

Conviene destacar también que dentro del bucle vuelve a tratarse la imagen del mapa de una forma muy similar a cómo se había procesado previamente, con sus filtros y modificaciones. Esto responde al motivo de que para que el robot realice cualquier movimiento de avance, el sistema debe volver a observar el entorno y obtener su primer potencial para dar paso a la aplicación del método Fast Marching$^2$.

Por ello con esta repetición el sistema es capaz de volver a calcularse la nueva trayectoria, la cual en la mayoría de los casos es la misma que la calculada anteriormente teniendo en consideración que su recorrido será menor ya que el robot ha avanzado una posición.

Pero existen momentos en los que el radar detecta un objeto móvil próximo al robot y debe modificar su trayectoria, lo que explica la necesidad de repetir la función de preprocesamiento de imagen para calcularla.

En este bucle además se dibujará en cada iteración todos los sensores, posiciones y mapas alternativos mediante la llamada a sus respectivas funciones de dibujo tales como `plot_path_2d()` que dibuja el camino restante a recorrer por el robot, `dibuja_robot()` que dibuja la posición actual del robot, o `plot()` empleado para dibujar la posición de otros objetos móviles y la posición final del robot.

### 4.5.1. Choques

Cuando el robot inicia su camino debe estar pendiente del entorno y sus variantes para ser capaz de actuar a tiempo real si un suceso inesperado ocurriera, como pudiera ser la detección de otro objeto móvil por las inmediaciones que pudiera interferir en la trayectoria.

Con esta idea en el horizonte se implementa una función dentro del bucle que facilitará en gran medida este proceso. Se trata de convertir al robot en un obstáculo y que así lo entienda el programa.

Desde un primer momento se está trabajando con el mapa en blanco y negro, siendo un 0 las zonas negras vistas como las paredes del embalse u obstáculos, y de color blanco, 1, las zonas libres por las cuales puede desplazarse el robot sin colisionar. Entonces bien, se implementa una función que convierte a los objetos móviles y sus vecinos más cercanos (hasta 8 píxeles vecinos formando en la mayor medida de lo posible un cuadrado) en obstáculos "pintándolos" de negro a vista del programa en el mapa ya procesado. De esta manera el obstáculo móvil sufre las mismas transformaciones que el resto de obstáculos y expande su perímetro como medida de seguridad para evitar colisiones.

El radar permanece activo en cada iteración vigilando sus puntos vecinos por lo que cuando detecte un conjunto de puntos negros (que modelan al objeto móvil) que ha traspasado su radio de vigilancia, calculará la trayectoria alternativa para lograr esquivar al objeto y evitar así la colisión.

Para calcular la trayectoria nueva vuelve a emplearse las funciones del apartado 3.3 que se encargan de la aplicación del algoritmo Fast Marching[2].

# Capitulo 5. CASO PRÁCTICO. EMBALSE DEL ATAZAR

Para dar un aspecto realista al proyecto y comprobar los resultados de la aplicación de Fast Marching$^2$ se ha elegido el entorno del Embalse del Atazar, ubicado en el norte de la Comunidad de Madrid y embalsando al río Lozoya. Se trata de un embalse que cuenta con 1070 hectáreas y tiene una capacidad de 425,3 hm^3.

De esta manera se busca modelar el desplazamiento autónomo de un barco en un entorno cerrado desde un punto inicial a un lugar final con la condición real de que si es necesario debe modificar su trayectoria para evitar colisiones con objetos móviles.

## 5.1. Metodología

El objetivo es el desplazamiento de un barco en el embalse desde un lugar inicial a un punto final de manera autónoma. Para ello el sistema de navegación debe ser capaz de planificar una ruta lo más óptima posible esquivando obstáculos no solo fijos como pueden ser los bordes o pequeñas islas, sino también móviles representados por objetos independientes en el plano del desplazamiento. Esto quiere decir que el sistema debe estar continuamente alerta empleando la información que recibe de sus sensores para poder así dar una respuesta a tiempo real y recalcular la trayectoria si fuera necesario.

### 5.1.1. Pasos

El sistema cuenta con una interfaz del embalse a vista horizontal y dos objetos móviles. El primero de los ellos es el elemento imprescindible conocido como ‘barco principal’ que constituye el objeto cuyo desplazamiento autónomo es deseado y estudiado, mientras que el segundo o ‘nave secundaria’ modela una embarcación que se desplaza sobre la superficie del embalse de manera simultánea e independiente al barco principal y que podrá o no interferir en la trayectoria de éste.

En primer lugar deben indicarse los puntos inicial y final de la trayectoria a recorrer por el barco principal e inmediatamente después se definen los puntos del segundo objeto móvil.

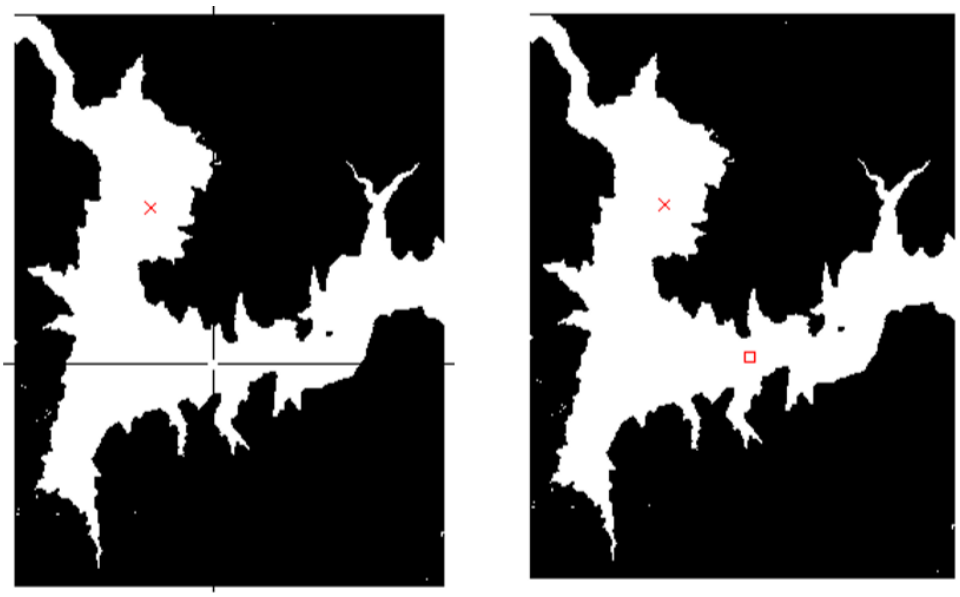
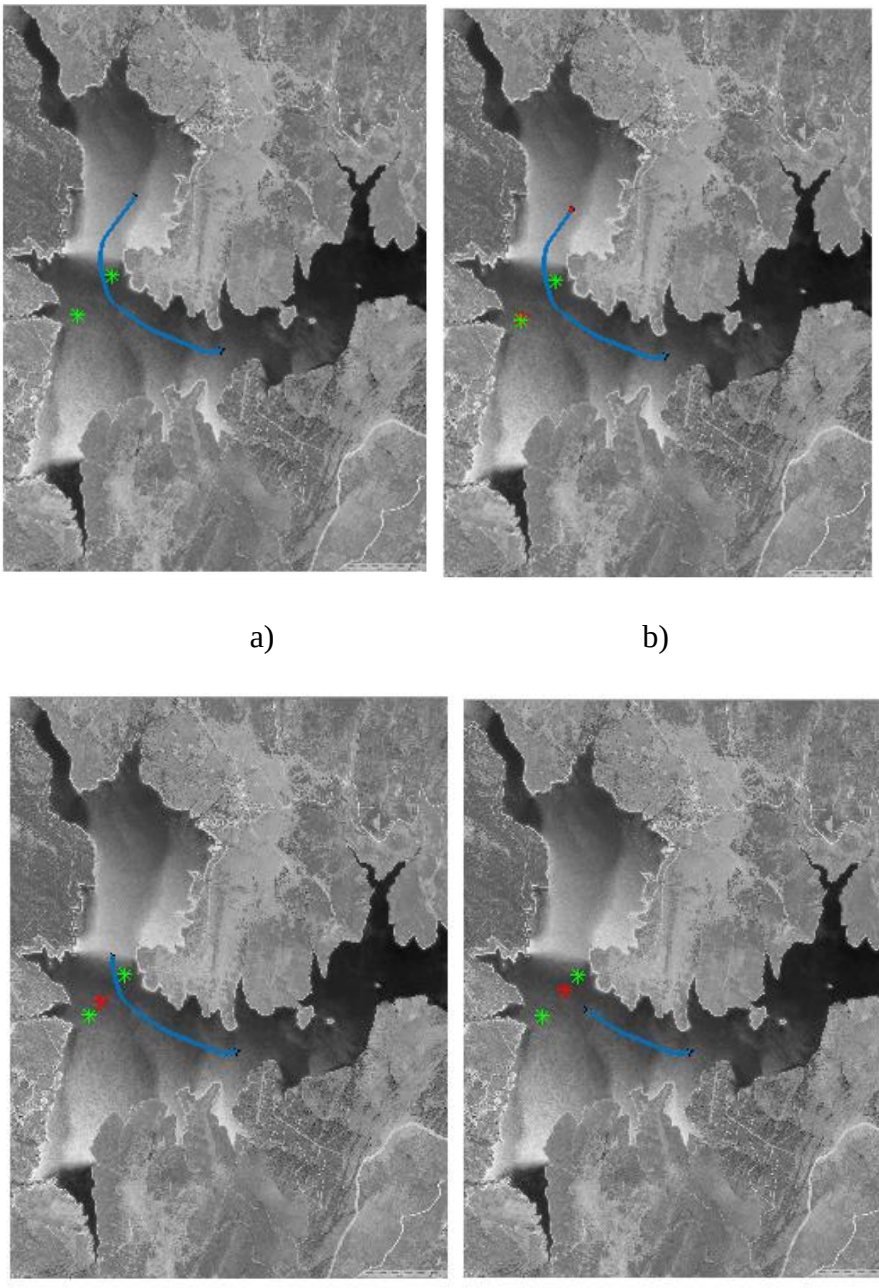


Fig. 5.1: Interfaz para seleccionar los puntos de inicio (x) y final del robot (cuadrado) con un puntero virtual. Figura de elaboración propia.

Gracias a un puntero virtual (Figura 5.1) podrá establecerse en el mapa en blanco y negro la ubicación donde comienza el movimiento del barco principal representado mediante una X roja, y el punto final de la trayectoria representado a su vez por un cuadrado rojo, para después hacer lo mismo con el objeto secundario.



a) b)

c) d)

Fig. 5.2: Representación sucesiva de la trayectoria a seguir por el robot (línea azul) con el paso del tiempo y puntos de inicio y final del segundo objeto móvil mediante asteriscos verdes. Un asterisco rojo representa a tiempo real el movimiento de este segundo objeto. Figura de elaboración propia.

Como se observa en la Figura 5.2, automáticamente se genera una trayectoria de color azul entre los puntos de inicio y final del barco principal que irá revisándose de manera continua ante posibles interferencias con otros elementos móviles. Este camino es el

resultado de haber aplicado la secuencia de preprocesado de la imagen y gradiente descritos en el apartado 4.

Por otra parte según se puede observar en la imagen, aparecen dos asteriscos de color verde que indican el comienzo y fin de la trayectoria segundo objeto, mientras que su posición a tiempo real será un asterisco de color rojo.

El movimiento del robot principal que es el punto rojo va "comiendo" la trayectoria azul a realizar según va avanzando con cada iteración, de esta manera el conjunto finalizará cuando el recorrido se haya realizado de manera completa. Si en algún momento el barco detectara algún objeto en su trayectoria que provocase una colisión, inmediatamente modificaría el "path" inicialmente calculado para esquivarlo como se explicará en el apartado 'Radar y obstáculos'.

Al finalizar el recorrido se dibuja una trayectoria roja revelando el camino final descrito por el barco principal; esto quiere decir que incluye las modificaciones que haya sufrido la trayectoria inicial.

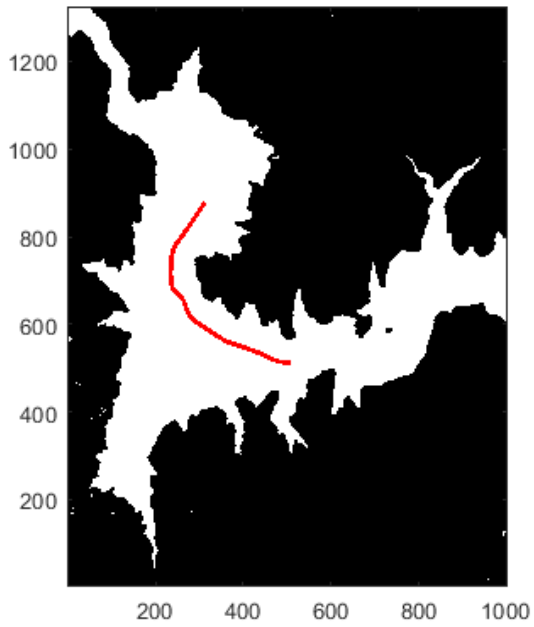



Fig. 5.3:Trayectoria final descrita por el barco principal. Figura de elaboración propia.

## 5.2. Fast Marching$^2$ en el Embalse del Atazar

En este apartado, de acuerdo con una mejor comprensión del proyecto tratará de unificarse el método Fast Marching^2 y la experiencia del usuario del sistema.

Partiendo de una imagen a color (Figura 5.4) con una vista de planta del embalse del Atazar se obtiene, al expandir los obstáculos, el primer potencial o mapa de velocidades tras una fase de preprocesamiento. Es conveniente recordar que la consecución de dicho mapa se constituye expandiendo los obstáculos con un coeficiente de seguridad por determinar que será explicado en el siguiente subapartado.

Inmediatamente después se computa la expansión de la onda generada por el método de Fast Marching$^2$ al aplicar la ecuación de Eikonal desde el punto inicial hasta llegar al objetivo. En este segundo potencial se puede observar una escala de grises donde los colores más oscuros corresponden a los más cercanos a la fuente de manera que el color se va clarificando conforme la distancia a la fuente aumenta.

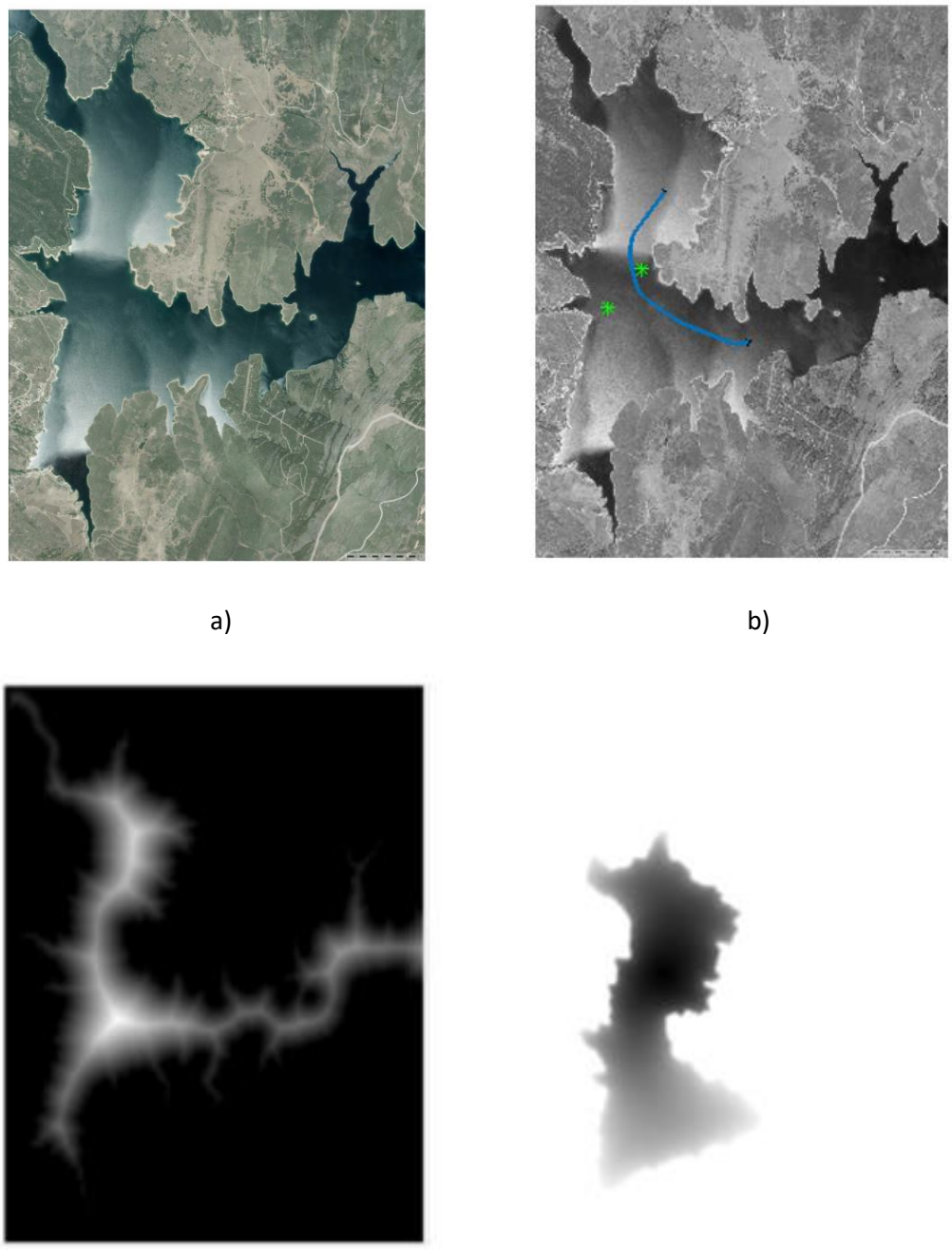



Fig. 5.4. (a) En la fila superior se observa el mapa original a color del embalse del Atazar a vista de pájaro. (b) A su derecha se muestra la trayectoria a realizar por el barco principal, mientras que en la fila inferior (c) se muestra el primer potencial de expansión de los obstáculos en distintos niveles de gris. Por último a la derecha (d) puede verse la expansión de la onda al aplicar el método Fast Marching. Figura de elaboración propia.

De esta manera se combinan ambos potenciales y se opera máximo gradiente para obtener el camino que será capaz de calcular el recorrido óptimo, el cual combina velocidad máxima con mínimo recorrido y una seguridad aceptable.

### 5.2.1. Coeficiente de seguridad

El método Fast Marching[2] busca la trayectoria óptima entre dos puntos, lo que significa que debe existir un balance entre velocidad y seguridad del camino a recorrer por el barco principal.

Como anteriormente se ha explicado, la velocidad del robot es totalmente proporcional a la distancia del mismo a los obstáculos, por lo que a mayor separación de los obstáculos, más rápido avanzará el robot.

Para ello se establece el coeficiente de seguridad, el parámetro esencial que participa en la generación de mapas de velocidades durante la fase de preprocesamiento de la imagen. Es el principal responsable del primer potencial cuando se produce la expansión de una primera onda desde los objetos.

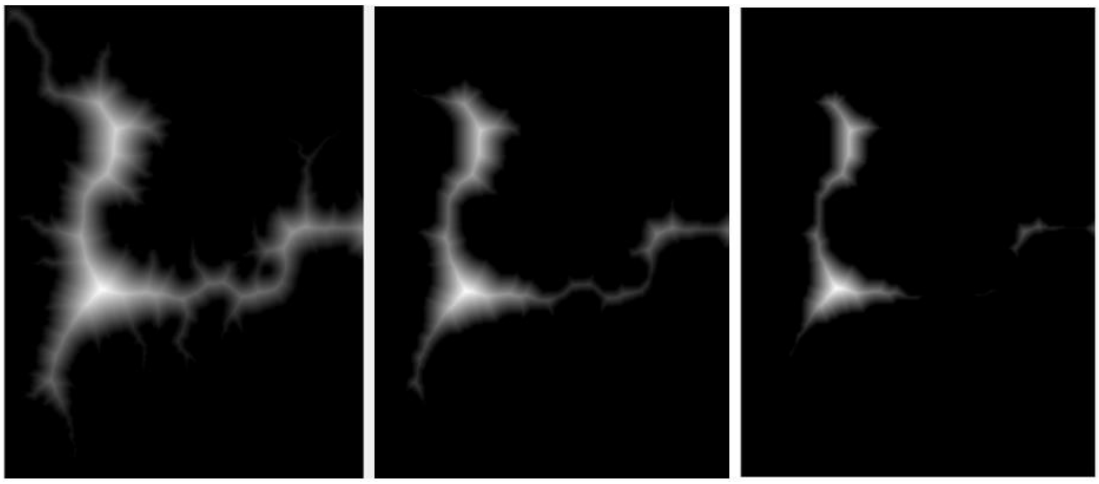


Fig. 5.5. Variación del coeficiente de seguridad. De izquierda a derecha el mapa con un coeficiente de seguridad 0, en el centro coeficiente 30 y después mapa con coeficiente 50. Figura de elaboración propia.

Como puede observarse en la Figura 5.6 el coeficiente de seguridad determina la magnitud de expansión del obstáculo. De esta manera si se desea una seguridad alta

durante el recorrido deberá ir aumentándose el coeficiente de seguridad como sucede en la imagen con coeficiente 30, mientras que al descender dicho coeficiente las zonas blancas por donde el barco principal pueda desplazarse son de mayor amplitud. Esto desemboca en que el barco principal tenga una cantidad superior de trayectorias a realizar, es decir un mayor abanico de caminos posibles.

Pero debe tenerse en cuenta el poder del coeficiente de seguridad ya que por el contrario si es demasiado elevado se corre el riesgo de que los obstáculos se expandan tanto que coman demasiados espacios libres y mengüen el mapa de tal manera que sea imposible alcanzar apenas algún recorrido como sucede en la tercera imagen de coeficiente 50.

## 5.3. Radar y obstáculos (Choques)

El sistema desarrollado trabaja a tiempo real, lo que provoca la necesidad de estar continuamente atento a los cambios en el entorno. Esto significa que el sistema no solo debe calcular la trayectoria inicial, sino que además es preciso estar alerta ante cualquier modificación que surja en el desarrollo del camino. Para ello se implementa un radar en el sistema que mide las constantes y toma decisiones en función de lo que las mismas le transmitan.

El radar es un elemento 360º muy importante en el conjunto pues es la forma de comunicarse con el entorno. Dicho elemento se representará con una circunferencia verde con el centro del objeto de color rojo como se observa en la Figura 5.7 (a) y (b).

El camino que va recorriendo el barco sigue siempre la dirección de máximo gradiente pues debe asegurar el recorrido más óptimo, pero ¿qué sucederá cuando un objeto desconocido entre en la lectura del radar?

Durante unos instantes el barco debe abandonar la trayectoria inicialmente generada para calcularse una nueva que esquive al objeto detectado desde el punto actual al punto final. Para ello empleará también la dirección de máximo gradiente representado al obstáculo como un punto pequeño verde dentro de la circunferencia que describe el radar (Figura 5.7 (b)).

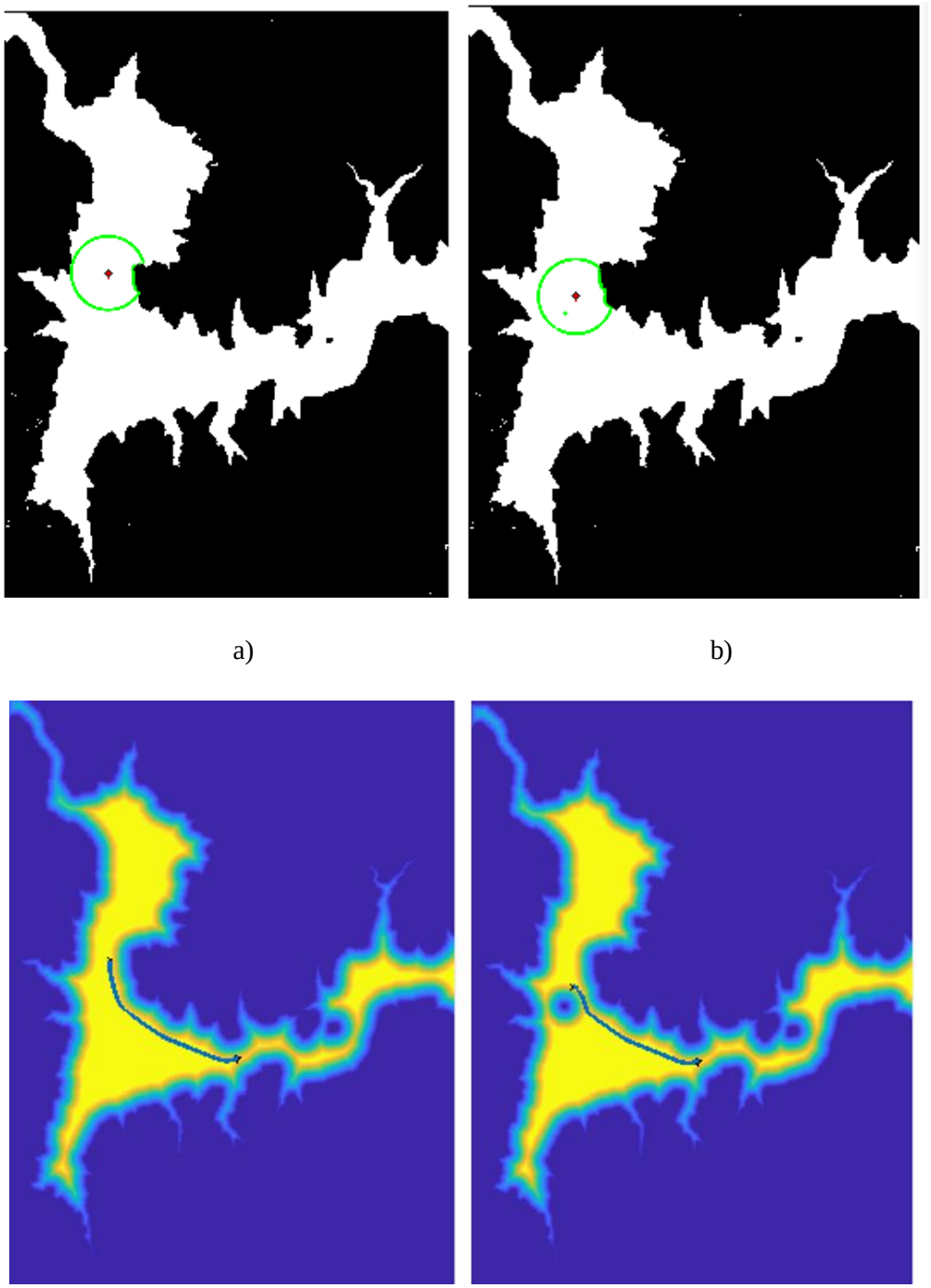



Fig. 5.6: Comparación de mapas sin detección de objetos (columna izquierda) vs mapas con aparición de objetos móviles en las proximidades (columna derecha). Figura de elaboración propia.

En la imagen 5.7 (c) se observa el mapa y el gradiente a seguir sin ningún tipo de interrupción, mientras que en la imagen (d) puede comprobarse la aparición de un objeto móvil en las inmediaciones del radar y su representación mediante círculos concéntricos en diferentes tonos de color azul. Esta escala es idéntica a la de grises pero en diferente tonalidad, siendo el azul más oscuro sinónimo de velocidad $F$=0 y el azul más clarito resultado de una velocidad permitida algo mayor.

Hablando sobre el recorrido generado, la nueva trayectoria es prácticamente idéntica a la inicial salvo por la evasión del obstáculo en la dirección de máximo gradiente.

# Capitulo 6. ENTORNO SOCIO-ECONÓMICO Y MARCO REGULADOR

En una época en la que la innovación y optimización desempeñan un papel tan importante es conveniente una predisposición íntegra por adaptarse a estos cambios tan espontáneos.

Hace varias décadas, allá por 1940, un ingeniero podía abarcar fácilmente un 40% del campo técnico existente, mientras que a día de hoy un ingeniero tiene suerte si conoce en profundidad algo más del 5%. Esto es muestra de que la tecnología avanza a un ritmo vertiginoso, y lo más importante de todo: no va a abandonar este camino.

Por ese motivo se requiere de una sociedad evolutiva que sea capaz de crecer junto con la tecnología para sacarle el máximo rendimiento y mejorar la calidad de vida.

## 6.1. Impacto socio-económico

Una de las principales líneas de dirección que el mundo tecnológico sigue es la reducción de costes, y de manera directa o indirecta este fenómeno implica una optimización de los sistemas de operaciones de cada empresa. Una de las herramientas para llevarlo a cabo es la automatización.

La automatización de los procesos promueve la atribución de tareas sistemáticas y mecánicas a robots para que el humano se encargue de labores de diferente naturaleza, las cuales un cerebro electrónico no puede desarrollar. Esto permite reducir tiempos y en algunos casos mejorar el producto final, pero inevitablemente conlleva a una destrucción de empleo, quizá no tan exagerada como algunos artículos recogen pero sí significativa.

Siguiendo la línea de dirección de rumbo empresarial los algoritmos de planificación de trayectorias proveen un sistema para desplazar embarcaciones entre dos puntos sin la necesidad de un conductor que lo dirija. Lo que supone la reducción de personal a bordo del barco, como el patrón porque si el barco es capaz de desplazarse de forma independiente, parte de la tripulación sería prescindible durante el trayecto.

En algunos casos quizá no sea tan necesario para la compañía despedir a esos trabajadores sino simplemente una reducción de tareas laborales que repercutirá directamente en su salario.

No obstante, esta última consideración abre un campo muy extenso porque no tendrá la misma repercusión la aplicación del método FM en un transatlántico que trasporte mercancía, que en una barca de uso personal y lúdico en un embalse donde no generaría ningún problema de empleo.

De lo anterior se desprende la cuestión de si es ético que por beneficio económico una empresa sustituya a una persona por una máquina. Por una parte ha de comprenderse el deseo de las compañías de sacar el máximo beneficio posible a su actividad laboral, pero por otro lado debe pensarse en las personas, y por ende sus familias, a las que una máquina sin necesidades biológicas desplaza de su puesto de trabajo.

Sea cual sea la solución, la responsabilidad claramente recae en manos de los dirigentes políticos, de que sean capaces de asegurar un crecimiento sostenible y gradual de la tecnología, y a su vez de promover la sociedad evolutiva que avance de la mano de la misma.

## 6.2. Presupuesto

[31] [32] Para el cálculo del presupuesto va a tenerse en cuenta las horas invertidas en el desarrollo del proyecto, los medios físicos empleados y las licencias software.

La mano de obra puede dividirse en horas de desarrollo y horas de supervisión. Las primeras corresponden a tareas de búsqueda de información y planteamiento de soluciones, mientras que las segundas representan el tiempo empleado en el cumplimiento de los requisitos y su exactitud.

Tabla 6.1 COSTE TOTAL MANO DE OBRA. Tabla de elaboración propia

| **Tipo de mano de obra** | **Cantidad (horas)** | **Coste unitario (€/hora)** | **Coste total (€)** |
|---|---|---|---|
| Trabajos de desarrollo | 280 | 10 | 2000 |
| Trabajos de supervisión | 70 | 10 | 1500 |
| Subtotal | | | 3500 |

A su vez, el ordenador en el que se ha desarrollado la totalidad del proyecto se incluye en el presupuesto.

Tabla 6.2 COSTE TOTAL MATERIALES. Tabla de elaboración propia

| **Materiales** | **Cantidad (unidades)** | **Coste total (€/unidad)** |
|---|---|---|
| Ordenador portátil HP TouchSmart 15-D019SS | 1 | 500 |

Por último, se han empleado dos licencias diferentes de software que son la de Matlab R2019b y la de Microsoft Office Word 2016.

La primera se ha empleado para la implementación de la totalidad del código de la aplicación y debido al acuerdo que mantiene la Universidad Carlos III con la compañía MathWorks su coste es nulo para el estudiante. No obstante, se incluirá el precio de la licencia que la universidad ha costeado.

Mientras que la segunda ha servido para el desarrollo de la memoria.

Tabla 6.3 COSTE TOTAL LICENCIAS DE SOFTWARE. Tabla de elaboración propia

| **Licencias de software** | **Tipo** | **Cantidad (unidades)** | **Coste unitario (€/unidad)** | **Coste total (€)** |
|---|---|---|---|---|
| Matlab R2019b [33] | Licencia de estudiante | 1 | 70 | 70 |
| Microsoft Office Word 2016 [34] | Licencia de estudiante | 1 | 106 | 106 |
| Subtotal | | | | 176 |

Con todos los subtotales de cada grupo es posible calcular el presupuesto total del proyecto.

Tabla 6.4 COSTE TOTAL DEL PROYECTO. Tabla de elaboración propia

| **Subtotales** | **Cantidad** | **Coste total (€)** |
|---|---|---|
| Mano de obra | 1 | 3500 |
| Materiales | 1 | 500 |
| Licencias de software | 1 | 176 |
| TOTAL | | 4176 |

## 6.3. Marco regulador

[35] [36] Cuando un sistema diseñado por el humano se desplaza de manera autónoma por el espacio, debe tenerse en cuenta que el robot no puede pagar una sanción si causa graves desperfectos ni cumplir condena. Por ello se está trabajando para estipular leyes que regulen el uso de los mismos.

Dando el primer paso muchos gobiernos han puesto límites al empleo de robots, colocando altos impuestos a las empresas en función del número de máquinas automatizadas que empleen. De esta manera evitan que las compañías puedan sustituir de manera masiva muchos trabajadores por máquinas.

La seguridad representa un ámbito de gran importancia en la utilización de los robots, y un barco autónomo debe procurar la máxima en cada desplazamiento. El sistema de control del robot tiene que estar preparado para detener la embarcación en caso de emergencia, empleando para ello las constantes del entorno que recibe a través de sus sensores.

Un error podría desembocar en un accidente con otra embarcación o directamente con el medio, puesto que a pesar de esquivar los obstáculos el sistema no está diseñado para reaccionar en distancias muy muy cortas. Esto se traduce en que si en un primer momento el robot evita a la embarcación que se le aproxima girando para un lado pero a su vez la embarcación tripulada, en un intento de anticiparse, gira para ese mismo lado, la colisión sería inevitable. De la misma manera cabe la posibilidad de que las trayectorias de dos embarcaciones no tripuladas se crucen, y si cuando están próximas, en el momento de cambiar su trayectoria giran hacia el mismo lugar, la colisión al igual que en el ejemplo sería inevitable.

En ese momento debe delimitarse la adjudicación de responsabilidades, viendo el grado de culpabilidad de las partes implicadas y por supuesto la gravedad de las consecuencias ocasionadas. Pero actualmente, al existir tantos escalones en el ciclo de vida del robot, el grado de culpabilidad se diluye entre los proveedores, los diseñadores, los programadores y por supuesto la persona que maneja el aparato si es que una de las partes es tripulada.

De esta manera se abre un campo de dimensiones gigantescas con multitud de variantes que debe ser cubierto por los organismos judiciales para dar solución.

La Unión Europea está trabajando para alcanzar una regulación exclusiva para los robots. En una primera instancia refleja que deben cumplirse los tres principios planteados por Isaac Asimov en 1992 que se resumen en que un robot no debe hacer daño a la humanidad.

# Capitulo 7. CONCLUSIONES

La metodología Fast Marching resuelve de manera eficiente los cálculos de trayectorias entre dos puntos. En un intento de asemejarse al mundo real, el sistema desarrolla correctamente el movimiento que le permite evitar obstáculos móviles si éstos se aproximan, y completar la trayectoria.

La fusión de los conceptos teóricos con la experiencia práctica mostrada en el proyecto permite introducir al lector en el mundo de la planificación de trayectorias teniendo en cuenta las premisas básicas de eficiencia, seguridad y fiabilidad que el mismo desarrolla. Estas consideraciones son controladas mediante técnicas de tratamiento de imágenes y métodos matemáticos que logran adaptar la información del entorno que reciben los sensores.

De cara al futuro quizá se podría reducir aún más la velocidad de computación con la mejora de la base matemática del algoritmo, por ejemplo, una optimización del gradiente o alguna técnica que lo ejecutara más rápido resultaría interesante.

Además se podría añadir la condición real de navegar a contracorriente, es decir, que las olas de agua le impidan avanzar como debería. Se recuerda que en este proyecto la velocidad de las olas es siempre positiva $F > 0$, por lo que la marea nunca va en dirección opuesta a la que el barco sigue.

Dando el salto al gigante marino y para ser aún más perfeccionistas, podría incluirse las dimensiones de la embarcación ya que no es lo mismo calcular una trayectoria para una barca de madera y reducidas dimensiones que para un crucero de 70 metros de altura. De la misma manera se necesita conocer la orografía del mar u océano para que los barcos grandes no se acercaran a la costa y evitar el choque de la carena con el suelo.

Como consecuencia del rápido avance de la tecnología la introducción de estas posibles mejoras no tardará en llegar, lo que pone de manifiesto la importancia del cálculo de trayectorias en la evolución de los transportes y la gran aportación que el método Fast Marching realiza en ello.

Con un coste computacional no muy elevado y una fiabilidad patente, la solución a tiempo real que proporciona el método cumple de manera adecuada las premisas básicas y genera trayectorias eficientes y completas totalmente aplicables al caso práctico expuesto.

# BIBLIOGRAFÍA


[1] S. TANG, N. ISMAIL, W. KHAKSAR y M. ARIFFIN, «A Review on Robot Motion Planning Approaches,» *Pertanika J. Sci. & Technol.*, vol. 20, nº 1, pp. 15-29, 2012.

[2] S. M. LAVALLE, Planning Algorithms, Illinois: Cambridge University Press, 2006, p. 842.

[3] D. FERGUSON, M. LIKHACHEV y A. STENTZ, «A Guide to Heuristic-based Path Planning,» Pittsburgh, USA, 2005.

[4] S. M. LAVALLE, «Motion Planning: The Essentials,» *IEEE Robotics and Auto*, vol. 18, nº 2, pp. 108-118, 2011.

[5] J. BARRAQUAND y J.-C. LATOMBE, «Robot Motion Planning: A Distributed Representation Approach,» Stanford, USA, Junio 1989.

[6] K. HAUSER, «Configuration Space,» Indiana, USA.

[7] W. BURGARD, C. STACHNISS, M. BENNEWITZ y K. ARRAS, «Introduction to Mobile Robotics. Robot Motion Planning,» Friburgo, 2011.

[8] M. ZEFRAN, «Continuous Methods for Motion Planning,» Penn Libraries, Pensilvania, Diciembre 1996.

[9] H. CHOSET, «Cell Decompositions,» de *Robotic Motion Planning*.

[10] O. KHATIB, «Real-time obstacle avoidance for manipulators and mobile robots,» de *Autonomous Robot Vehicles*, vol. 5, Nueva Jersey, Springer, 1986, pp. 396-404.

[11] H. CHOSET, «Potential Functions,» de *Robotic Motion Planning*.

[12] Y. K. HWANG y N. AHUJA, «A Potential Field Approach to Path Planning,» *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 8, nº 1, pp. 23-32, Feb 1992.

[13] «Why is there a negative gradient?,» 2018. [En línea]. Available: https://www.quora.com/Why-is-there-a-negative-gradient.

[14] «What is gradient descent algorithm?,» 2018. [En línea]. Available: https://www.quora.com/What-is-gradient-descent-algorithm.

[15] «What is an intuitive explanation of gradient descent?,» [En línea]. Available: https://www.quora.com/What-is-an-intuitive-explanation-of-gradient-descent.

[16] H. SAFADI, «Local Path Planning Using,» McGill University School of Computer Science, 2017.

[17] H. CHOSET, «A* and D* Search,» de *Robotic Motion Planning*.

[18] E. DIJKSTRA, de *A note on two problems in connexion with graphs*, vol. 1, Numerische Matematik, 1959, pp. 269-271.

[19] J. B. KRUSKAL jr, «On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem,» de *Proceedings of the American mathematican Society*, vol. 1, Feb. 1956, p. 48_50.

[20] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST y C. STEIN, «Depth-first search,» de *Introduction to Algorithms*, 2ª ed., MIT Press y McGraw-Hill, 2001, pp. 531-539.

[21] H. CHOSET, «Bug Algorithms,» de *Robotic Motion Planning*.

[22] S. GARRIDO, L. E. MORENO y J. V. GOMEZ, «Motion Planning Using Fast Marching Squared Method,» de *Motion and Operation Planning of Robotics System: Background and Practical Approaches*, G. Carbone y F. Gomez Bravo, Edits., Springer, Abril 2015, pp. 223-248.

[23] S. GARRIDO, L. E. MORENO, J. V. GOMEZ y A. VALERO GOMEZ, «The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories,» *IEEE Robotics & Automation Magazine*, pp. 111-120, Diciembre 2013.

[24] S. GARRIDO, L. E. MORENO y M. IZQUIERDO, «Calculo de trayectorias para vehiculos autonomos marinos (USV) utilizando el metodo Fast Marching Square sometido a campo vectorial,» de *XXXVII JORNADAS DE AUTOMATICA*, Madrid, Septiembre 2016.

[25] M. L. MAMANY, «Ecuación de la Eikonal,» 18 Octubre 2016. [En línea]. Available: https://es.scribd.com/document/327966300/ecuacion-Eikonal.

[26] S. GARRIDO, L. E. MORENO, M. FERNANDO y D. ÁLVAREZ, «Quality Study of Robot Trajectories Based on the Anisotropic Fast Marching Method,» de *Advances in Intelligent Systems and Computing*, Noviembre 2017.

[27] «Fast Marching Methods: A boundary value formulation,» 2006. [En línea]. Available: https://math.berkeley.edu/~sethian/2006/Explanations/fast_marching_explain.html.

[28] S. GARRIDO, L. E. MORENO y A. BARGUEÑO JUAREZ, «Modelización simulada en tiempo real de la evolución de un incendio mediante el Método Fast Marching,» de *XXXVII JORNADAS DE AUTOMATICA*, Madrid, Septiembre 2016.

[29] S. GARRIDO, L. E. MORENO, J. V. GOMEZ y D. ALVAREZ, «Fast Marching based Globally Stable Motion Learning,» *Soft Computing*, Diciembre 2015.

[30] G. Peyre, «Toolbox Fast Marching,» 27 Junio 2009. [En línea]. Available: https://es.mathworks.com/matlabcentral/fileexchange/6110-toolbox-fast-marching. [Último acceso: 25 Abril 2019].

[31] Facultad de Ciencias y Artes de la Universidad Católica de Ávila, «Elaboración de un trabajo de fin de grado en ingeniería: criterios y recomendaciones,» 2017-2018. [En línea]. Available: https://www.ucavila.es/images/files/GuiaAcademica/17-18/comun/TFG/CRITERIOS.ELABORACION.TFG.INGENIERIAS.17.18.pdf.

[32] E. JANÉ SOLER, «Presupuesto,» de *Estudio de un sistema de UAV mediante masas móviles*, Barcelona, 2015, pp. 1-9.

[33] «MathWorks pricing and licensing,» 2019. [En línea]. Available: https://es.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=student.

[34] « Microsoft Office comprar licencia para el hogar,» 2016. [En línea]. Available: https://products.office.com/es/compare-all-microsoft-office-products?tab=1.

[35] M. J. SANTOS GONZALEZ, «Regulación legal de la robótica y la inteligencia artificial: Retos del futuro,» *Revista jurídica de la Universidad de León*, vol. 1, nº 4, pp. 25-50, 2017.

[36] R. LIMÓN, «¿Quién es responsable de los daños que cause un robot?,» 25 Abril 2018. [En línea]. Available: https://elpais.com/tecnologia/2018/04/24/actualidad/1524562104_998276.html.