Departamento de ingeniería de sistemas y automática

Universidad Carlos III de Madrid

Desarrollo de interfaz gráfica para integrar distintos algoritmos de planificación de trayectorias

Autor: David Palomo Pérez

Tutor: David Álvarez Sánchez

Titulación: Grado de Ingeniería electrónica industrial y automática.

Leganés, Septiembre de 24

Universidad
Carlos III de Madrid

Agradecimiento

Este proyecto quiero dedicárselo a mi madre, aquella que ha hecho de mí la persona que soy, educándome, animándome y siempre intentando darme lo mejor. Sin ella, esto no hubiera sido posible.

También quiero dar las gracias a mis tío/as, mis primo/as y mi abuelo.

Gracias a José y Nina, así como a sus hijas Ainhoa y Lidia por ser parte de imprescindible de mi vida. Gracias a mis pesados vecinos, Isidoro y Carmen, por aguantar mi mal humor.

Gracias a mis amigos de toda la vida y a mis compañeros de la universidad por los maravillosos años que hemos compartido.

Por último, gracias a mis abuelas Altamira y Mari, que siempre me acompañan.

Resumen

La robótica en los últimos años se ha centrado en aplicaciones que tienen como finalidad la exploración o la incursión en terrenos. Para esta finalidad, en los últimos años se han desarrollado algoritmos encargados de planificar trayectorias.

En este proyecto se propone la integración de tres algoritmos distintos formados por el algoritmo de planificación de trayectorias Fast Marching Method. Estos tres algoritmos serán integrados en dos interfaces gráficas, una dedicada al ajuste de los parámetros necesarios y otra en la cual se mostraran los resultados de las demostraciones.

Estos tres algoritmos son: Fast Marching Square, mejora el método Fast Marching, obteniendo una trayectoria rápida y segura; Fast Marching Learning, dedicado al aprendizaje de trayectorias; y Fast Marching Square Formations, el cual representa una formación de robots y se analiza el movimiento de estos a través de diferentes obstáculos.

Abstract

Robotics in recent years has focused on applications that are intended exploration or land incursion . For this purpose , in recent years have developed algorithms planners trajectories.

In this project the integration of three different algorithms formed by the path planning algorithm Fast Marching Method is proposed. These three algorithms will be integrated into two graphical interfaces , one dedicated to setting the required parameters and one in which the results of the demonstrations will be shown .

These three algorithms are: Fast Marching Square, improves Fast Marching method , obtaining a fast and safe path; Fast Marching Learning, dedicated to learning paths ; and Fast Marching Formations Square , which represents education of robots and motion of these is analyzed through different obstacles.

ÍNDICE

Sección 1: Introducción al Proyecto fin de Grado

1.1 Motivación

La robótica es la rama de la ingeniería que se dedica al diseño, construcción y disposición estructural entre otros. Estas tareas tienen en común la aplicación de esfuerzos físicos e inteligencia.

En los años 50 se creó el primer robot. Y con ellos se comenzó a estudiar la manera de emular el procesamiento de la información por parte de los humanos, lo que daría lugar a la inteligencia artificial. Hasta entonces se les denominaba autómatas, puesto que se dedicaban a realizar trabajos repetitivos dentro de la industria.

En la actualidad, debido a los numerosos avances tanto en hardware como en software, las aplicaciones de estos en diversos campos son ilimitadas, como por ejemplo: en la industria (soldadura, moldeado en la industria plástica), en los laboratorios (preparación de muestras), en el espacio (exploración espacial)…

Aplicaciones como la exploración espacial, necesitan de una planificación de trayectorias. La planificación de trayectorias es crear un algoritmo capaz de encontrar una ruta a partir de un punto inicial y uno final, siempre y cuando existiera. Una vez conseguido este objetivo, el siguiente paso era, además de encontrar el camino entre un punto inicial y final, es que fuera el más cortó y seguro. De estas funciones se encarga Fast Marching Method y sus derivados.

1.2 Objetivo

El objetivo de este proyecto es la integración de los algoritmos de Fast Marching en dos interfaces gráficas de usuario en MATLAB para poder generar demostraciones sobre la planificación de trayectorias

Para poder realizar esto, partimos de tres algoritmos: Fast Marching Square, que se encarga de calcular trayectorias teniendo en cuenta el camino más corto y la seguridad de la trayectoria; Fast Marching Learning, cuya principal función es reproducir trayectorias en diferentes condiciones; y Fast Marching Square Formations, generador de trayectorias con una formación de 3 robots . Estos algoritmos deben ser integrados en dos interfaces con la herramienta de MATLAB, GUIDE, con la cual se realizara la interfaz.

Los objetivos generales propuestos para este proyecto son:

- Introducción a la programación en MATLAB.
- Aprendizaje de diseño de interfaces gráficas con GUIDE.
- Estudio del algoritmo de planificación de trayectorias.
- Conseguir reproducir las demostraciones del los tres algoritmo.
- Poder desarrollar mapas de demostración para poder posteriormente ser cargados en los tres algoritmos.

- Poder guardar las imágenes resultantes de cada algoritmo.
- Desarrollar la posibilidad de guardar íntegramente los ensayos en un espacio de trabajo.
- Desarrollar la posibilidad de guardar trayectorias de aprendizaje en el algoritmo Fast Marching Learning para poder reproducir la demostración en diferentes condiciones.
- Estudio y compresión de los resultados obtenidos sobre el proyecto.

1.3 Fases de desarrollo del proyecto

El proyecto ha sido desarrollado de acuerdo a las siguientes fases:

1. Fase de estudio: análisis del problema planteado y búsqueda de posibles soluciones. En esta fase primero se plantea el problema, que es lo que se quiere y por qué. Una vez resuelto esto, se analizan todas las posibles soluciones, recopilando la mayor cantidad de información posible y analizándola.

2. Fase de programación: En esta fase se pone en práctica la/s soluciones estudiadas, comprobando que se van cumpliendo los objetivos y el diseño marcados en la primera fase. En esta fase se adapta cada algoritmo a la interfaz de usuario por separado. Para ello, se siguen las pautas siguientes:

    2.1. Estudio: estudiamos cómo funciona el algoritmo, ejecutando una pequeña parte, para ser consciente de los elementos necesarios en nuestra interfaz.
    2.2. Desarrollo: En esta fase se implementa en MATLAB/GUI la solución desarrollada anteriormente. En esta fase también se realizan pruebas de funcionalidad y corrección de errores que puedan surgir al programar o derivados del código.

3. Fase de evaluación: Valoración de los resultados obtenidos. En esta fase, ya hay una versión funcionando de la interfaz, la cual es estudiada y se sugiere cómo es posible mejorarla. Es una fase de depuración y optimización.

4. Fase de documentación: fase en la que se trascribe todo el trabajo realizado en las fases anteriores a la memoria del proyecto.

1.4 Estructura de la memoria

La memoria del proyecto sigue la siguiente distribución:

- *Sección 1: Introducción al proyecto fin de grado*: introducción al proyecto, explicando el motivo de éste, cual ha sido la metodología de trabajo y como está estructurado.
- *Sección 2: Marco teórico, algoritmo de alto nivel:* explicación de los algoritmos FM2, FML y FM2F desde el punto de vista teórico, en qué consiste, si son importantes y p para que se emplean.
- *Sección 3: Matlab:* introducción al programa MATLAB, más concretamente a su herramienta GUIDE, explicando el entorno de trabajo que se ha utilizado y como ha sido empleado cada momento.
- *Sección 4: Desarrollo del proyecto*: Explicación detallada del trabajo, analizando la implementación, justificando la distribución, analizando el funcionamiento y los resultados obtenidos.
- *Sección 5: Ejecución y conclusiones*: ejecución de las interfaces, realizando demostraciones, para poder analizar los resultados obtenidos, además de analizar las posibles mejoras o diferentes opciones que se habían pensado. En este apartado se repasara si se han alcanzado los objetivos.
- *Sección 6: Anexo:* en este apartado se realiza un presupuesto del proyecto.
- *Sección 7: Referencias:* recapitulación de todo el material consultado para la realización del proyecto.

Sección 2: Marco Teórico

2. Fast Marching Method (FMM)

Fast Marching Method (FMM) es un método de desarrollo para la planificación de trayectorias de robots. En este proyecto habrá tres variaciones de este algoritmo, como ya se ha mencionado antes: Fast Marching Square (FM2), Fast Marching Learning (FML) y Fast Marching Square Formations (FM2F).

FMM utiliza el movimiento de una onda física (una expansión de una onda) para obtener el camino más corto y así poder obtener también su velocidad. Para ello calcula el tiempo T que debe tardar una onda en llegar a cada punto del espacio. Puede haber más de un punto de origen, que generara una ola. La evolución de la onda lleva consigo una velocidad F, la cual no tiene porque ser igual en todas las partes. Este movimiento de la onda es definido por la ecuación Eikonal.

$$1 = F(x)|\nabla T(x)|$$

Donde F(x) es la velocidad de expansión de la onda, T(x) es el tiempo que tarda en llegar la onda y x es la posición. Una característica importante es que la función T(x) de una onda que crece originado en un solo punto y tendrá un solo mínimo global y no tendrá ningún mínimo local**.** De tener algún mínimo local, supondría que un punto tiene un valor T inferior a un punto cercano al origen, lo que resulta imposible.

Si nos fijamos en la figura 1, podemos ver como se parte de dos puntos, los cuales tendrán un tiempo T =0. Según va evolucionando la onda, observamos que ambas se expanden de forma individual. El proceso iterativo se realiza a la misma velocidad con la cual la onda física crece. En este tipo de mapas se pueden encontrar tres tipos de células:

- Congeladas (negro): células por la cuales ya se ha expandido la onda y tiene un valor T fijo.
- Próximas (gris): célula próxima a ser alcanzada en por la onda en siguientes iteraciones, con un valor T que puede variar.
- Desconocidas (blanco): célula sin un valor conocido T, ya que la onda no ha llegado todavía.

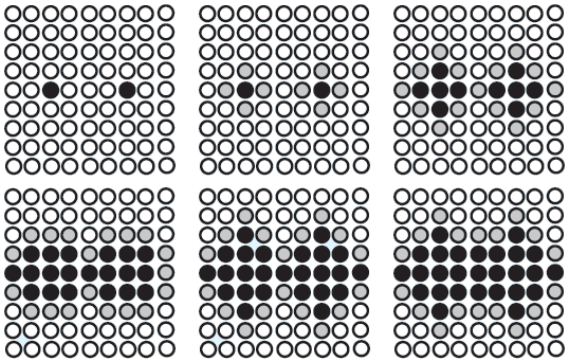


FIGURA 1. "*Expansión de la onda de dos puntos*"

Si nos fijamos en figura 2, se representa la evolución de onda en 3D, teniendo como tercer eje el tiempo T.

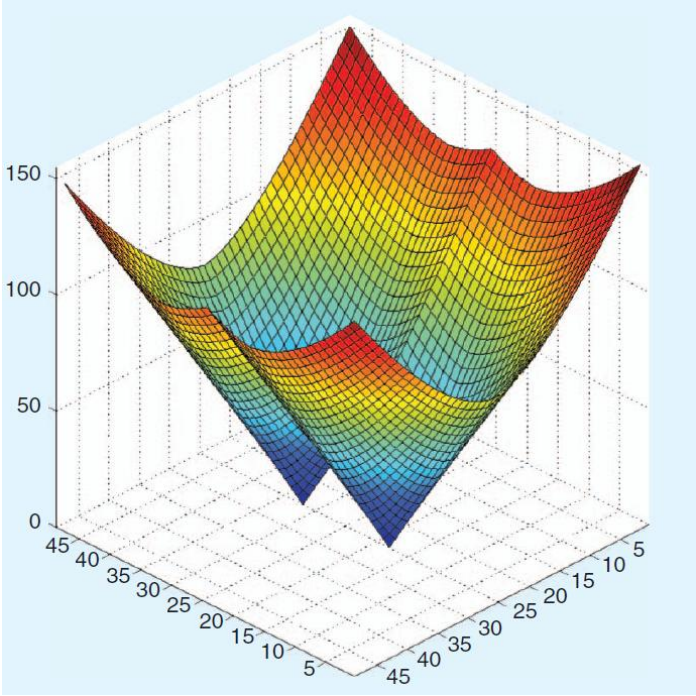


FIGURA 2. "*Evolución espacial de la onda en función del tiempo*"

En la figura 3 hay 4 imágenes, las cuales representan la evolución del método FMM. La imagen correspondiente al mapa sin ningún tipo de alteración. En la imagen 2 de la figura se representa un mapa de grises con la expansión de la onda originada en el punto inicial (más oscuro) y finalizando en el punto final (más claro) con una dilatación de los obstáculos. De la expansión de la onda por el mapa definido se generan las isocurvas (imagen 4).

Las isocurvas son la unión de todos los puntos que se pasan en el mismo instante de tiempo. Si calculamos la pendiente máxima en cualquier dirección normal a la isocurva, obtendríamos la dirección de la curva al expandirse (Imagen 3). En la imagen 3 se representa el mapa de distancias en 3D, obtenido calculando el gradiente para x e y. También se observa que solo tenemos un máximo y un mínimo global, por lo tanto la solución obtenida es única.

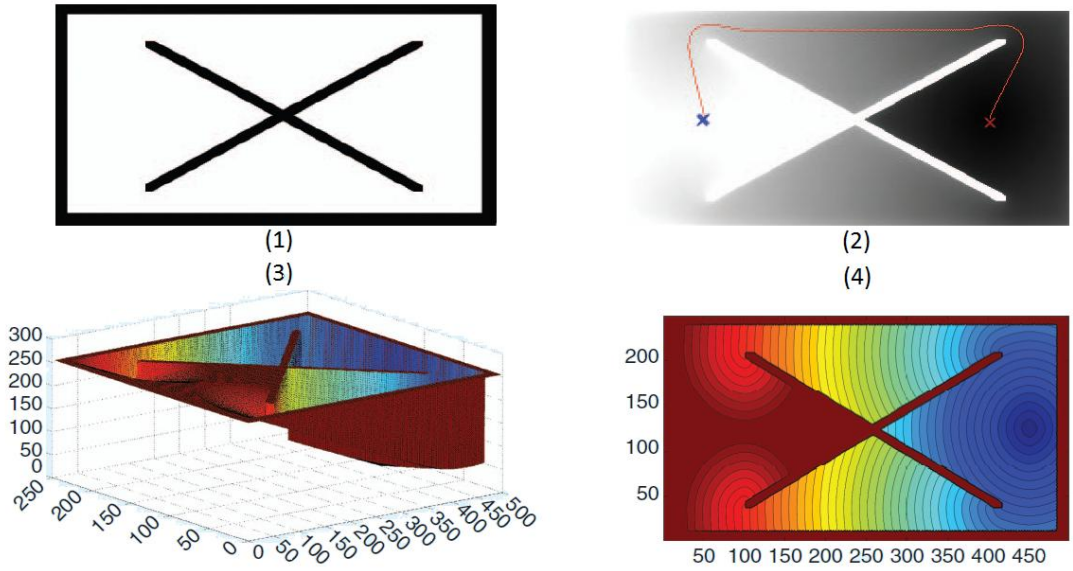


FIGURA 3."*Fases de cálculo FMM*"

Este algoritmo calcula el una trayectoria, pero se debe tener en cuenta que el camino más corto, no es a veces el más rápido, puesto que puede haber obstáculos. Para que la trayectoria sea segura se opta por ampliar los obstáculos antes de calcular la trayectoria. Por ello se utiliza el diagrama de Voronoi.

El diagrama de Voronoi es la asociación de los puntos que pertenecen a la frontera entre regiones. Estos puntos son asociados con el objeto más cercano. Se utiliza para obtener una "hoja de ruta", reduciendo el tiempo empleado en la búsqueda del camino.

El método de Voronoi sigue 4 pasos:

- Procesamiento previo: el mapa es convertido en uno binario, donde los obstáculos tendrán valor 0, representado por el color negro; y los espacios libres tendrán valor 1, con color blanco. Además se aumenta el tamaño de los obstáculos y las paredes.
- Voronoi: se obtendrá el diagrama utilizando técnicas de tratamiento de imagen morfológicas.
- FMM: es utilizado el método FMM para crear la propagación de una onda desde el punto final.
- Ruta: el método del gradiente es aplicado a la onda previamente calculada.

Este método utiliza una velocidad de expansión constante F.

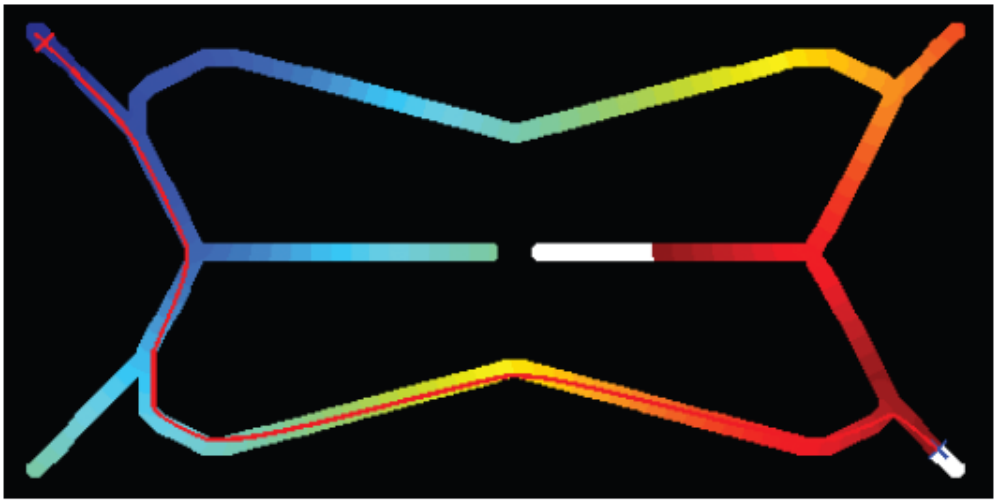


FIGURA 4. "*Voronoi*"

Una opción diferente a Voronoi es FM2. Este método utiliza, como el método anterior, un mapa de grises (0-obstaculo y 1-espacio libre). En este caso habrá múltiples fuentes de onda, porque serán originados en los obstáculos (imagen 1 de la figura 5). El valor T de la onda será mayor cuanto más se aleja de los obstáculos. El valor T será proporcional a la velocidad máxima para el robot en cada punto. Como resultado, la velocidad será menor cuanto más próximo este a un obstáculo y mayor cuanto más alejado este. En un obstáculo habrá un valor T=0, lo que crea una imposibilidad de colisión.

Para calcular la trayectoria se realiza una expansión de una onda por segunda vez, pero desde un punto, y la velocidad será proporcional al tiempo T en función de la posición (x,y), como se muestra en la imagen 2 de la figura 5. Esta velocidad será la adecuada y segura conforme a cada obstáculo presente en el mapa.

Comparando el método Voronoi con el FM2, es necesario el segundo puesto que es más fácil de implementar.

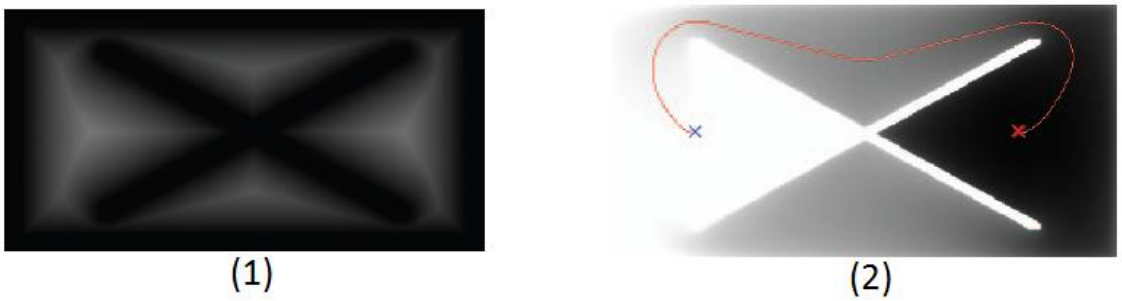


FIGURA 5. "*Método FM2*"

2.1.Fast Marching Square (FM2)

El método Fast Marchine Square (FM2) es un método utilizado para poder calcular la ruta más corta y segura entre dos puntos. El método explicado en la anterior sección se utiliza para calcular la trayectoria más corta, pero queda limitada a que puede proporcionar una ruta próxima a los obstáculos, generando un riesgo. Por ello se utiliza FM2.

Dentro del algoritmo FM2 diferenciamos dos variedades:

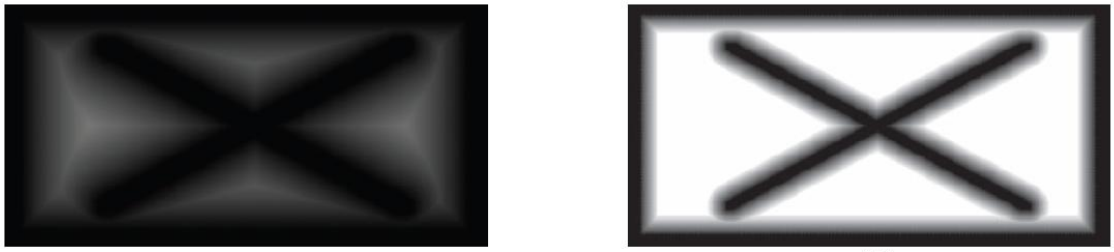
- Saturada
- Heurística

En este proyecto no se incluye la versión Heurística, por lo que nos centraremos en la versión saturada.

Para la variación saturada, el mapa binario es primero escalado y después saturado. Para escalarlo se tienen en cuenta la velocidad máxima permitida y la distancia de seguridad al obstáculo más cercano.

La saturación indica la distancia que debe mantener con los obstáculos. Para una mejor comprensión, observemos la figura 6, la cual tienes dos grados de saturación. La primera tiene un grado de saturación 1, mientras que la segunda tiene un grado de saturación próximo a 0.

En la de máxima saturación podemos observar que además del aumento de distancia con respecto a los obstáculos, también se aumenta la distancia con los límites del mapa, de manera que al ejecutar el método, obtendremos una trayectoria que no es la intuitivamente corta, como se aprecia en la imagen 2 de la figura 6. Sin embrago, si es el trayecto más corto, pero para esa saturación. Esto también condicionara la velocidad.



(1) (2)

FIGURA 6. (1) "*Saturación 1*"; (2) "*Saturación 0.2"*

Si por el contrario tenemos en cuenta la saturación mínima establecida en la imagen 2 de la figura 6, el camino si será el esperado más cortó, puesto que la distancia es la óptima. Además si nos fijamos en la figura 7 donde comparamos la distancia recorrida en ambos casos y su velocidad podemos apreciar las diferencias entre distintas saturaciones.

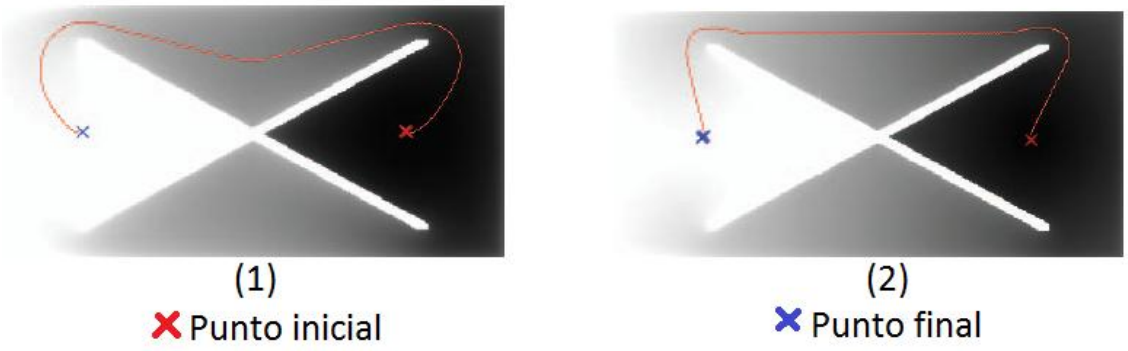


FIGURA 6. "*Comparación de trayectorias*"

Comparando los gráficos de la figura 7, podemos entender y definir que una mayor saturación implica en ciertos mapas donde no es necesario, el aumento del recorrido con respecto a un mapa saturado óptimamente. En cuanto a la velocidad, puesto que no tiene que ir bordeando, la trayectoria saturada de manera mínima mantiene una alta velocidad, además de conseguir una velocidad más alta en todo momento que la trayectoria sobresaturada. Por lo tanto, el grado de saturación correcta implica recorrer la menor distancia en la mayor velocidad posible.

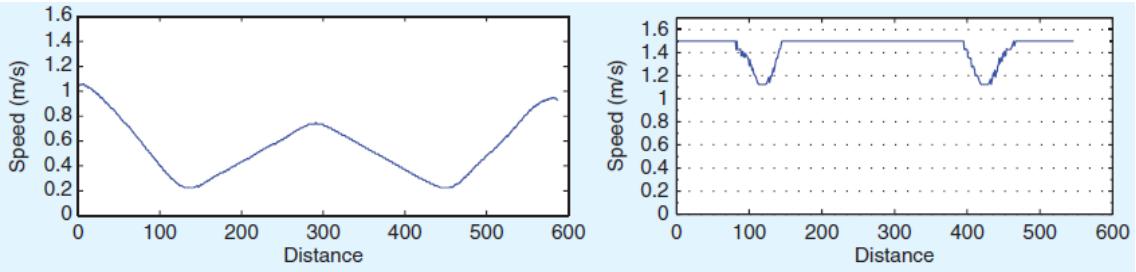


FIGURA 7. "*Comparación de velocidad y distancia recorrida*"

2.2. Fast Marching Learning (FML)

El método Fast Marchine Learning (FML) tiene como objetivo poder reproducir el robot el movimiento enseñado.

Fast Marching Learning está fundamentado en el método Fast Marching Method y Fast Marching Square, de manera que sigue los pasos de estos:

- Se crea un mapa donde todos los puntos contenidos en el son representados con 1 (negro) y los espacios sin puntos con 0 (blanco).
- Se aplica la operación de dilatación en el mapa denominado AOI (Area Of Influence).
- Se convierte el mapa en un mapa de velocidades debido a la aplicación del método de ondas del algoritmo FMM para determinar la velocidad.
- Se utiliza la saturación para poder conseguir la distancia de seguridad necesaria para estar dentro de los limites, obteniendo el mapa de velocidades final
- Aplicar el algoritmo FMM en toda el área de trabajo utilizando los puntos del centroide de todas las trayectorias y teniendo en cuenta el mapa velocidades obtenido en la figura 8.

Si nos fijamos en la figura 8, podemos observar lo descrito en los pasos anteriores, como tenemos la trayectoria de puntos en la parte izquierda de la figura, la cual es cambiada a blanco (0) los puntos de la trayectoria y a negro (1) los espacios vacios. En la imagen del medio se aplica el área de influencia para aumentarla. Y por ultimo en la imagen de la derecha obtenemos el mapa de velocidades el cual se le ha aplicado una saturación para estar en los límites. De esta manera obtenemos el mapa de velocidades.

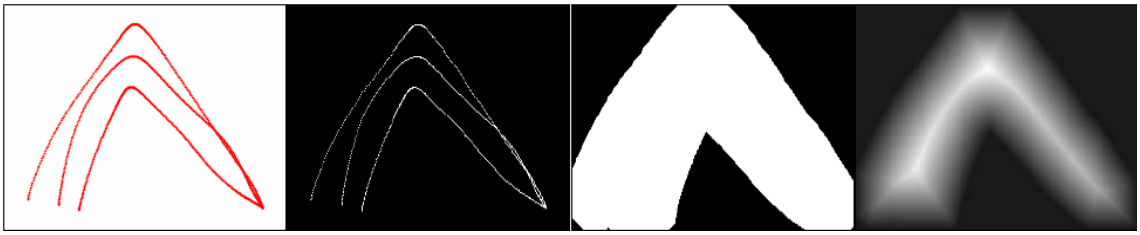


FIGURA 8. "*Calculo del mapa de velocidades F*"

En el caso de añadir un obstáculo en el mapa de velocidades, es necesario volver a calcular la saturación y actualizarse. Además, también es necesaria la actualización mediante la propagación de onda del método FMM. Además se debería volver a calcular el mapa de velocidades de nuevo porque el nuevo obstáculo puede modificar la solución final.

El método FML está compuesta por distintas cualidades como:

- Dualidad: normalmente se diseñan distintas trayectorias que tienen una dirección similar. Si por el contrario, se diseñan trayectorias en direcciones opuestas, este algoritmo la llevara a delante en el mismo movimiento, pero ejecutándola en sentido opuesto.
- Determinismo: esta cualidad se da en todos los algoritmos de Fast Marching , puesto que la salida será siempre la misma mientras no se modifica la entrada. Esto se mantiene, y en ningún momento cambia puesto que el comportamiento, debido a los datos y el algoritmo, son fáciles de predecir.
- Comportamiento sin experiencia previa: el algoritmo FML calcula el camino más rápido proporcionado por el mapa de velocidades. Aun sin experiencia sobre los puntos iniciales, como se mantiene el valor de saturación en aquellos puntos sin experiencia, el camino más rápido también será el más corto.
- Una trayectoria de aprendizaje: dependiendo del parámetro AOI y del número de trayectorias, la demostración puede ser exitosa, ejemplo de ello es la figura 9. Basta con mostrar una sola trayectoria con un AOI de valor mayor que si se hicieran muchas demostraciones. De esta manera no es necesarias varias trayectorias para demostrar el algoritmo.

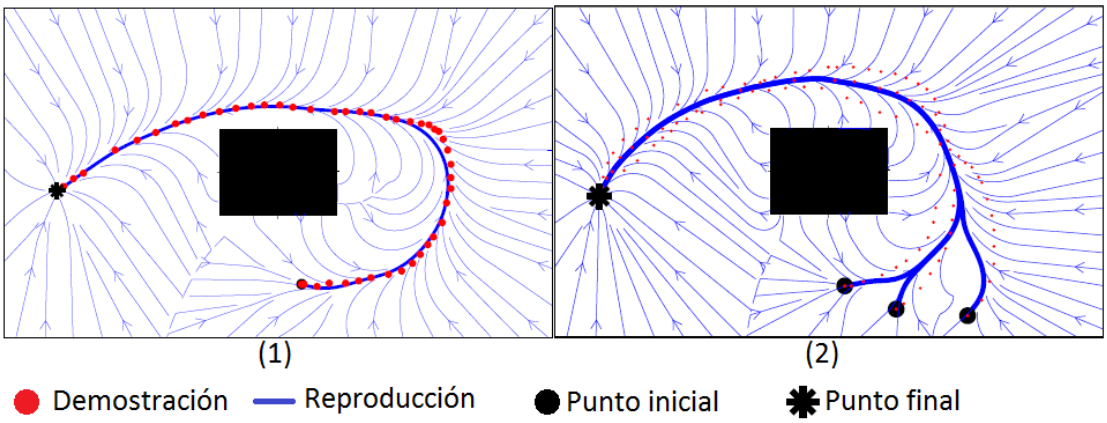


FIGURA 9. "*Demostración de trayectorias de aprendizaje*"

Este algoritmo posee el parámetro de saturación mencionado en el algoritmo FM2 y área de influencia. A continuación se analizan los dos:

- Saturación, sat: es el valor del mapa de velocidad F en lugares sin experiencia. Los lugares con experiencia tienen una velocidad de propagación mayor, puesto que habrán sido alcanzados por la onda ya comentada en el apartado de FM, como se demuestra en la figura 10. Es un parámetro entre la experiencia enseñada al robot y el resto. Este parámetro puede tomar valores mayores que 0 y menores o iguales a 1.

- Área de influencia, aoi: se encarga de ampliar los puntos introducidos para dar conectividad a las manifestaciones y el entorno. Depende del tamaño del espacio de trabajo, tomando valores de alrededor 10% para una sola demostración y alrededor del 5% para varias demostraciones (en la interfaz gráfica se ha tomado como máximo la mitad del lado más pequeño del espacio de trabajo). Si se le adjudica un valor elevado, se dilata demasiado la demostración, perdiendo el sentido de esta. En caso de un valor bajo, seguirá estrictamente el camino marcado, sin dilatación alguna. Este parámetro se mide en pixeles y en voxels (células).

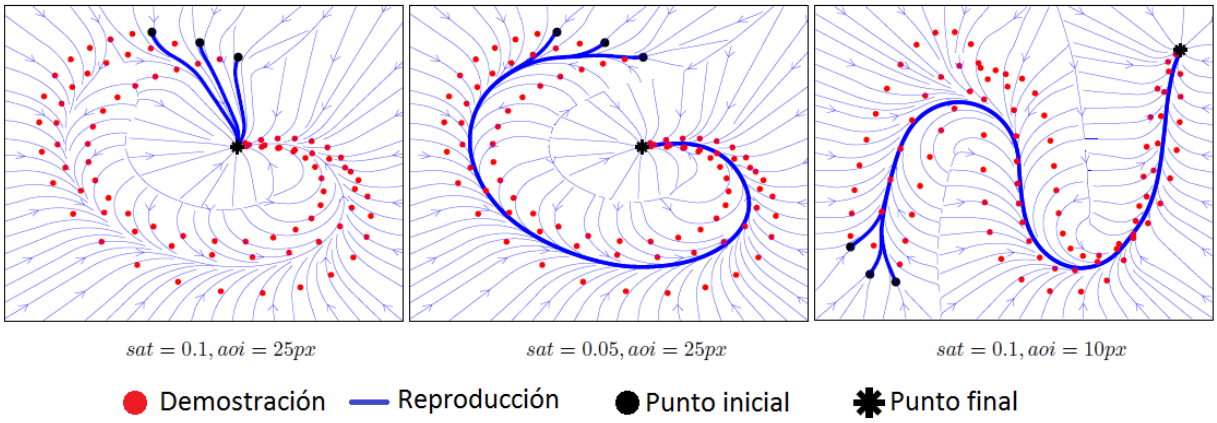


FIGURA 10.”*Diferentes valores sat y aoi*”

2.3. Fast Marching Square Formations (FM2F)

El tercer y último algoritmo utilizado en este proyecto es el Fast Marching Square Formations (FM2F) consiste principalmente en calcular la trayectoria por la que se van a mover varios robots, así como sus posiciones y orientaciones en la formación de seguimiento líder-seguidor. Este algoritmo sigue los siguientes pasos:

- Modelaje: se perfila la información sensorial de los obstáculos y los espacios, en blanco y negro.
- Ampliación de los objetos: se amplía los obstáculos y las paredes para evitar que haya colisión alguna con el robot (Figura 11, primera imagen).
- Aplicación del método FM 1º: después de identificar los obstáculos, se propaga una onda a todos los obstáculos y las pareces, donde la velocidad será 0. Como resultado, obtendremos el mapa potencial, en el cual cada pixel es proporcional a la distancia más cercana al obstáculo. Este mapa es similar al mapa de velocidades o de índice de refracción, por ello se utilizan las leyes de trasmisión de ondas electromagnéticas y de la luz para calcular la trayectoria del robot (figura 11, imagen del medio).
- Aplicación del método FM 2º: se propaga una onda electromagnética, cuyo origen es el punto final y que se propaga hasta el punto de situación del robot. Una vez alcanzado al robot, se aplica una geodésica en el mapa potencial, obteniendo el camino óptimo (Figura 11, imagen tercera).

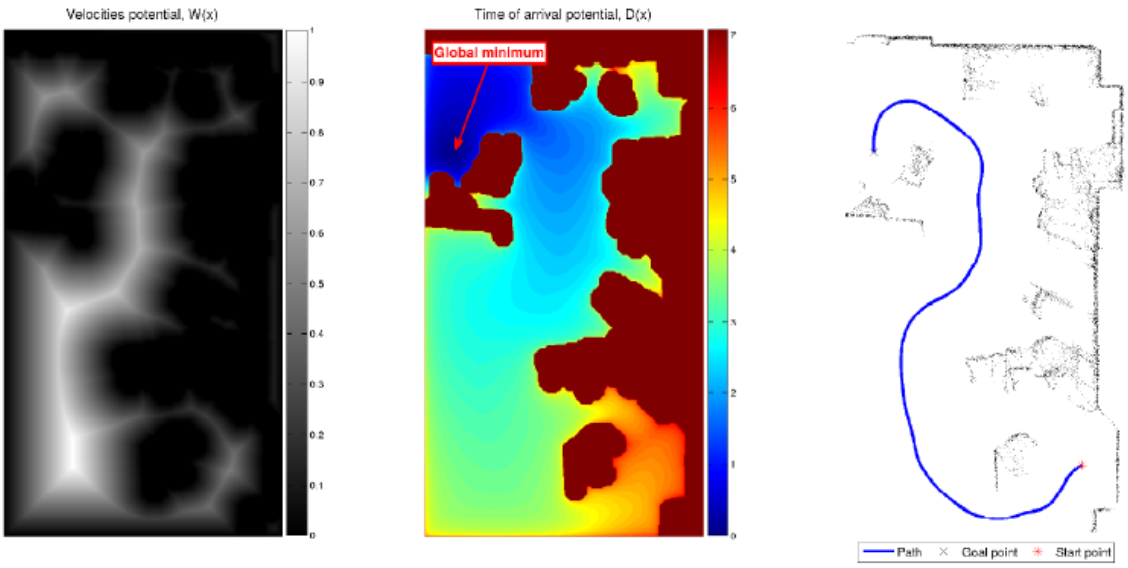


FIGURA 11. "*Ejecución paso a paso de FM2F*"

Como resultado de los pasos explicados, el resultado final tendrá una trayectoria con las siguientes características:

- Inexistencia de mínimos locales: cuando se realiza la expansión de la onda al calcular el segundo potencial se asume que solo hay un mínimo absoluto. Es imposible que haya

algún mínimo de tipo local, puesto que cuando se calcula el segundo potencial, se asume su inexistencia.

- Rapidez de respuesta: esto es debido a la baja complejidad del algoritmo acorta el tiempo de cálculo.
- Trayectorias planas y estables: producido por un mapa de velocidades sin discontinuidad. La estabilidad de las trayectorias es fruto del planificador (el método), evitando colisiones, proponiendo trayectorias seguras.
- Integridad: si existe una solución, se encontrara una trayectoria mediante la propagación de la onda.

El algoritmo FM2F, como ya se ha comentado, se basa en el algoritmo FM2 del cual se habla en el apartado 2.1 de este mismo capítulo. Concretamente, del algoritmo FM2 se utiliza un mapa para calcular dos potenciales. El mapa de velocidades denominado W(x) para todos los robots de la formación, en donde los espacios libres son 0 (representado en blanco) y los obstáculos y las paredes son 1 (representado en negro). Este mapa es utilizado para poder calcular el denominado primer potencial utilizando el método FMM del algoritmo FM2. Una vez calculado, se calcula el segundo potencial, el cual representa la distancia al punto final o de meta en la métrica. El proceso descrito es el que se le aplica al líder.

En el caso de los seguidores, se ejecutan las siguientes premisas:

- Cada ciclo de t, se incluye el mapa W (t) con la posición de los otros robots, como puntos de obstáculos (negro).
- Cada tiempo t, se calcula un primer potencial W (t).
- La geometría de la formación definida se irá deformando dependiendo de la posición de los obstáculos y de los robots, evitando chocar, puesto que para cada seguidor también se calcula un punto final.
- Los puntos meta o final serán distintos, puesto que mientras el líder llegara al punto final calculado por los potenciales, los seguidores llegaran a punto finales parciales, calculados como ya se ha mencionado en apartados anteriores.
- Al igual que con el líder, la ruta obtenida de los potenciales será la de menor distancia, con la aplicación del descenso de gradiente en el potencial segundo D.
- Todos los robots se mueven siguiendo una iteración, es decir, que hasta que todos no han hecho la iteración, no pasan a la siguiente.

Esta es la secuencia de ejecución que se genera en el líder y en los seguidores. La geometría de seguimiento definida es un triangulo equilátero, en el cual el líder esta en el vértice y los seguidores en la base, como se observa en la figura 12.

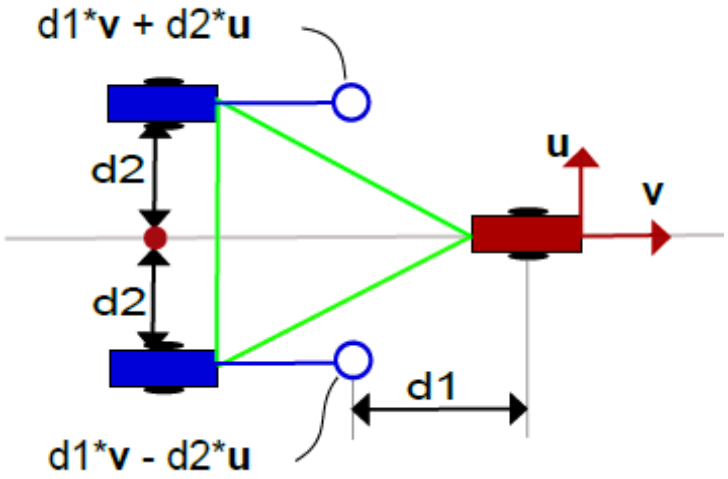


FIGURA 12.” *Formación de líder-seguidor/seguidor*”

Un punto distinto de actuación con respecto a los otros algoritmos es que los obstáculos y los robots líderes se incluyen en el mapa de velocidades, lo que aumenta la incertidumbre en la posición de ambos. El tratamiento de la incertidumbre en términos de de cálculo se repasa en la diagrama 1, donde W es designada como primer potencial. También se incluye el proceso que tiene lugar en cada avance de los robots.

Debido a la incertidumbre que se puede generar, la secuencia de de calculo que se sigue es la representada en la figura 13, donde se parte de que cada robot de la formación tiene su W (t), con el cual también se incluirá una función de incertidumbre, puesto que se utiliza la información que nos daría el sensor que tenga el robot.

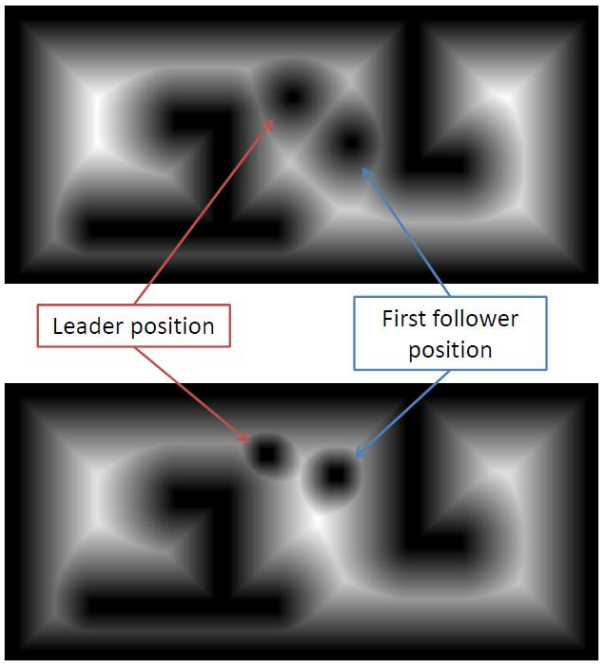


FIGURA 13. “*líder y seguidor afectados por saturación e incertidumbre*”

Para que esta función de incertidumbre sea eficaz, funciona de la siguiente forma:

- Se crea un mapa de grises, en el cual se representa el grado de incertidumbre, asumiendo valor 1 si hay incertidumbre y 0 si no existe incertidumbre alguna.
- En la posición central, se representa de negro (valor 0) el robot.
- En este mapa se aplicara el método FM, donde se calcula el mínimo entre el mapa de grises y 1, para establecer el valor máximo (1, blanco), para saber dónde está el robot (0, negro) y donde no está (1 , blanco).

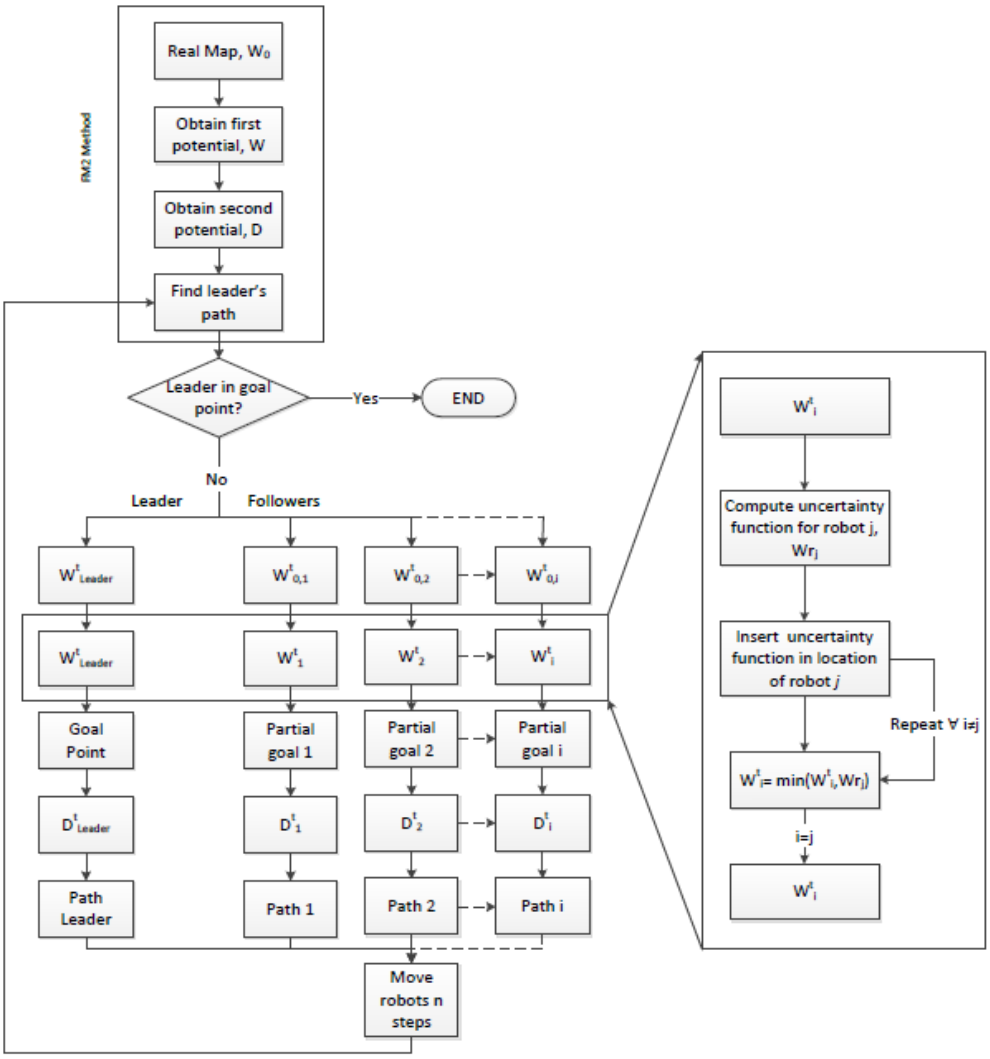


DIAGRAMA 1. "*Secuencia de cálculo de incertidumbre*"

Un aspecto destacable que aporta el método FM2 es la velocidad de saturación. En los anteriores algoritmos se utilizaba para un solo robot, de manera que pudiera recorrer una trayectoria de forma segura, con la mayor velocidad que le permita esa condición. En este caso, como estamos hablando de una formación, afectara a la geometría de formación. Este parámetro permite tener menos deformación en la geometría puesto que la velocidad es

constante, además no se deforma hasta que uno de los robots no está lo suficientemente cerca.

Un concepto que se tiene también muy en cuenta en este algoritmo, ya no solo para la ejecución de este en ordenadores, sino para la vida real son los obstáculos móviles. Los obstáculos móviles son la gente o vehículos móviles. Estos obstáculos afectaran a la formación, reagrupándola y siguiendo el camino que calcule en cada iteración el líder, puesto que se calcula teniendo en cuenta a los objetivos parciales.

En caso de detectar un obstáculo, la secuencia seguida es:

- El líder obtendrá una trayectoria libre de obstáculos a corto y medio plazo.
- El obstáculo, como ya se ha mencionado, tiene un cierto grado de incertidumbre, por lo tanto, se utiliza la función de incertidumbre mencionada para todo obstáculo móvil, teniendo en cuenta su tamaño y su velocidad.

La única diferencia entre realidad y simulación es que los obstáculos están incluidos en el primer potencial de todos los robots de la formación. En un laboratorio, se utilizaría un sensor con un rango de 360 grados.

Sección 3: MATLAB

MATLAB es una herramienta de software matemático que ofrece la posibilidad de implementar algoritmos, creación de interfaces de usuario y comunicación con otros programas en otros lenguajes. Posee para ello diferentes herramientas. En este caso, para este proyecto se utiliza la herramienta GUI de MATLAB, la cual se utiliza para poder desarrollar interfaces.

Para crear un interfaz, lo primero que hay que hacer es abrir la herramienta. Para ello, en la ventana de comandos solo hay que escribir "guide". Aparecerá una ventana donde se puede elegir entre crear una interfaz desde cero o con varios elementos ya situados en la interfaz, como se muestra en la figura 14.

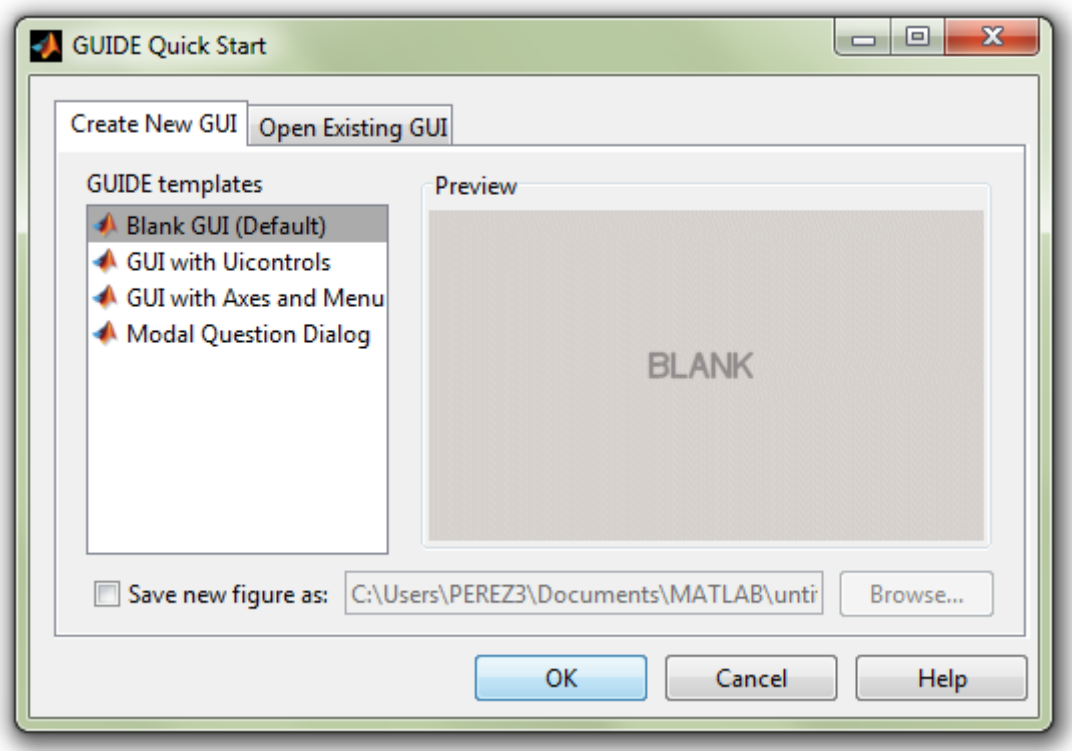


FIGURA 14." *Menú creación Interfaz*"

En seleccionando la primera opción, Black GUI, aparecerá la ventana donde se desarrolla la parte gráfica de la interfaz. En la figura 15 se encuentran los siguientes elementos:

- Barra de edición y reproducción: en esta barra, además de los menús guardar, abrir una nueva interfaz, cortar, copiar, pegar también hay opciones para editar la programación, ordenar los elementos de la interfaz o ejecutar la interfaz.
    - Align objects (1): permite alinear los elementos de la interfaz alineándolos y ordenándolos. Esto es muy útil para conseguir un orden y una mejor presentación.
    - Menú editor (2): este editor es el encargado de generar menús desplegables dentro de una interfaz. Cada pestaña desplegable genera un callback que se puede utilizar posteriormente para realizar diferentes opciones.
    - Tab order editor (3): establece el orden de ejecución de una interfaz en caso de utilizar la tecla tabulador.

- Toolbar editor (4): editor de la barra de edición, pudiendo modificar que queremos que aparezca en la barra, pudiendo sustituir por ejemplo la opción de guardar por la de nueva interfaz.
- Editor (5): esta opción permite acceder al editor de la interfaz, donde se puede programar la función que realiza cada elemento incluido en la interfaz.
- Property inspector (6): este botón muestra las características "físicas" que se muestran en la interfaz. También permite ver si existe un callback o un createFcn.
- Objects browser (7): permite buscar entre los distintos elementos de la interfaz el elemento deseado.
- Run (8): ejecuta la interfaz gráfica.

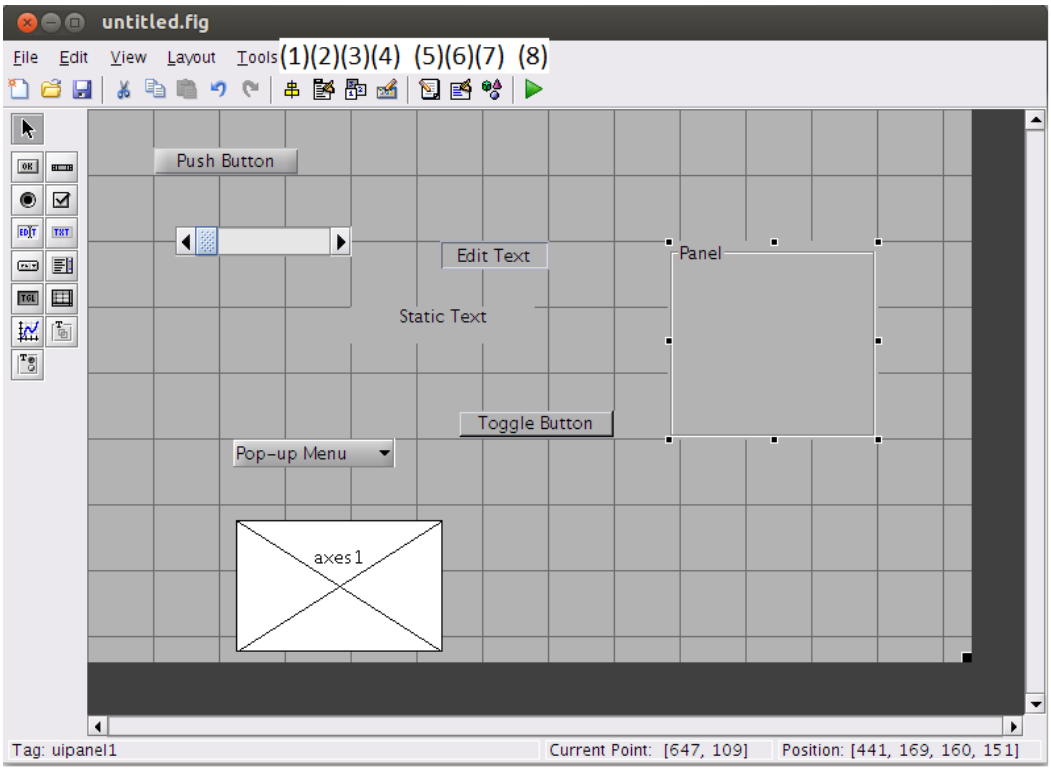
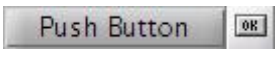


FIGURA 15. "*Creación interfaz gráfica*"

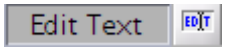
- Menú de herramientas: situada en la parte izquierda de la figura 15, en el se encuentran los elementos necesarios para crear una interfaz. A continuación se enumeran los elementos empleados durante la interfaz

- Push Button: botón que se ejecuta al presionar. Este botón permite ejecutar una sola función, al contrario que un togglebutton. Este elemento es empleado en las interfaces Draw y fm2app.



- Slider: barra utilizada asignar valores entre un rango determinado. Es utilizado para variar el valor de parámetros como saturación, AOI o incertidumbre.

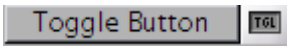
- Edit text: sirve para escribir valores y poder mostrar valores resultantes. En este proyecto son utilizados para mostrar el valor asignado a cada parámetro y para mostrar las dimensiones de los mapas.



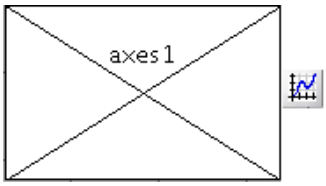
- Static Text: se utiliza para escribir textos sobre todo de ayuda. En el proyecto es utilizado para ilustrar los user hints o consejos de ayuda.

- Pop-up menú: menú desplegable utilizado para mostrar diferentes opciones. Es utilizado para mostrar las diferentes opciones de guardado.

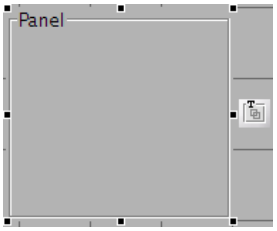
- Toggle button: este botón puede ejecutar dos funciones, puesto que tiene dos posiciones. Cada posición se corresponde con un estado, siendo 1 para activado y 0 para desactivado. Es utilizado para mostrar los ejes en la interfaz Draw.



- Axes: es un eje que se utiliza no solo para representar gráficos sino para cargar imágenes o generar animaciones. Este es el elemento principal de la interfaz gráfica fm2app2, donde se muestran todos los mapas y resultados.



- Panel: es un contenedor de todo tipo de objetos, lo que facilita el diseño de la interfaz y facilita poder modificar esta. Es utilizado en todas las interfaces del proyecto, agrupando todos los elementos sobre ello.



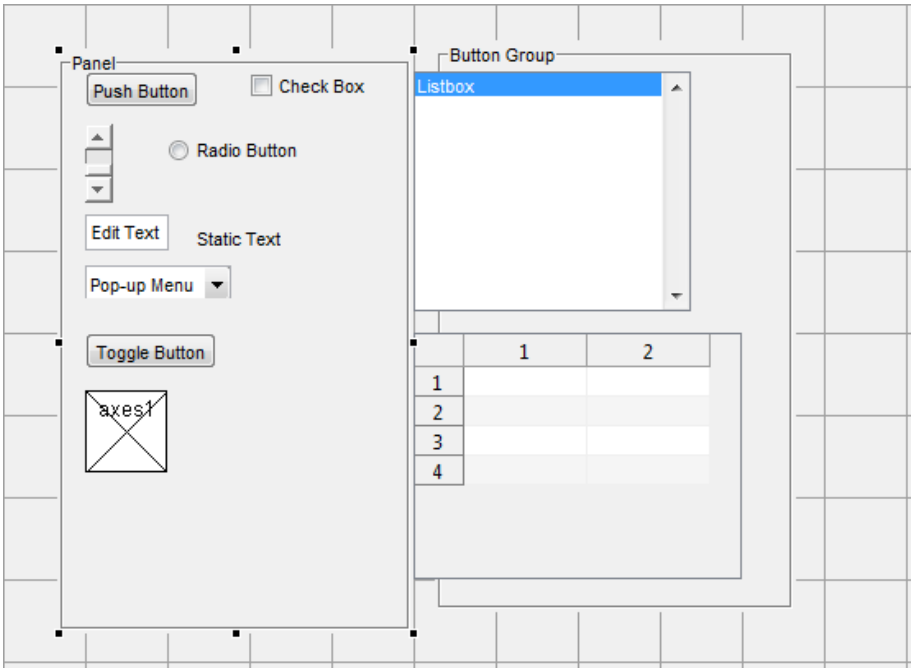


FIGURA 16. "*Elementos de una interfaz*"

Una vez colocados todos los elementos se accede mediante el menú editor a la interfaz de programación, donde se programa la función que se ejecutara en el elemento.

Por último, al colocar cada elemento, se pueden definir los parámetros de dimensión, posición, el Tag que identifica a cada elemento y editar el contenido en caso de que sea un popupmenu, edit_text o static_text. En la figura 17 se muestra un ejemplo de un inspector correspondiente a un Panel.

Una vez programadas las funciones y distribuidos los elementos se ejecutara la interfaz utilizando el botón run.

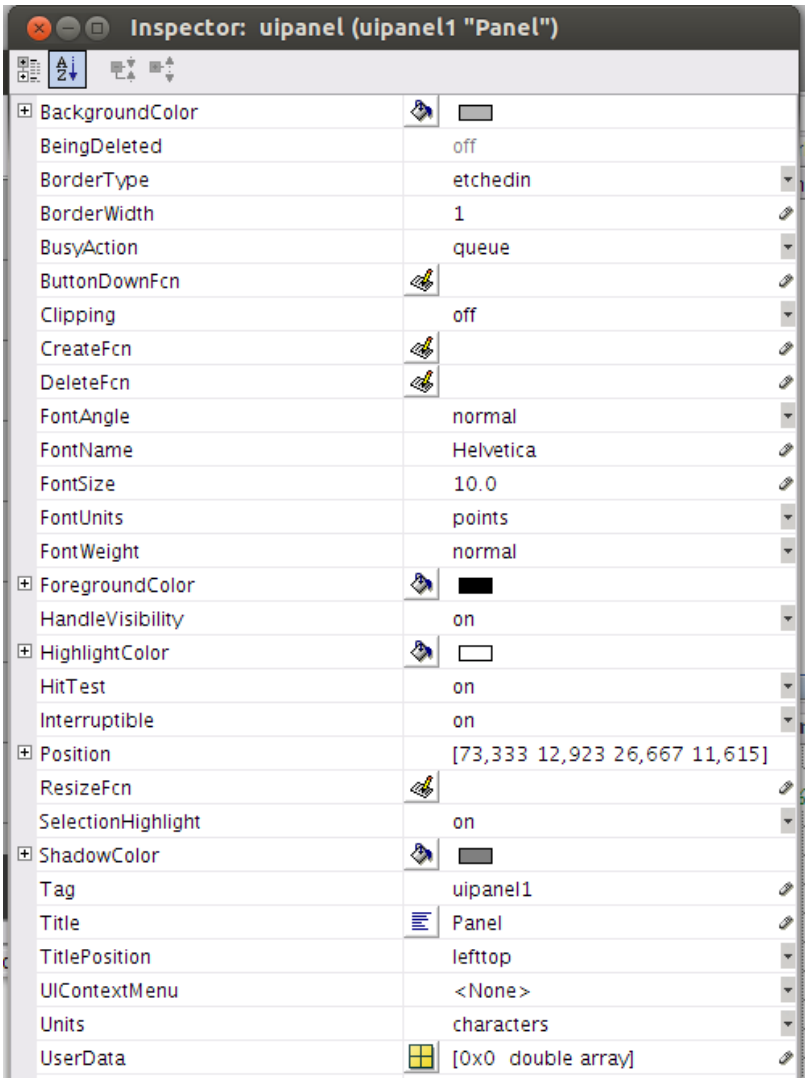


FIGURA 17. "*Inspector*"

Sección 4: Desarrollo del proyecto

En esta sección se abordará la integración de los algoritmos explicados en el sección 2 en la herramienta GUIDE. El proyecto está dividido en tres interfaces gráficas:

- Draw.m: en ella podremos crear nuevos mapas para después utilizarlos para las demostraciones de los tres algoritmos. Además de poder crear nuevos mapas, se pueden modificar mapas ya existentes.
- FM2app.m (Panel): en esta interfaz se encuentra el menú principal, donde podemos cargar mapas, acceder a la interfaz Draw.m, o ir modificando distintos parámetros, además de poder ejecutar las demostraciones.
- FM2app2.m (Pantalla de resultados): en esta interfaz encontraremos todos los mapas y los gráficos con los que se trabaja y se obtiene los resultados.

Como se ha explicado, existen tres interfaces, pero la estructura en la que se va a ir explicando en las siguientes hojas es la siguiente:

- Draw.m
- FM2app.m
    - FM2
    - FML
    - FM2F
- FM2app2.m
    - FM2
    - FML
    - FM2F

De esta forma, como cada diseño dentro de la interfaz FM2app y FM2app2 son distintos, se podrá obtener una mejor comprensión del diseño y su implementación.

4.1 Draw.m

Esta interfaz es el lugar de trabajo donde se puede desarrollar nuestros mapas binarios que luego podrán utilizarse en la interfaz FM2app y FM2app2 para poner en práctica las distintas demostraciones de cada algoritmo integrado. A continuación vamos a explicar el funcionamiento y posteriormente se explicara la implementación necesaria.

FUNCIONAMIENTO.

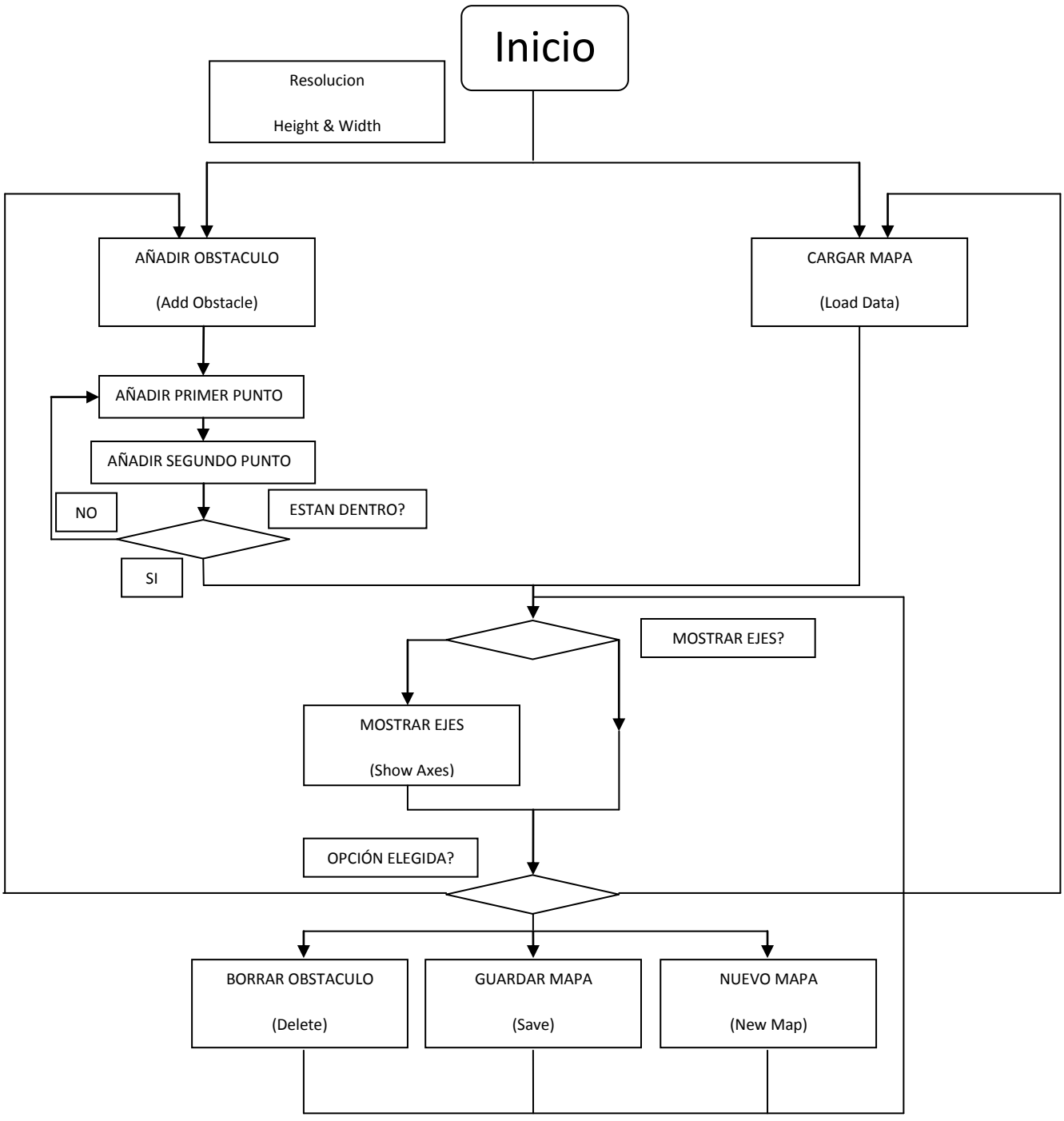


DIAGRAMA 2. "*Diagrama Funcionamiento Draw*"

Como se explica en el diagrama 2, al acceder a la interfaz Draw (Figura 18), las opciones disponibles en la interfaz serán las siguientes:

- Resolution (1): dos cuadros de texto donde hay que introducir las medidas anchura y altura del mapa que se quiera diseñar.
- Load Data (4): esta opción permite cargar un mapa ya trabajado al cual se le quieran hacer modificaciones.

En caso de querer trabajar con un nuevo mapa, se deberá introducir en los cuadros de altura y anchura las dimensiones para poder generar uno nuevo. Una vez introducidos los valores correctos, se debe pulsar el botón Añadir obstáculo (2) (Add obstacle).Esto permite dos cosas: poder dibujar el mapa puesto que ya se tienen las medidas e incluir un primer obstáculo.

Esto está diseñado así debido a que al introducir las medidas no se ejecuta ningún tipo de algoritmo que permita tener un mapa dibujado. Además, se obliga a dibujar por primera vez, debido a que en la mayoría de los casos no interesa un mapa sin obstáculos.

En caso de cargar un mapa, se puede hacer en tres formatos: *bmp,* png o *mat. Una vez pulsado este botón, se abre un cuadro de dialogo donde se permite cargar el mapa y cargarlo en el eje. Elegir esta opción desactiva la posibilidad de introducir una nueva anchura y altura.

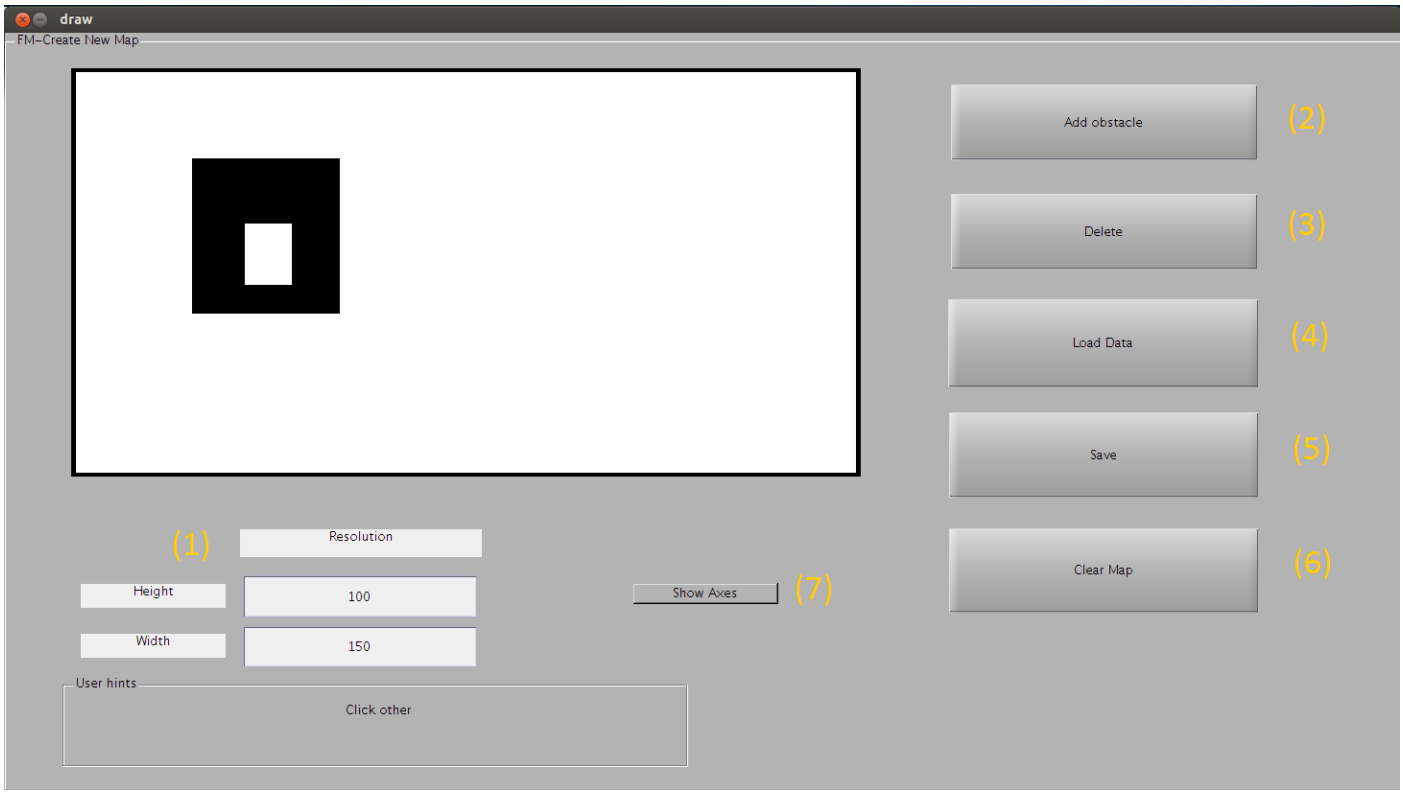


FIGURA 18.”*Resultado final espacio diseño mapas*”

Una vez cargado/creado un mapa, se habilitan y se pueden ejecutar las siguientes opciones:

- Add Obstacle (2): permite añadir un obstáculo de geometría rectangular.
- Delete (3): elimina una porción seleccionada.
- Load Data (4): carga un mapa.
- Save (5): guarda el mapa disponible en el eje.
- New Map (6): borra todos los obstáculos del mapa.
- Show Axes (7): Muestra los ejes de coordenadas en el eje.

Add Obstacle

Este botón permite añadir al mapa un obstáculo de geometría rectangular. Para ello, habrá que definir dos puntos opuestos que permitan obtener los cuatro puntos. Como se especifica en la figura 19. De esta manera al dibujar en la interfaz, los puntos definidos son P1 y P2. Con las coordenadas de P1 podemos definir la coordenada *x* de P3 y la coordenada y de P4. De la misma manera con la coordenada P2 definimos la coordenada *y* de P3 y la coordenada x de P4, definiendo los cuatro vértices del obstáculo.

Posteriormente a la definición de los puntos, se rellena el obstáculo, generando un obstáculo opaco. La definición de los obstáculos de puede hacer en los 4 sentidos que se muestran las flechas, siendo el final de la flecha la coordenada P2 y el centro donde convergen el punto P1.

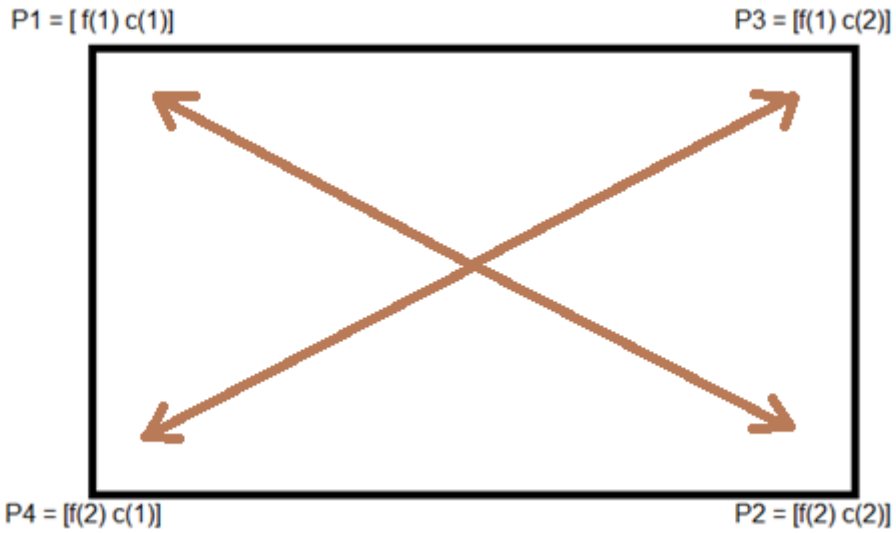


FIGURA 19. "*Definición del obstáculo*"

Delete

Al ejecutar este botón se tiene que definir dos puntos P1 y P2 iguales a los definidos en Add obstacle, puesto que sigue el mismo método que este, solo que en vez de dibujar, borra el área comprendida en la región rectangular. En la figura 20 se muestra el resultado de ejecutar este botón.

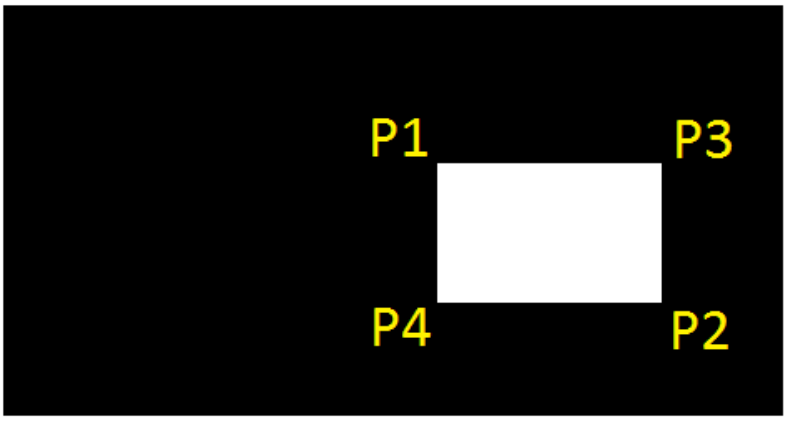


FIGURA 20. "*Definición del área de eliminación*"

Load Data

Como se explicaba arriba, se trata de un botón que permite cargar un mapa guardado.

Save

Una vez terminado el mapa de trabajo, este botón permite guardar el mapa creado en formato *mat.

New Map

Al ejecutar esta opción, el mapa que se haya creado o cargado, es borrado completamente, dejando el mapa tan solo con los márgenes generados por defecto.

Show Axes

Esta opción permite mostrar los ejes de coordenadas, como se muestra en la figura 21. Para que esta opción pueda mostrarse, debe de ser pulsada antes de que se pulse Add Obstacle, Delete o New Map.

Al ser un *ToggleButton* (botón de doble estado), al pulsarlo no se genera un evento y se ejecuta la función que desempeñe, sino que al pulsarlo tomara un valor que hará que se muestren los ejes y no se desactivara esta opción hasta que vuelva a ser pulsado.

Todos los elementos descritos hasta ahora, incluyendo el eje donde se cargan los mapas, están agrupados en un *Uipanel.*

Un *uipanel* es un panel en el cual se agrupan los elementos de la interfaz. Ha sido elegido porque en caso de algún rediseño del tamaño de la interfaz, también redimensiona el tamaño de los elementos que haya en su interior. En este caso si se utiliza la propiedad *ResizeFcn* para redimensionar la ventana de la interfaz y sus componentes.

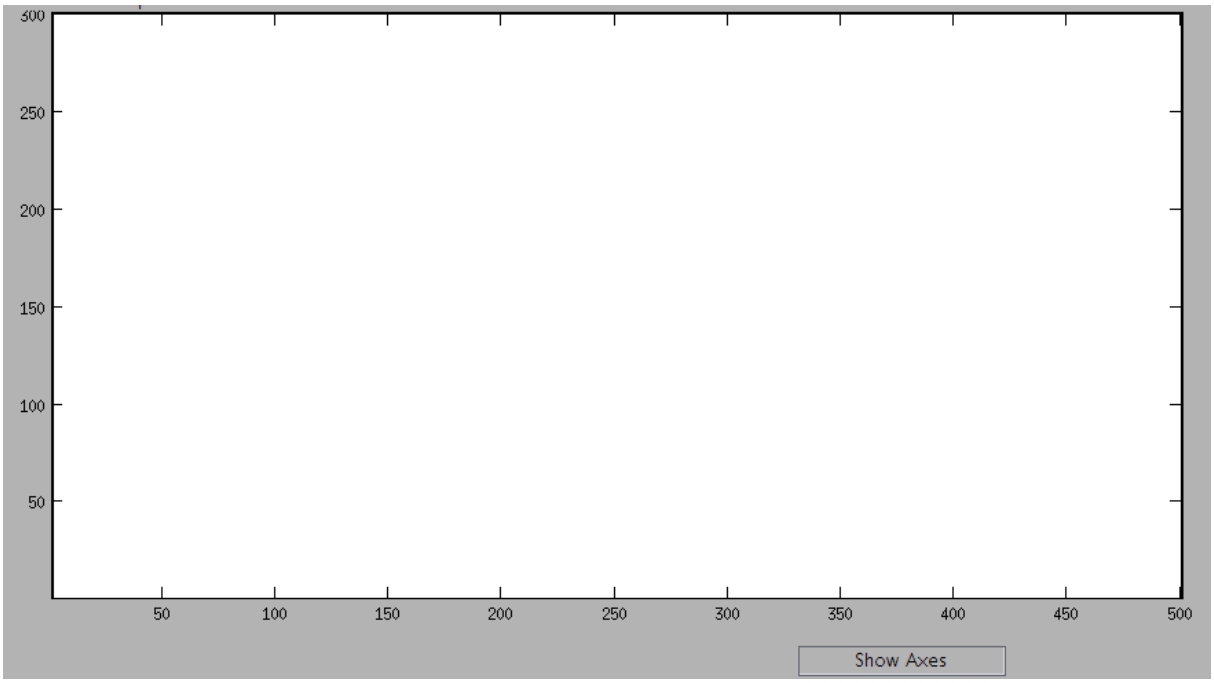


FIGURA 21. "*Show Axes=1*"

4.2 FM2app.m

En este apartado vamos a analizar las interfaces correspondientes al panel de cada algoritmo, a través del cual podremos modificar las diferentes variables que correspondan a cada algoritmo.

Para poder seleccionar entre las distintas *uipanel* de los algoritmos, utilizamos por un lado el menú editor, para crear una barra de menú en la interfaz y crear pestañas para seleccionar las distintas ventanas o *uipanel.* En la figura 22 podemos ver como es el menú editor.

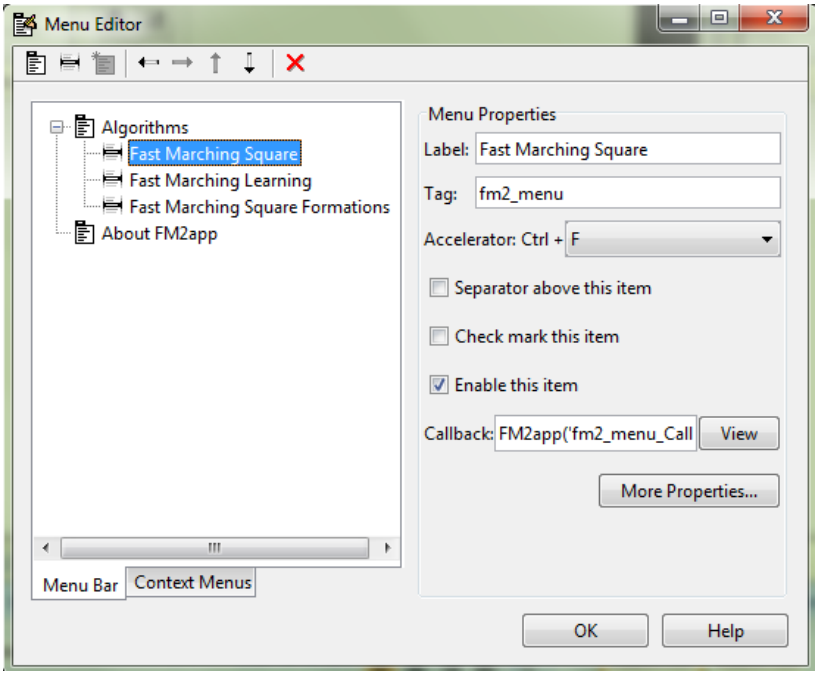


FIGURA 22. "*Menú editor*"

4.2.1 FM2 (Fast Marching Square)

Este algoritmo tiene como finalidad conseguir encontrar la trayectoria más corta entre dos puntos en relación a su velocidad y la distancia recorrida. Por tanto, en esta interfaz se podrán configurar los parámetros de saturación para poder obtener los diferentes resultados que posteriormente visualizaremos en la interfaz FM2app2, en la ventana correspondiente.

FUNCIONAMIENTO.

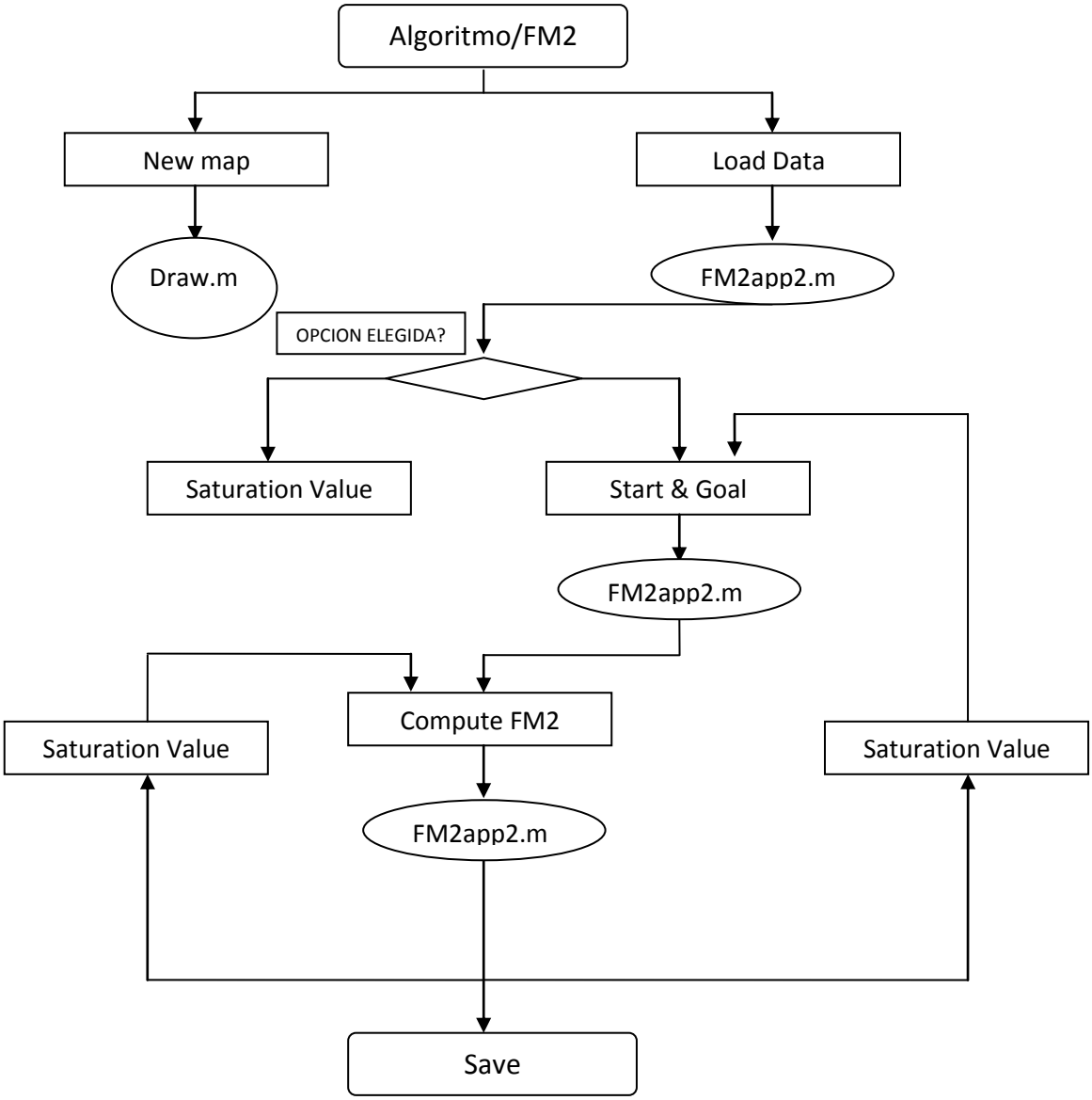


DIAGRAMA 3. "*funcionamiento FM2*"

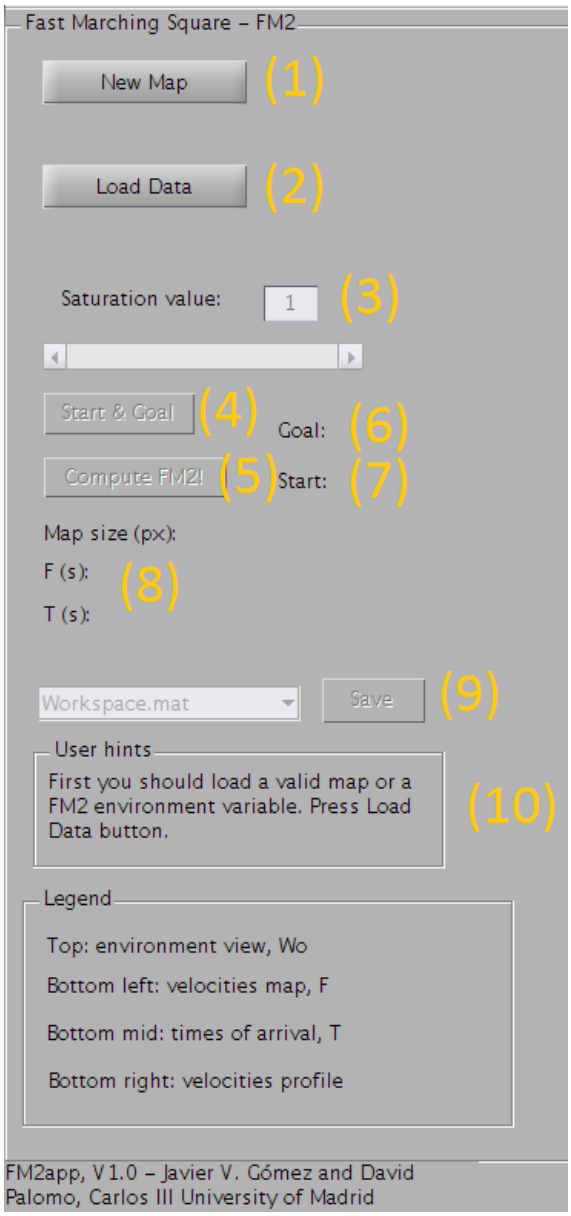


FIGURA 23. "*Panel FM2*"

Como se explica en el diagrama 3, al ejecutar fm2app.m, se muestra un panel. En la barra de herramientas, accediendo al menú Algorithms podemos seleccionar el algoritmo. Para este caso FM2. Una vez seleccionado, se mostrara el panel del algoritmo FM2 como el de la figura 23.

Para comenzar a ejecutar el algoritmo, primero estarán disponibles dos opciones:

- New Map (1): nos permitirá acceder a la interfaz Draw.
- Load Data (2): esta opción permite cargar un mapa listo para usar.

En caso de pulsar New Map, aparecerá la interfaz Draw, donde podremos diseñar un mapa para posteriormente utilizarlo en este algoritmo.

La otra opción es cargar el mapa mediante el botón Load Data. Al pulsarlo, accederemos a la interfaz de gráficos fm2app2.m y seguidamente nos aparecerá una ventana donde podremos seleccionar un mapa. Los mapas que se cargan pueden ser en formato *bmp, *png, *mat (New map create) y *mat (Wokspace FM2). Al cargar el mapa se mostraran las dimensiones en el *text_edit* (9, superior).

Una vez seleccionado el mapa, se cargara dos mapas: el mapa de obstáculos, el cual será donde se representara la trayectoria; y el mapa de saturación, el cual mostrara el mapa saturado en mayor o menor medida en función del valor. En la sección de la interfaz fm2app2 correspondiente a FM2 se explica con mayor detalle.

Tras elegir entre cualquiera de las dos opciones comentadas, se desbloquean las siguientes opciones:

- Saturation (3): propiedad del algoritmo que permite establecer una distancia de seguridad con los obstáculos.
- Start & Goal (4): esta opción permite introducir los puntos inicial y final para el cálculo de la trayectoria.

Saturation

Antes de comenzar a ejecutar el algoritmo es necesario ajustar la saturación. La saturación es una propiedad que permite ajustar la distancia de seguridad entre los obstáculos y el robot. Este parámetro condiciona la velocidad y la distancia recorrida dentro de la trayectoria más corta. Los valores que se pueden ajustar serán entre 0 y 1, donde 0 corresponde a una saturación inexistente y 1 al máximo de saturación. El valor de saturación es mostrado por una *box*. La variación de este parámetro es reflejada en el mapa de saturación, el cual se actualiza cada vez que es modificado el valor.

Start & Goal

Una vez ajustado el valor de saturación, y antes de poder ejecutar el algoritmo, se deben introducir un punto inicial de la trayectoria y un punto final de la trayectoria. Pulsando el botón Start & Goal permite precisamente esto. Una vez pulsado, se activa la interfaz fm2app2 y en el mapa de obstáculos introducimos por ratón el punto inicial y posteriormente el final. Las coordenadas correspondientes al punto inicial y final se reflejan en los *text_edit* (6) y (7) respectivamente.

En caso de que sea un punto fuera del mapa o este situado encima de un obstáculo, no será válido y habrá que volver a introducir el punto. Esto es notificado con un mensaje de error, como el de la figura 24.

FIGURA 24. "*Mensajes puntos no validos*"

El siguiente paso para poder ejecutar la demostración es pulsar el botón Compute FM2.

Compute FM2 (5)

Al pulsar ser pulsado, se ejecuta el algoritmo. Este botón generara en la interfaz de fm2app2 un grafico de resultados, donde se representara la velocidad en función de la distancia del recorrido de la trayectoria; un mapa de velocidades, representando la onda que genera el algoritmo para calcular la velocidad desde el punto inicial; y un nuevo mapa de obstáculos actualizado representando la trayectoria final.

También se generaran dos resultados (8): la velocidad en segundos (situado en el medio) y el tiempo de llegada en segundos también (situado el ultimo). El primer resultado se genera al ajustar el valor de saturación. El segundo corresponde al resultado del tiempo de llegada final obtenido.

Save (9)

Una vez finalizada la demostración, se puede guardar los resultados mediante el botón save y el *popupmenu*. El *popupmenu* permite guardar las siguientes opciones, como se muestra en la figura 25, donde también se representan los gráficos referenciados a continuación en cada nombre:

- Workspace.mat: esta opción permite guardar el espacio de trabajo de la demostración, guardando los tres mapas y el grafico, además de toda la información que se muestra en la figura 24 (punto inicial, punto final, la trayectoria final y el valor de saturación). Este workspace será guardado en formato *mat.
- Environment (1): esta opción permite guardar el mapa de obstáculos con la trayectoria dibujada. Al ejecutar el botón save, se podrá salvar en formato *pdf y *eps.
- Velocities map (2): permitirá guardar el mapa con saturación. El formato de guardar será en *pdf y *eps.
- Times-of-arrival (3): esta opción selecciona el mapa de expansión de la onda para el cálculo de la velocidad. Se guardara en formato *pdf y *eps.
- Vels.profile (4): hace referencia a la gráfica de resultados de la demostración. Sera guardado en formato *pdf y *eps.

Una vez seleccionado en el *popupmenu* entre las distintas opciones, se presiona el botón save y aparecerá una ventana emergente donde poder elegir el sitio donde guardarlo.

Otra alternativa a guardar puede ser volver a ajustar el valor de saturación y ejecutar el algoritmo presionando Compute FM2 con los mismos puntos inicial y final o volver a introducir los puntos presionando Start & Goal, como se indica en el diagrama 3.

FIGURA 25. "*Resultado final FM2*"

Para diseñar esta interfaz, se ha seguido una estructura en sintonía con los pasos a seguir de arriba hacia abajo. Se sitúa en la zona superior la parte perteneciente al añadir el mapa o los workspaces guardados. En la zona media los parámetros, y más abajo el botón/es para introducir puntos y ejecutar el algoritmo. Siguiendo este diseño, se facilita el seguir los pasos en orden y de la manera correcta.

Además, para ayudar al usuario en todo momento a que nos se pierda, se han añadido user hints (consejos en cuadros de texto) que varían en función del paso en el que estemos. De esta manera, los user hints siguen el diagrama representado al principio de esta sección.

4.2.2 FML (Fast Marching Learning)

Esta ventana de la interfaz FM2app tiene como objetivo obtener trayectorias de aprendizaje, que nos permita entender cómo afecta la saturación o el área de influencia en las trayectorias.

FUNCIONAMIENTO.

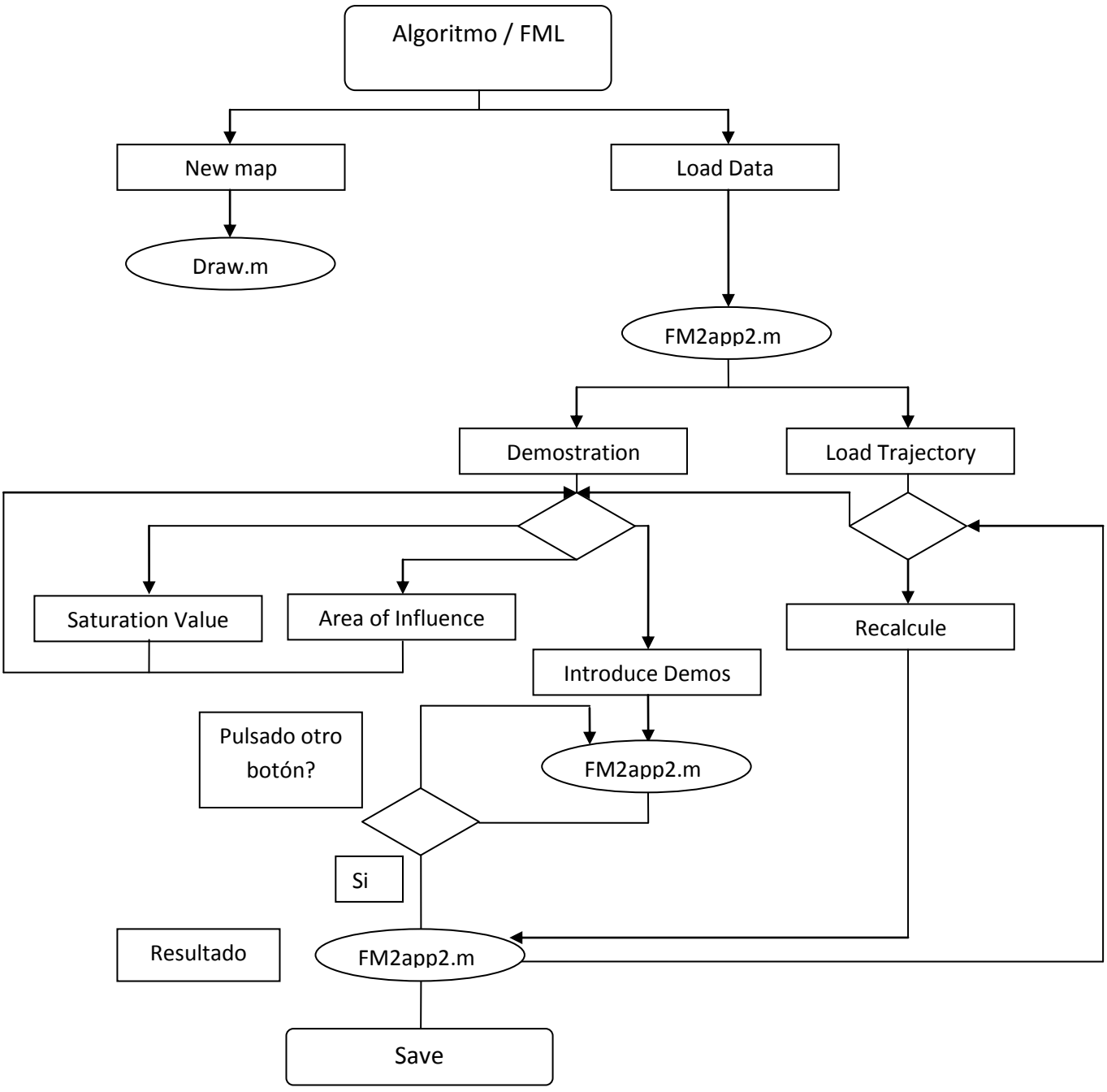


DIAGRAMA 4. "*funcionamiento interfaz FML*"

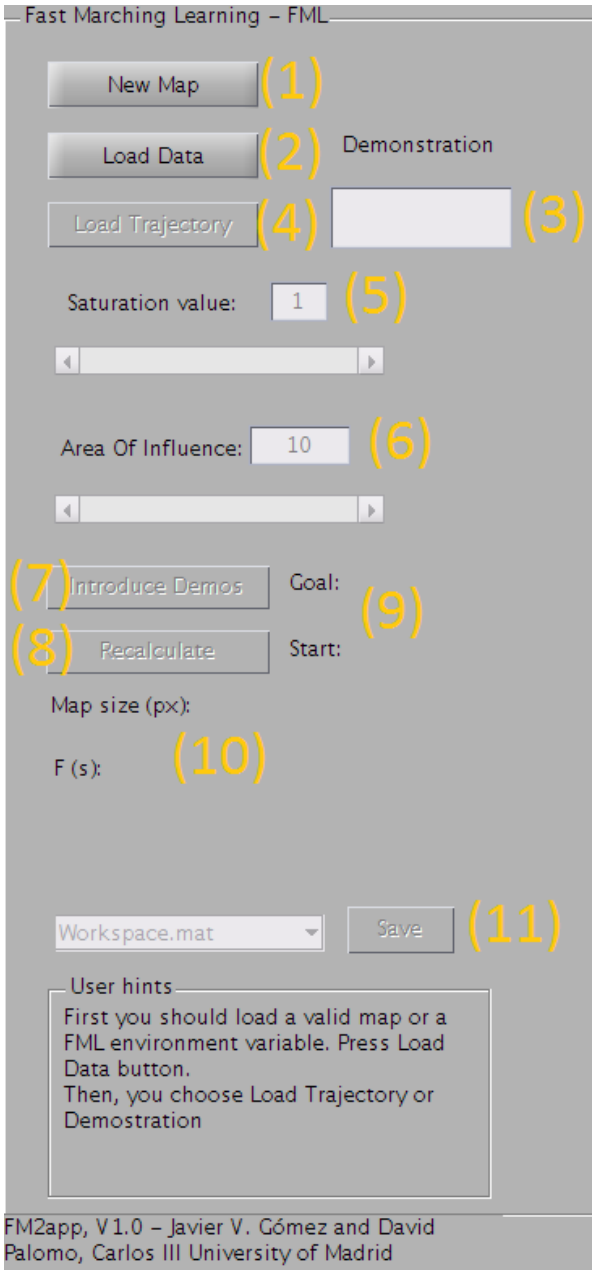


FIGURA 26. "*Panel FML*"

Para comenzar, como se explica en el diagrama 4, al ejecutar fm2app.m aparecerá un panel como el de la figura 26, pero sin ningún algoritmo. Para seleccionarlo se presiona en la barra de herramientas la opción Algorithms y dentro se selecciona el algoritmo. En este caso, FML. Una vez seleccionado, aparecerá un panel como el de la figura 26.

Inicialmente solo habrá dos campos disponibles, como ya sucedía en FM2:

- New map (1): permite acceder a la interfaz Draw, donde se podrá diseñar un mapa para posteriormente utilizarlo en la demostración.
- Load Data (2): esta opción permite cargar el mapa ya disponible.

En caso de cargar un mapa, se presionara el botón Load Data. Al presionarlo se activara la interfaz fm2app2. Seguidamente aparecerá una ventana emergente para seleccionar un mapa en formato *bmp, *png, *mat (Wokspace FML) y *mat (New map create). A continuación se cargara en el eje de demostración y el mapa de saturación, mostrando el mapa saturado en mayor o menor grado en función del valor. En la sección de la interfaz fm2app2 correspondiente a FML se explica con mayor detalle.

Una vez cargado el mapa, surgen dos posibles alternativas para ejecutar la demostración, como se detalla en el diagrama 4:

- Crear la demostración y ejecutar el algoritmo.
- Cargar una demostración guardada y ejecutar el algoritmo.

A continuación se analizaran las dos alternativas.

CREAR UNA NUEVA DEMOSTRACIÓN

Para ejecutar la demostración se necesita el número de trayectorias que se va a introducir y los puntos que forman la trayectoria. Por tanto, en el edit_text de demostraciones (3), se introducirá el número de demostraciones o trayectorias que se quiera introducir. Una vez introducido, se habilitaran los siguientes campos:

- Saturation (5): propiedad que permite ajustar la distancia de seguridad respecto de los obstáculos.
- Area of Influence (6): se encarga de ampliar los puntos introducidos para dar conectividad a las reproducciones y el entorno (obstáculos).
- Introduce Demos (7): permite introducir los puntos que componen la demostración y posteriormente ejecutar el algoritmo FML.

Saturation

Como se describía en el algoritmo FM2, esta propiedad permite establecer la distancia de seguridad entre los obstáculos y el robot. Para ello, mediante la *slider* se puede ajustar que es mostrado en una *box*. Este parámetro puede tomar valores entre 0,01 y 1, donde 0.01 es el mínimo valor y 1 el máximo. Debe tenerse en cuanta que este valor condiciona la velocidad y la distancia de la trayectoria. Este valor, como pasaba en FM2, condicionara el resultado en segundos mostrado en el *text* (10). La variación de este parámetro es reflejada en el mapa de saturación, el cual se actualiza cada vez que es modificado el valor.

Area of Influence (AOI)

Esta propiedad permite ajustar la amplitud de los puntos introducidos para dar mayor conectividad entre la reproducción y entorno. Este valor se establece en función de las dimensiones del mapa utilizado, de manera que el máximo valor permitido corresponde a la mitad del lado más pequeño. Al igual que la saturación, este valor se ajusta utilizando una *slider* y visualizando el valor en una *box*.

Introduce Demos

Una vez ajustados los valores de saturación y de AOI, el siguiente paso es introducir los puntos que formaran cada demostración. Para ello se utiliza el ratón y el botón derecho, presionándolo encima del lugar deseado dentro del mapa de demostración situado en la interfaz gráfica FM2app2. Se podrán introducir tantos puntos como sean necesarios. El primero y el último punto serán mostrados en el *text* (9). Estos puntos deberán estar dentro del mapa y fuera de cualquier obstáculo situado en el mapa.

En caso de estar en alguna de las dos situaciones descritas anteriormente, se mostrara una ventana emergente como la presente en la figura 27.

FIGURA 27. "*Mensajes de error FML*"

Cuando se quiere finalizar la introducción de puntos de esa demostración (en caso de tener solo una, se finaliza el proceso de introducir demostraciones) se pulsa un botón distinto al utilizado para introducir demostraciones. Esto dará lugar a la ejecución del algoritmo FML, lo que generara un mapa de resultados como el de la figura 28.

Finalizada la demostración, se habilitaran los últimos campos:

- Recalcule (8): permite volver a ejecutar el algoritmo sin necesidad de introducir nuevas demostraciones.
- Save (11): permite guardar los resultados obtenidos.

Recalcule

Esta opción está pensada para poder realizar comprobaciones sobre el efecto que provoca la saturación y AOI sobre las demostraciones introducidas. Al pulsarlo se ejecutara directamente el algoritmo, recalculando el resultado con los nuevos ajustes.

Save

El botón save es el encargado de guardar la imagen o workspace seleccionado. Para seleccionar aquello que se quiere guardar, se utiliza el popupmenu disponible a la izquierda del botón. En él, se puede elegir entre las siguientes opciones, que están referenciadas a la figura 28:

- Workspace.mat: esta opción permite guardar el espacio de trabajo de la demostración, guardando los tres mapas, además de toda la información importante como el punto inicial, punto final, la trayectoria final y los valores de saturación y AOI. Este workspace será guardado en formato *mat.

- Trajectory.mat: esta opción permite guardar las demostraciones introducidas, para poder reproducir la demostración en otro mapa con los mismos puntos. También se guardara en formato *mat.
- Map of Result (1): esta opción permite guardar el mapa de obstáculos con el resultado de la demostración. Se podrá salvar en formato *pdf y *eps.
- Velocities map (2): permitirá guardar el mapa saturado. El formato de guardar será en *pdf y *eps.
- Map of Demos (3): esta opción selecciona el mapa que tiene almacenado cada punto de la cada demostración. Se guardara en formato *pdf y *eps.

Una vez seleccionado en el *popupmenu* entre las distintas opciones, se presiona el botón save y aparecerá una ventana emergente donde poder elegir el sitio donde guardarlo.

CARGAR UNA DEMOSTRACIÓN GUARDADA

En este caso, una vez cargado el mapa de demostración en su correspondiente eje, se habilita el campo Load Trajectory. Este botón permite acceder a un archivo *mat que tendrá almacenados puntos de demostraciones anteriores. Una vez cargado se podrán ajustar los valores de saturación y AOI o introducir más demostraciones. La diferencia respecto a crear una nueva demostración es que también se podrá directamente ejecutar el botón recalcule, puesto que ya se tienen unas demostraciones introducidas.

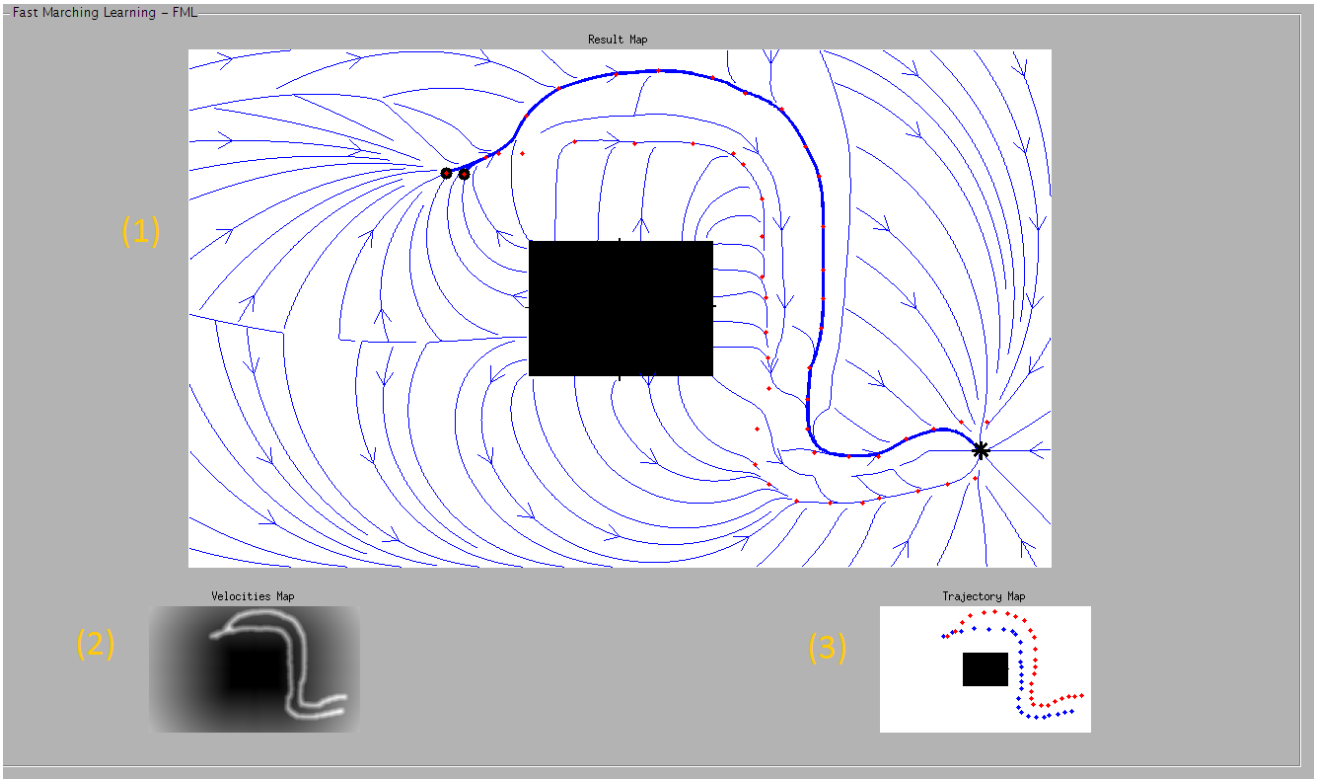


FIGURA 28. "*interfaz gráfica FML*"

Para diseñar esta interfaz, como en el algoritmo FM2, se ha seguido una estructura en que sigue el diagrama situado al principio de esta sección. Se sitúa en la zona superior la parte

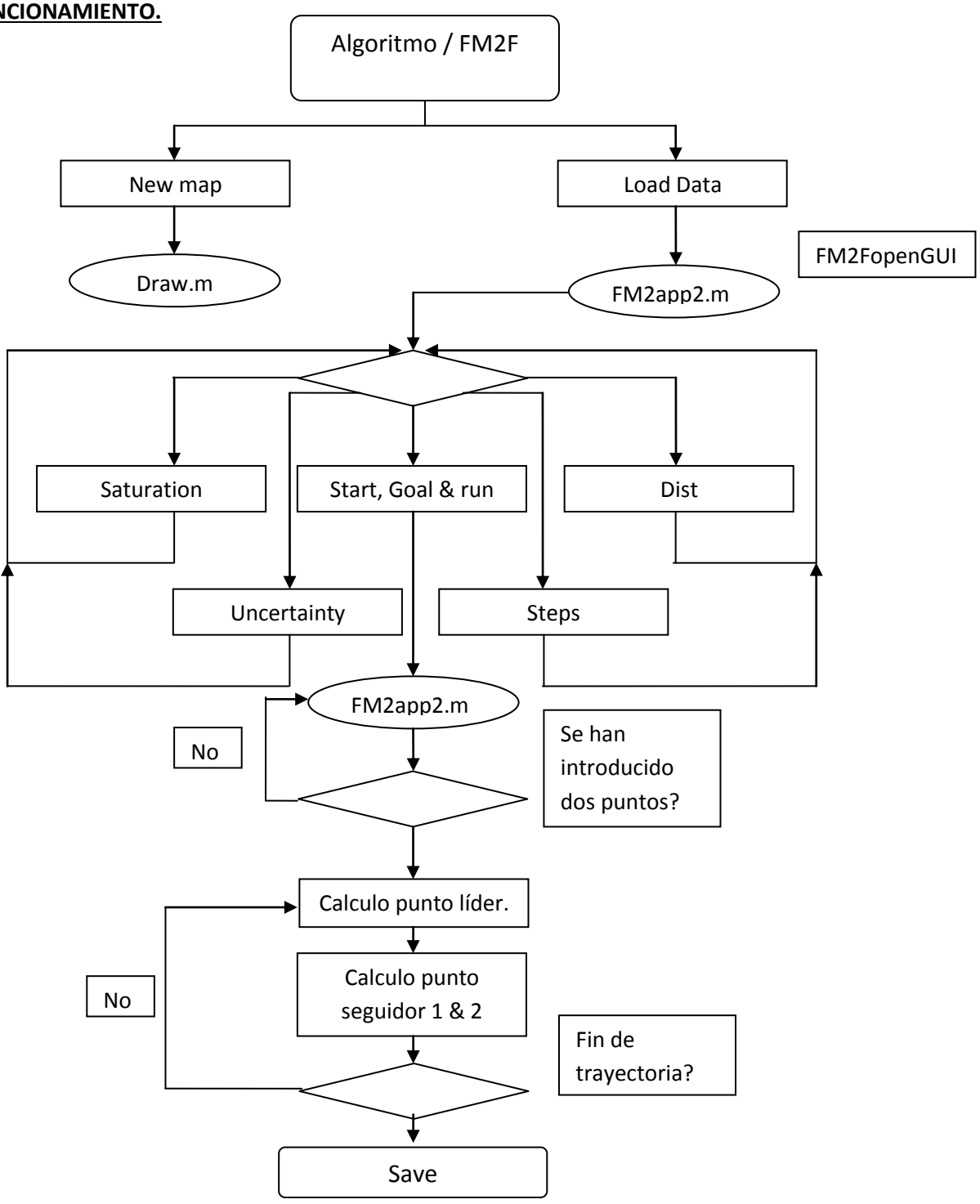
perteneciente al añadir el mapa o los workspaces guardados. En la zona media los parámetros, y más abajo el botón/es para introducir puntos y ejecutar el algoritmo. Siguiendo este diseño, se facilita el seguir los pasos en orden y de la manera correcta.

Además, para ayudar al usuario en todo momento a que nos se pierda, se han añadido user hints (consejos en cuadros de texto) que varían en función del paso en el que estemos. De esta manera, los user hints siguen el diagrama representado al principio de esta sección.

4.2.3 FM2F (Fast Marchine Square Formations)

Este último algoritmo tiene como objetivo analizar el efecto del algoritmo robots Formatios ajustando parámetros como la saturación, la incertidumbre, la distancia entre robots o el número de pasos, generando una animación de robots en una formación y mostrar dos gráficas donde se muestren los resultados de la simulación.

FUNCIONAMIENTO.



DIARGRAMA 5. "*funcionamiento interfaz FM2F*"

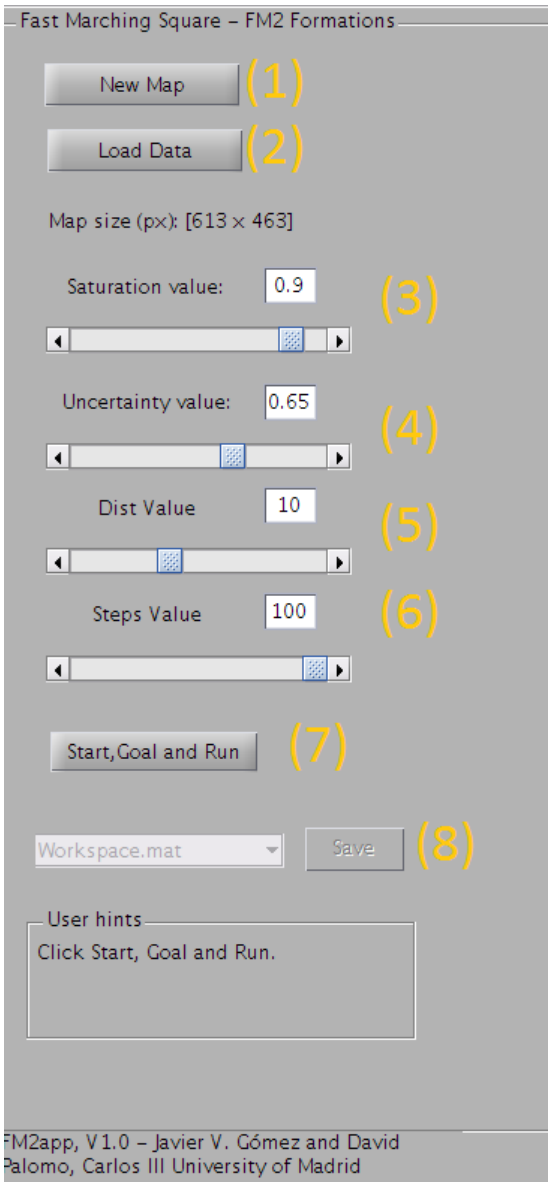


FIGURA 29. "*Panel FM2F*"

Como se explica en el diagrama 5, una vez ejecutada la interfaz fm2app, aparecerá un panel vacio. Para poder seleccionar el panel correspondiente al algoritmo deseado, en la barra de herramientas, se selecciona la opción Algorithms. Una vez pulsado aparecen diferentes nombres de algoritmos, para este caso se selecciona FM2F.

Una vez seleccionado, aparecerá un panel como el de la figura 29, pero con solo dos campos habilitados:

- New map (1): esta opción ejecuta la interfaz gráfica Draw, la cual nos permite diseñar un mapa para posteriormente utilizarlo.
- Load Data (2): este botón permite al usuario cargar un mapa disponible.

En caso de cargar un mapa, se activara y aparecerá por pantalla la interfaz fm2app2 y una ventana emergente, donde se podrá seleccionar un mapa que queramos en formato *bmp,

*png, *mat (Wokspace FM2F) y *mat (New map create). Una vez seleccionado, aparecerá el mapa seleccionado (imagen 1, figura 30) y el mapa saturado (imagen 2, figura 30) para poder visualizar el efecto al ajustar la saturación. En la sección de la interfaz fm2app2 correspondiente a FM2F se explica con mayor detalle.

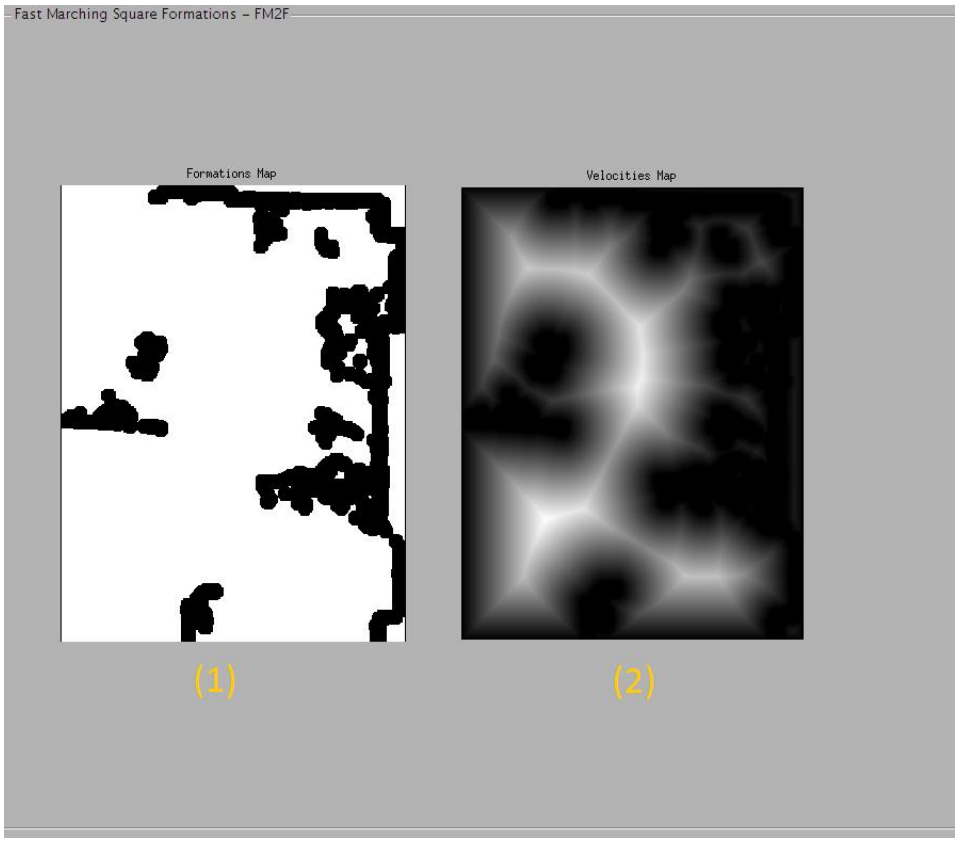


FIGURA 30."*Mapas de obstáculos y velocidad*"

Una vez cargado el mapa que se utilizara en la demostración, se habilitaran las siguientes opciones:

- Saturation (3): permite ajustar la distancia de seguridad entre los obstáculos y los robots de la formación.
- Uncertainty (4): habilita la posibilidad de aumentar o disminuir la incertidumbre sobre la cual se calcula cada uno de los movimientos de la formación.
- Dist (5): ajusta la distancia entre los seguidores y el líder de la formación.
- Steps (6): permite establecer mayor o un menor número de cálculos por etapa.
- Start, Goal and Run (7): al pulsarlo podremos introducir el punto inicial y final de la animación y comenzar a ejecutar el algoritmo.

Saturation

En este caso, como en los dos algoritmos anteriores, este parámetro condiciona la distancia de seguridad entre los obstáculos y la formación. Este parámetro implica el aumento o disminución de la distancia recorrida. Para ajustar, se utiliza una *slider* y una *box* para visualizar el valor. Los valores que se pueden establecer 0 y 1. La variación de este parámetro es reflejada en el mapa de saturación, el cual se actualiza cada vez que es modificado el valor.

Uncertainty

Habilitado este campo, se puede ajustar el área de incertidumbre alrededor de los robots de la formación. Este valor se ajusta utilizando una *slider* y se visualiza en una *box.* Este parámetro puede ajustarse a 1 (mínima incertidumbre) y 0 (amplia incertidumbre), condicionando un mayor movimiento de los seguidores, disminuyendo la distancia entre ellos. Una vez ajustado este valor y ejecutándose la animación, se podrá visualizar en el mapa de saturación el área de incertidumbre como se muestra en la figura 31.

FIGURA 31."*Mapas de incertidumbre*"

Dist

Parámetro que configura la distancia que tiene que haber entre los dos seguidores y el líder en perpendicular. Se utiliza, como en los demás parámetro una *slider* para ajustar el valor y una *box* para visualizarlo. Este parámetro tendrá como valor mínimo 1 y como máximo 25. Dist condicionara junto con otros parámetros el mantener o no la formación ante la aparición de obstáculos y como abordarlos. En la figura 32 se representa la formación, donde d2 es el parámetro Dist.

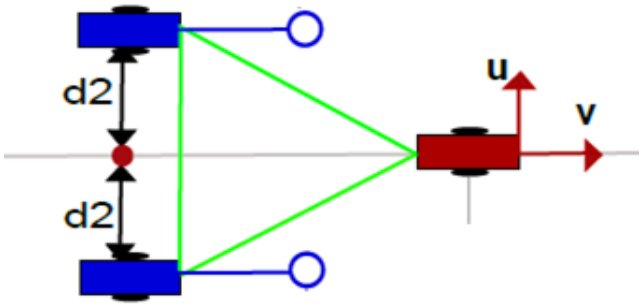


FIGURA 32. "*Distancia entre líder y seguidores*"

Steps

Esta opción permite configurar el número de cálculos que se realiza en cada movimiento de la formación. Los valores configurables serán 1 como mínimo, lo que aumenta el número de cálculos y 100, el número mínimo de cálculos. Se ajusta utilizando una *slider* y una *box*.

Start, Goal & Run

Una vez ajustados todos los valores, se procede a insertar los puntos inicial, donde se situara la formación; y el punto final. Para ello se debe de pulsar el botón Introduce Demos. Seguidamente aparecerá la interfaz gráfica fm2app2 donde están los mapas de demostración. Los puntos introducidos deben estar dentro del mapa y fuera de cualquier obstáculo. Para introducirlos se utiliza el botón derecho del ratón. Una vez introducidos los puntos, automáticamente se comienza a ejecutar el algoritmo y comienza a mostrarse la animación, paso a paso.

Finalizada la ejecución, se ejecutara una animación completa sobre la trayectoria realizada y seguidamente aparecen dos gráficas y se habilitara la opción save (8). El resultado final será como el de la figura 33.

Save

El botón save es el encargado de guardar la imagen o workspace seleccionado. Para seleccionar aquello que se quiere guardar, se utiliza el *popupmenu* disponible a la izquierda del botón (8). En él, se puede elegir entre las siguientes opciones, que están referenciadas a la figura 29:

- Workspace.mat: permite guardar el espacio de trabajo de la animación, guardando los los valores de saturación, incertidumbre, distancia entre seguidores y steps. También se almacenan los mapas de saturación y de animación, además de las dos gráficas de resultado. Este workspace será guardado en formato *mat.
- Map of animation (1): esta opción permite guardar el mapa de obstáculos con la formación de los robots. Se podrá salvar en formato *pdf y *eps.

- Map of Saturation/Uncertainty (2): permitirá guardar el mapa saturado y con la representación de la incertidumbre en los seguidores. El formato de guardar será en *pdf y *eps.
- Result Speed/Iteration (3): esta opción selecciona el grafico correspondiente a la variación de la velocidad en función de la iteración. Se guardara en formato *pdf y *eps.
- Result %dirección/iteration (4): selecciona el grafico que representa el porcentaje de modificación de la formación. Se guardara en formato *pdf y *eps.

Una vez seleccionado en el *popupmenu* entre las distintas opciones, se presiona el botón save y aparecerá una ventana emergente donde poder elegir el sitio donde guardarlo.

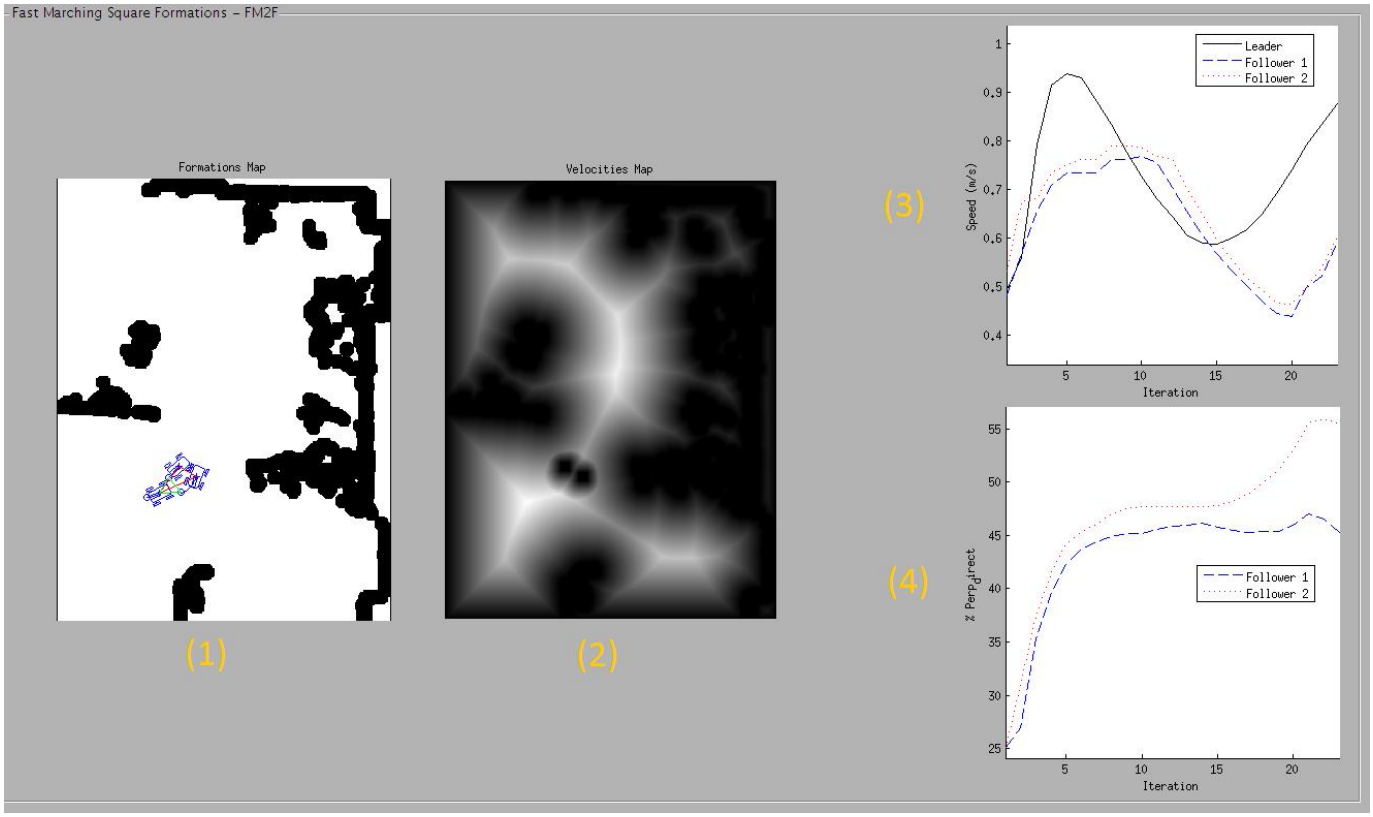


FIGURA 33. "*interfaz gráfica FM2F*"

Para diseñar esta interfaz, como en el algoritmo FM2 y FML, se ha seguido una estructura en que sigue el diagrama situado al principio de esta sección. Se sitúa en la zona superior la parte perteneciente al añadir el mapa o los workspaces guardados. En la zona media los parámetros, y más abajo el botón/es para introducir puntos y ejecutar el algoritmo. Siguiendo este diseño, se facilita el seguir los pasos en orden y de la manera correcta.

Además, para ayudar al usuario en todo momento a que nos se pierda, se han añadido también user hints (consejos en cuadros de texto) que varían en función del paso en el que estemos. De esta manera, los user hints siguen el diagrama representado al principio de esta sección.

4.3 FM2app2.m

Esta tercera interfaz está formada por los ejes que representan los mapas de cada algoritmo en donde se trabaja y se obtienen los resultados.

Como ocurría en la interfaz FM2app.m, se utiliza el menú editor para seleccionar entre las distintas ventanas de los algoritmos (uipanels). Todos los ejes se encuentran agrupados en un uipanel, debido a que es más práctico, puesto que si redimensionamos el uipanel, se redimensiona todo automáticamente, lo que resulta cómodo y sencillo. En la figura 34 se muestra el menú utilizado para editar la cantidad de ventanas desplegables que habrá disponible en la barra el menú al ejecutar la interfaz

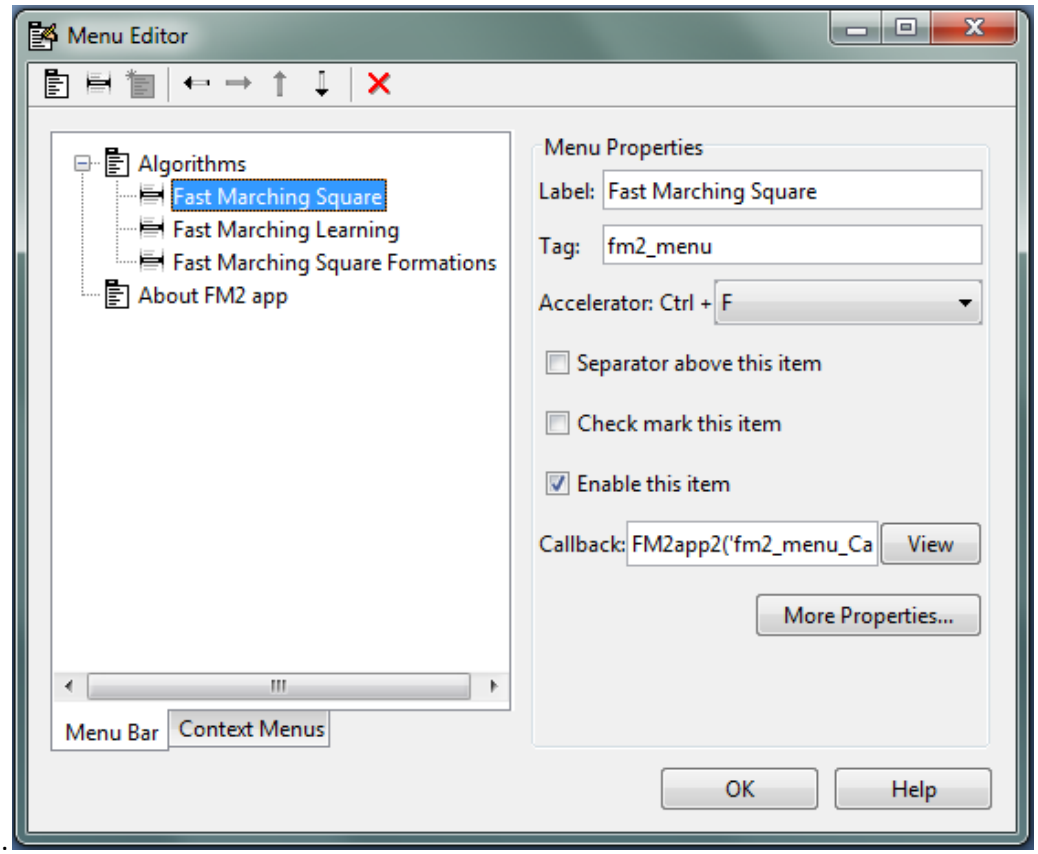


FIGURA 34. "*Menú editor de la interfaz FM2app2*"

4.3.1 FM2 (Fast Marching Square)

En este algoritmo, como se comento en la interfaz FM2app, tiene como finalidad conseguir encontrar la trayectoria más corta entre dos puntos en relación a su velocidad y la distancia recorrida. Para poder conseguir esto, en esta interfaz se muestra el mapa de demostraciones, el mapa de velocidades calculado en la anterior interfaz y el mapa de expansión de onda, junto con una gráfica de resultados de relación velocidad/distancia recorrida por el robot.

FUNCIONAMIENTO.

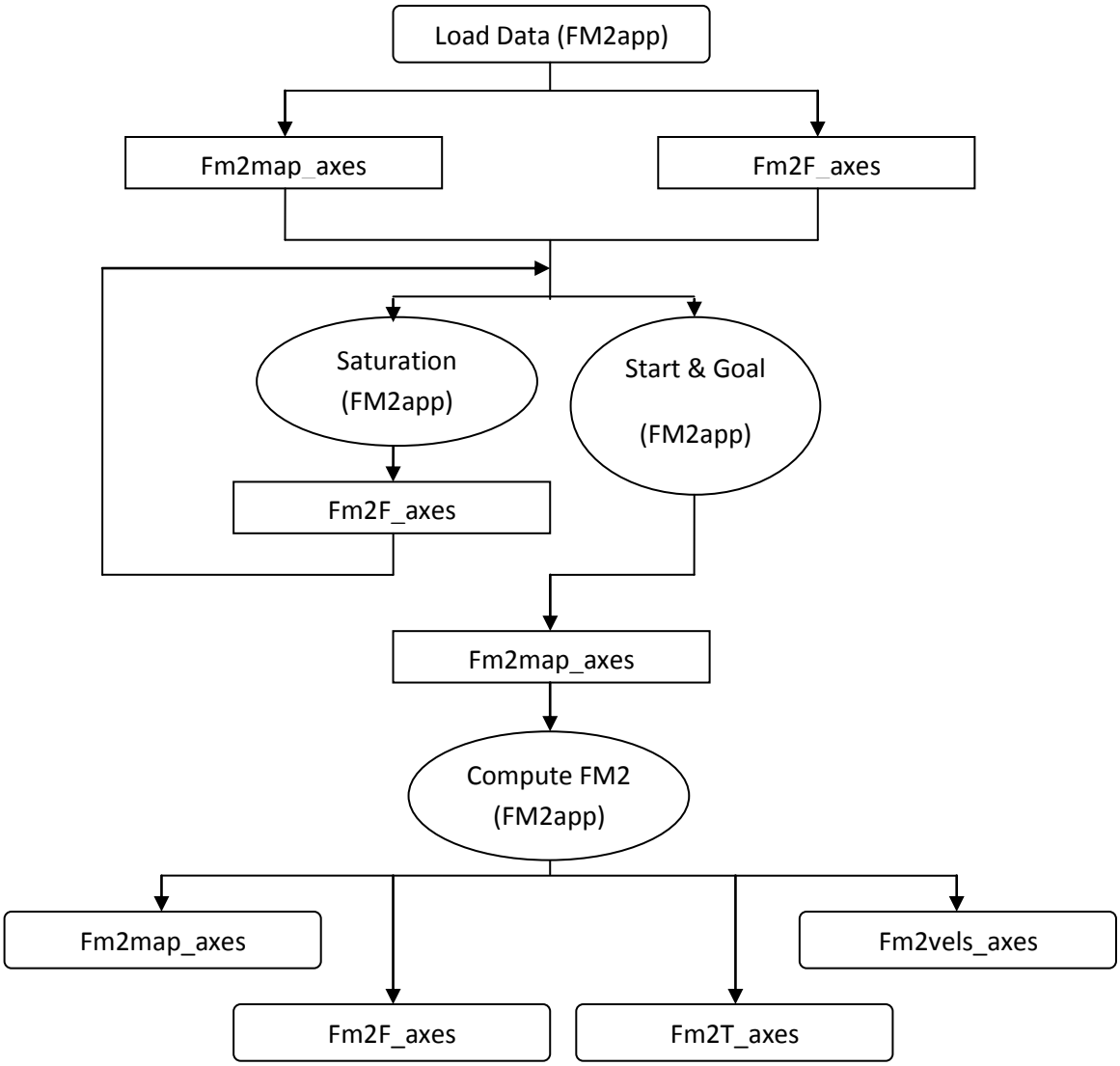


DIAGRAMA 6. "*Funcionamiento FM2 parte 2*"

Como se detalla en el diagrama 6, el funcionamiento de la parte gráfica del algoritmo FM2 comienza cuando se carga un mapa, pulsando el botón Load Data. Tras ser pulsado, se acedera a la interfaz fm2app2 y se abrirá una ventana emergente para cargar el mapa que se quiera utilizar. Una vez seleccionado, se cargara un mapa de dos formas distintas. Por un lado, como se cargara en el eje fm2map el mapa con los obstáculos, y por otro lado se cargara en el eje fm2sat el mismo mapa pero en función de la saturación.

Este mapa de saturación cambiara en función del valor que de asignemos. Para introducir los puntos una vez pulsado Start & Goal se utilizara el mapa de obstáculos cargado en el eje fm2map.

Finalmente, para poder obtener el resultado de la interfaz, se deberá activar Compute FM2. De esta manera obtendremos tres resultados:

- Fm2map: se representa la trayectoria final con el parámetro de saturación ajustado.
- Fm2T: este mapa representa la expansión de la onda utilizada para calcular la trayectoria.
- Fm2vels: grafico que representa la velocidad y distancia recorrida en la trayectoria.

A continuación se explica detalladamente cada eje de la interfaz para el algoritmo FM2:

- Fm2map_axes: mapa de demostración con obstáculos que tendrá el punto inicial y final, además de la trayectoria resultante, tal y como se muestra en la figura 35.

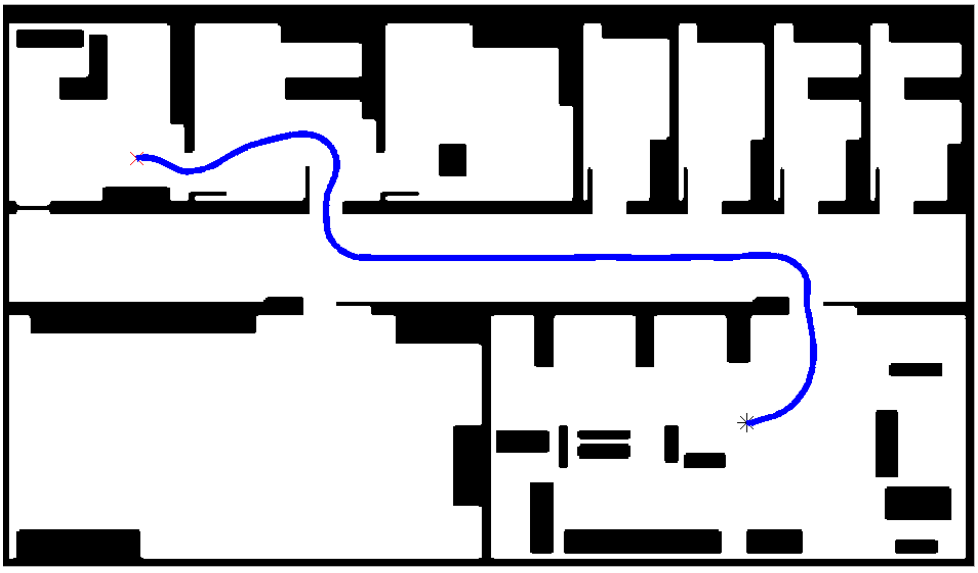


FIGURA 35."*Mapa de de demostración*"

- Fm2F_axes: en este mapa se muestra el mapa de velocidades explicado, en el cual se tiene en cuenta la saturación, en el ejemplo de la figura 36, se utiliza un valor de 1.se representa con 0 los obstáculos (negro) y los espacios vacios con 1 (blanco).

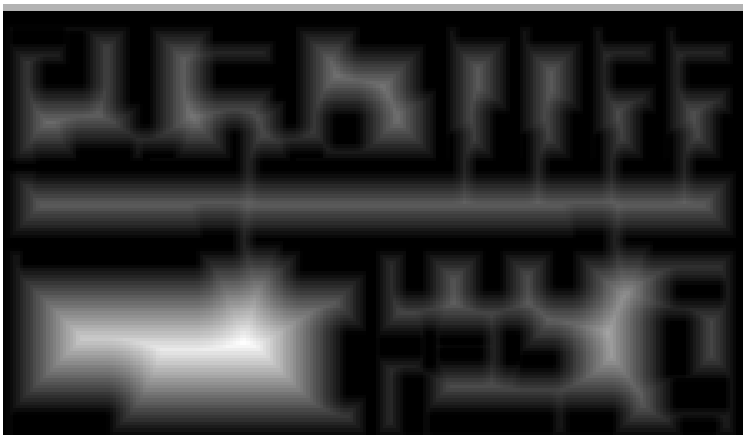


FIGURA 36. "*Mapa de velocidad F*"

- Fm2T_axes: mapa donde se representa la expansión de la onda para poder obtener la trayectoria más corta. Como se observa en la figura 37, el lugar más oscuro pertenece al inicio de la onda, y el color más claro de esta pertenece al punto inicial de la onda.

FIGURA 37."*Mapa de expansión de la onda, T*"

- Fm2vels_axes: gráfica de resultado donde se representa la velocidad alcanzada en el recorrido y la distancia recorrida, como se observa en la figura 38.

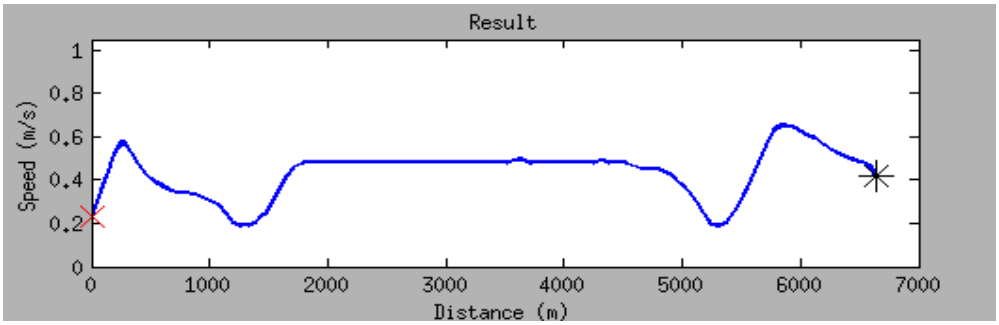


FIGURA 38. "*Gráfica de resultados*"

Como se muestra en la figura 39, así es como queda la disposición de la interfaz en el apartado del algoritmo FM2.

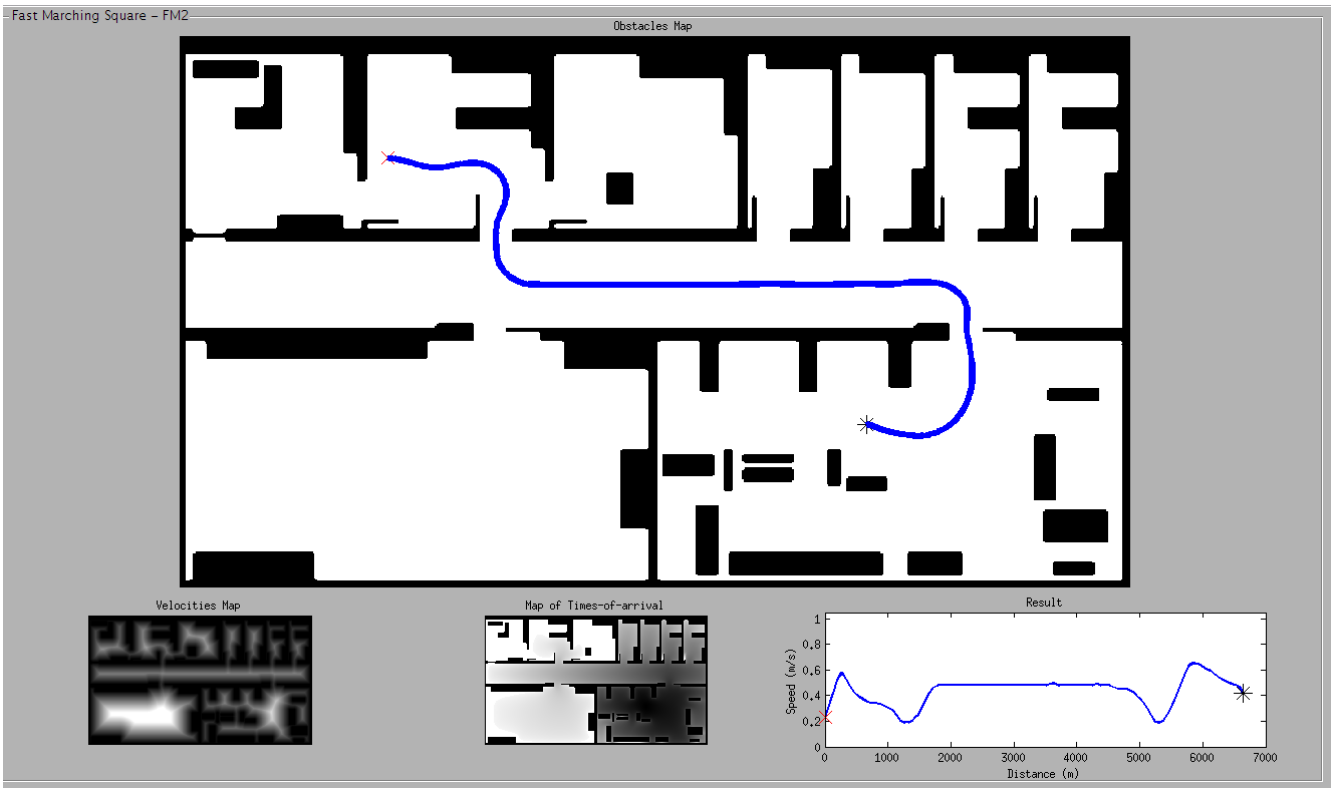


FIGURA 39. "*Disposición final FM2app2/FM2*"

4.3.2 FML (Fast Marching Learning)

Esta ventana de la interfaz FM2app tiene como objetivo obtener trayectorias de aprendizaje, que nos permita entender cómo afecta la saturación o el área de influencia en las trayectorias.

FUNCIONAMIENTO.

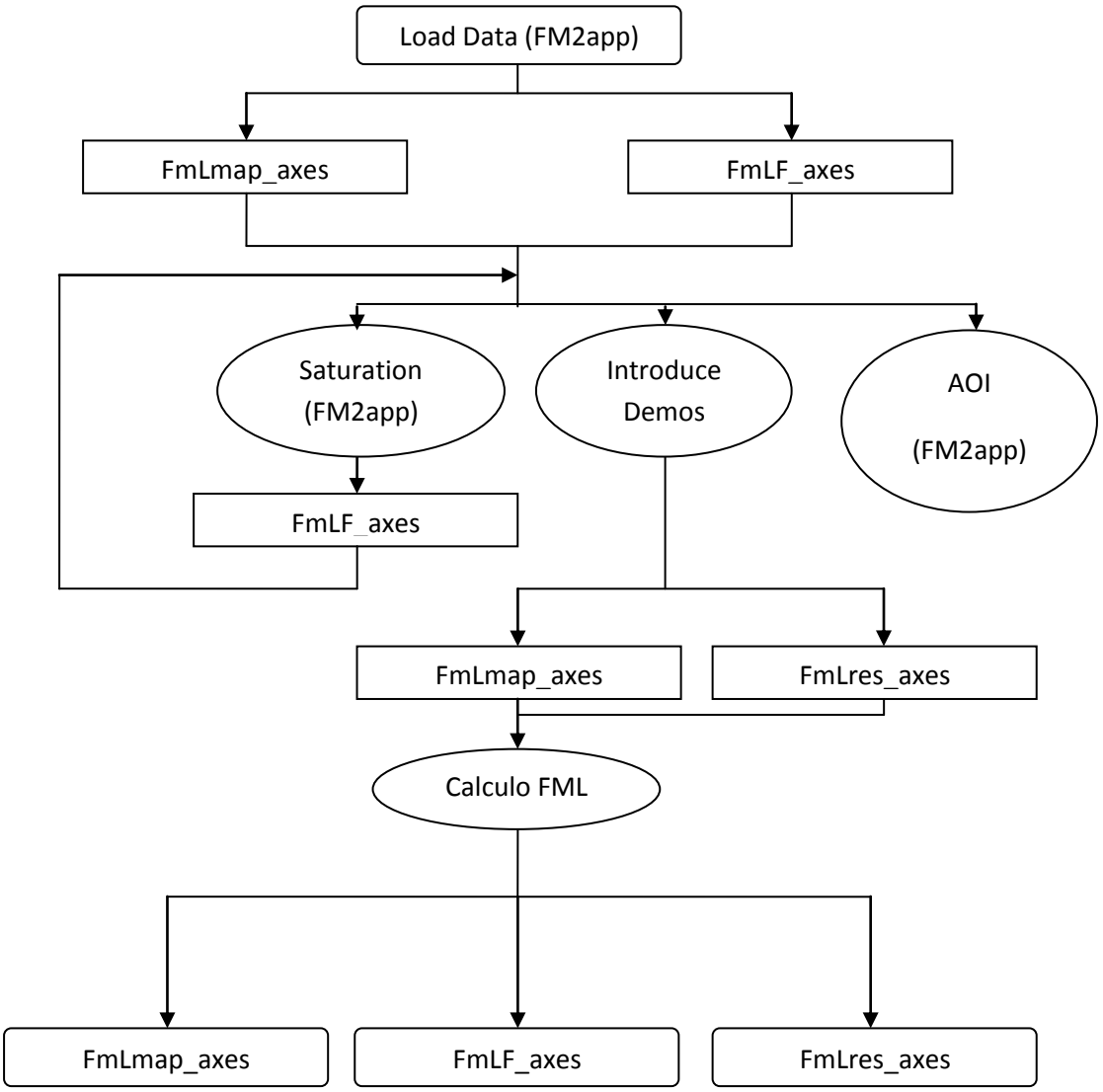


DIAGRAMA 7. "*Funcionamiento FML 2 parte*"

Como se detalla en el diagrama 7, la primera interactuación con la interfaz gráfica fm2app2 en el caso del algoritmo FML es al pulsar Load Data en la interfaz fm2app. Esta opción permite cargar el mapa que se utilizara para la demostración. Este mapa se cargara en el eje fmLmap. Además, se cargara el mapa seleccionado en función de la saturación en el eje fmLF. Este mapa cambiara cada vez que se asigne un nuevo valor de saturación.

Si variamos el valor del área de influencia, solo se verá representado en el mapa que se obtenga con la demostración.

Después de ajustar los valores, para poder introducir los puntos que forman la demostración, se utilizara el mapa de obstáculos situado en el eje fmLmap. Paralelamente, se dibujaran los puntos en el eje fmLres. Una vez introducido todos los puntos y pulsado otro botón, se actualiza los mapas situados en los ejes fmLmap, representando la trayectoria de aprendizaje y fmLF representando los puntos que conforman la trayectoria en función de la saturación.

A continuación se detallan los ejes presentes en la interfaz.

- FmLmap_axes: primeramente será el mapa de obstáculos donde se introducirán los puntos de cada demostración, para finalmente tener el mapa de resultados con las trayectorias de aprendizaje, como en la figura 40.

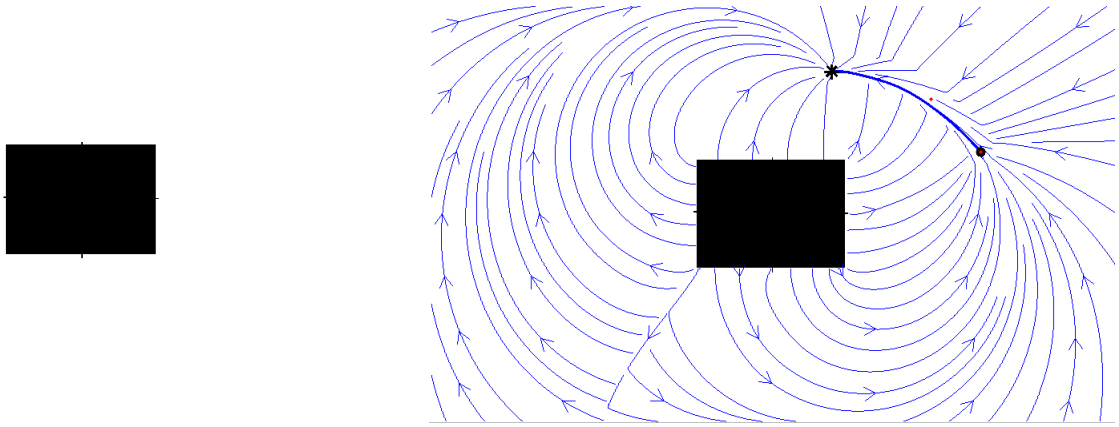


FIGURA 40. "*(izquierda) Mapa de obstáculos; (derecha) mapa de resultados*"

- FmLF_axes: mapa de velocidades en función de la saturación. Como se ha explicado en el anterior apartado, se representa con 0 los obstáculos (negro) y los espacios vacios con 1 (blanco). Un ejemplo es la figura 41, donde se aplica una saturación de 1.

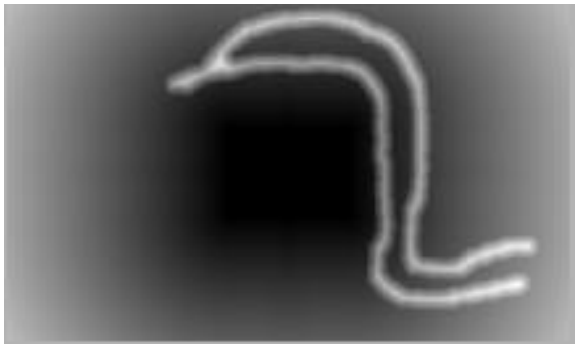


FIGURA 41. *"Mapa de velocidad F"*

- FmLres_map: este mapa registra todos los puntos de cada demostración que se ha introducido, lo que facilita la comprensión y mejora la disponibilidad de la información, como se muestra en la figura 42.

FIGURA 42."*Mapa de demostraciones*"

Finalmente, la distribución de los ejes del algoritmo FML quedara de la siguiente manera:

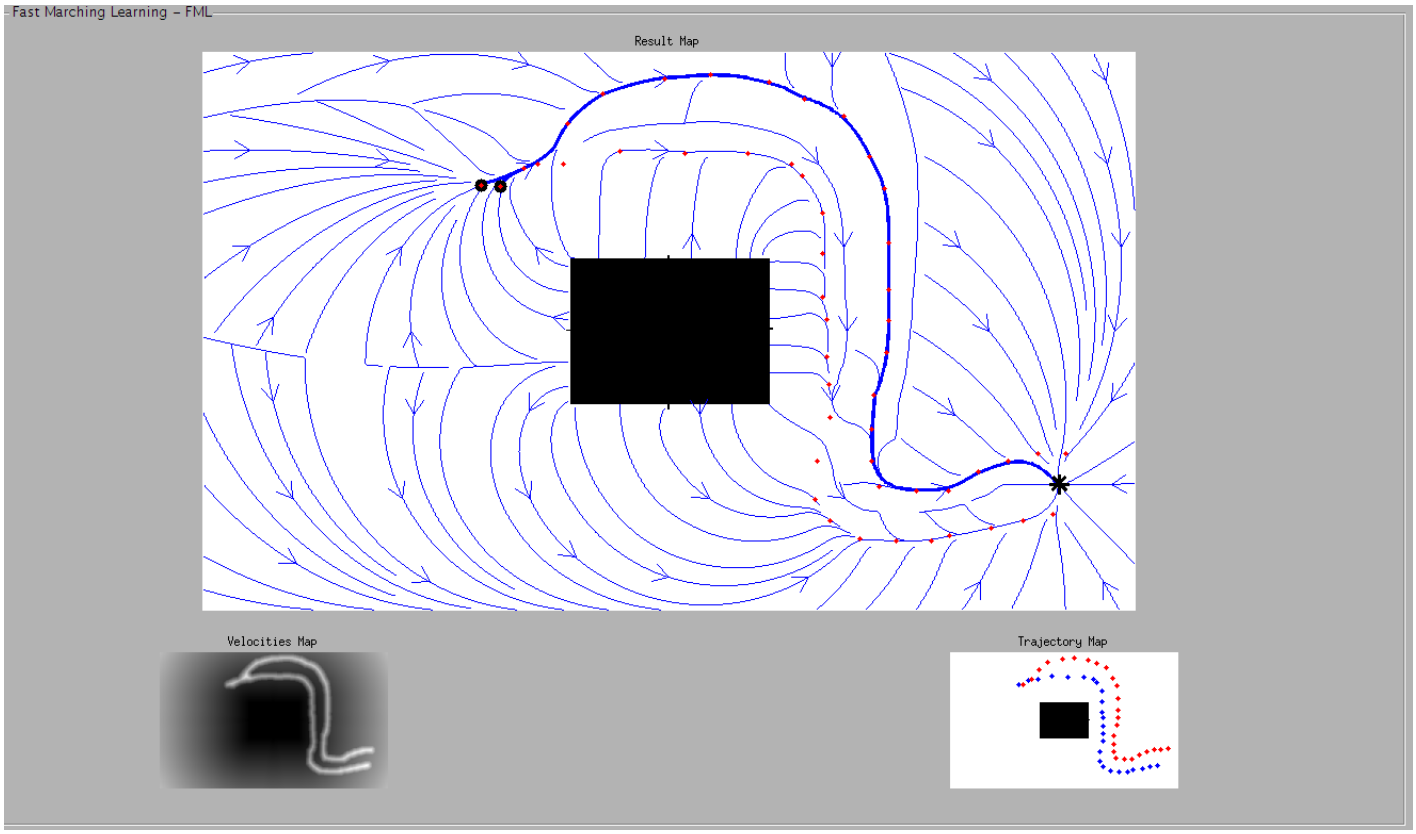


FIGURA 43.”*Distribución de FML*”

4.3.3 FM2F (Fast Marching Square Formations)

Este último algoritmo tiene como objetivo analizar el efecto de una formación de robots ajustando parámetros como la saturación, la incertidumbre, la distancia entre robots o el número de pasos, generando una animación de robots en una formación y mostrar dos gráficas donde se muestren los resultados de la simulación.

FUNCIONAMIENTO.

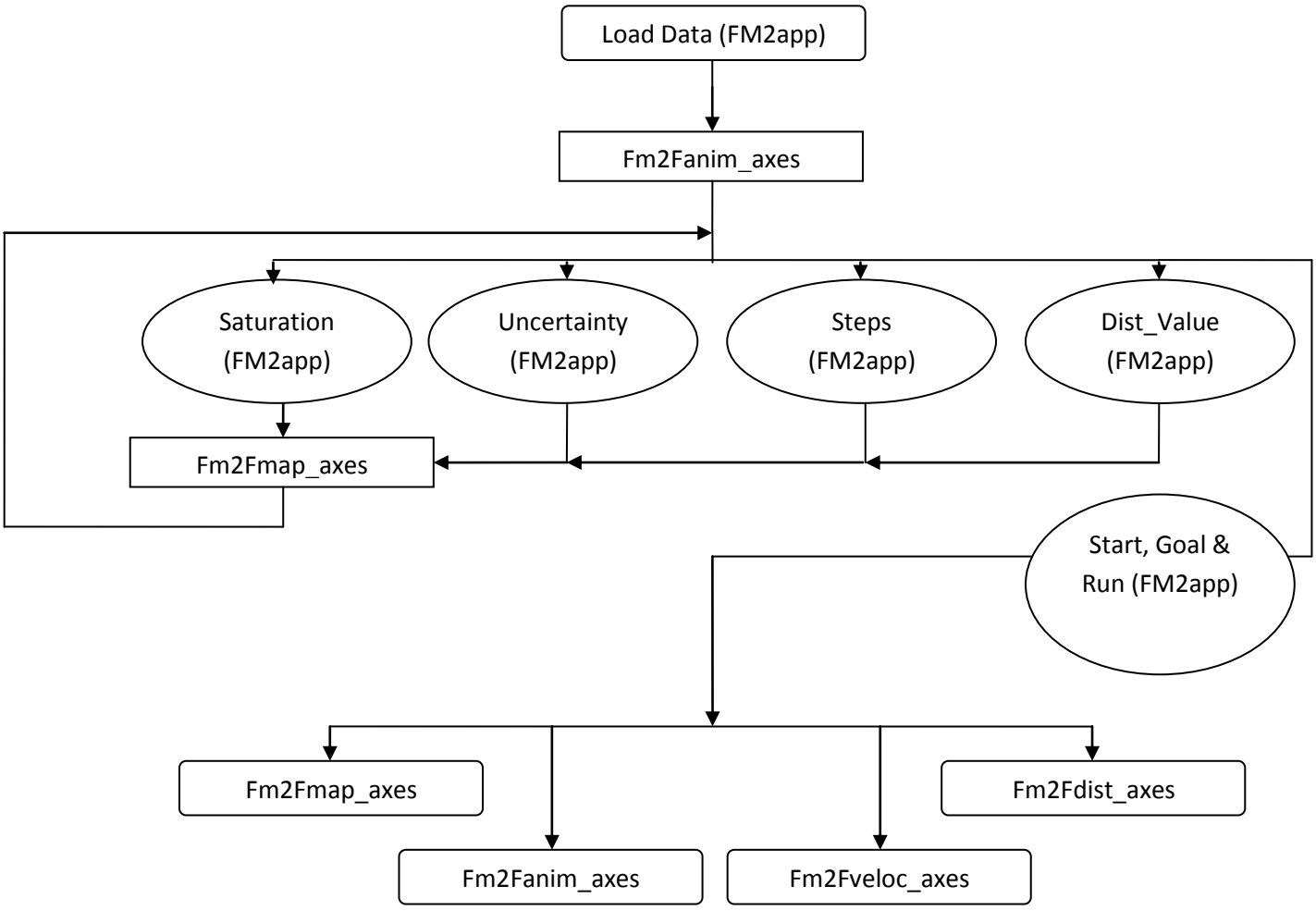


DIAGRAMA 8.” *funcionamiento interfaz FM2F 2 parte*”

Como se explica en el diagrama 8, la primera vez que se activa la interfaz gráfica fm2app2, como pasaba en los algoritmos anteriores, es al activar el botón Load Data. Esta opción carga un mapa que será utilizado para la animación. Este mapa será cargado en fm2Fmap. Además se cargara el mapa en función de la saturación e incertidumbre en el eje fm2Fanim.

Una vez cargados los mapas, se podrá ajustar los parámetros saturación, incertidumbre, steps y distancia de eje. La variación del parámetro de saturación actualizara el mapa saturado.

Una vez ajustados los parámetros, se procederá a introducir los puntos inicial y final. Una vez hecho esto, se comenzara a ejecutar la animación de formación dividiéndose en dos partes:

- Formación (fm2Fmap): se irá representando el movimiento de los tres robots que conforman la formación. Una vez finalizado, se realiza una animación del movimiento.
- Saturación e incertidumbre (fm2Fanim): se representara la formación y la trayectoria seguida con la influencia de la saturación y la incertidumbre dibujada alrededor de los robots que conforman la formación.

Finalizada la experiencia, se representaran dos gráficos en los siguientes ejes:

- Fm2veloc_axes: se representara la velocidad a lo largo de las iteraciones.
- Fm2fdist_axes: se representa el porcentaje de deformación de la formación.

A continuación se analiza más detalladamente los ejes de este algoritmo:

- Fm2Fmap_axes: en este mapa se representa la formación en su estado final y la trayectoria seguida.

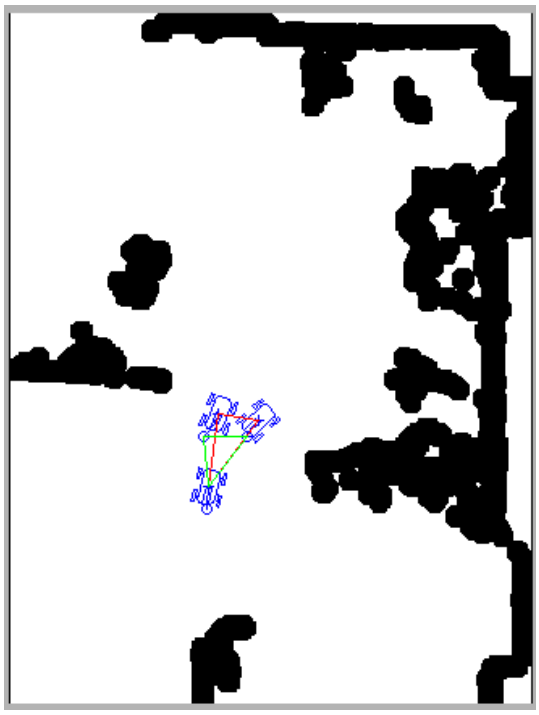


FIGURA 44.”*Mapa de formación*”

- Fm2fanim: se ejecuta al mismo tiempo que el anterior durante la simulación, y además de tener la representación de la saturación, veremos la influencia de la incertidumbre.

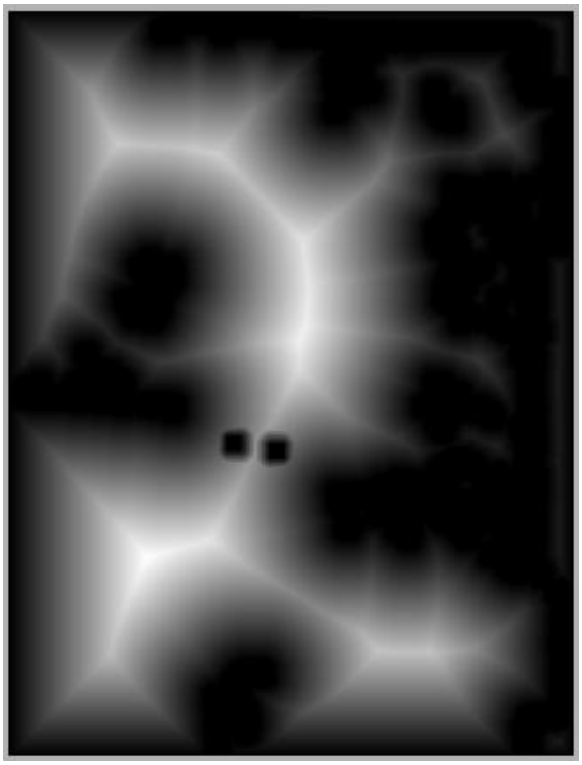


FIGURA 45."*Mapa saturado*"

- Fm2veloc_axes: se representa la velocidad en las iteraciones que se han hecho.

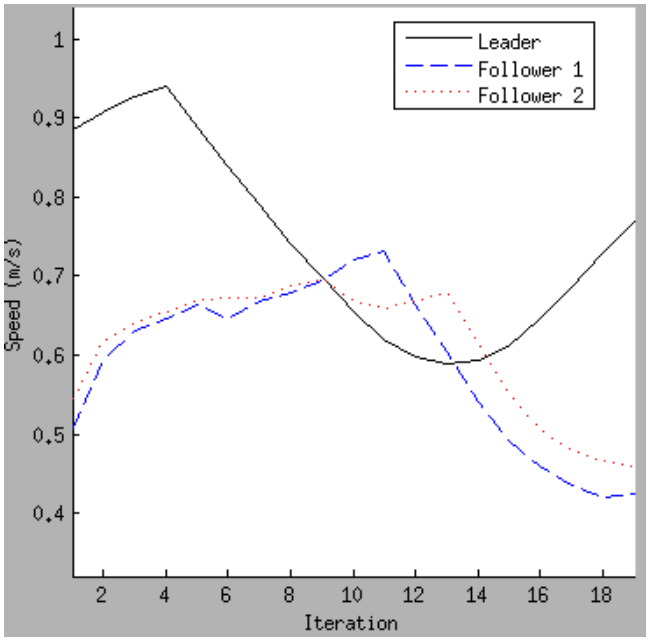


FIGURA 46."*Mapa Velocidad*"

- Fm2fdist_axes: se representa cuanto han tenido que modificar la formación los seguidores para esquivar los obstáculos.

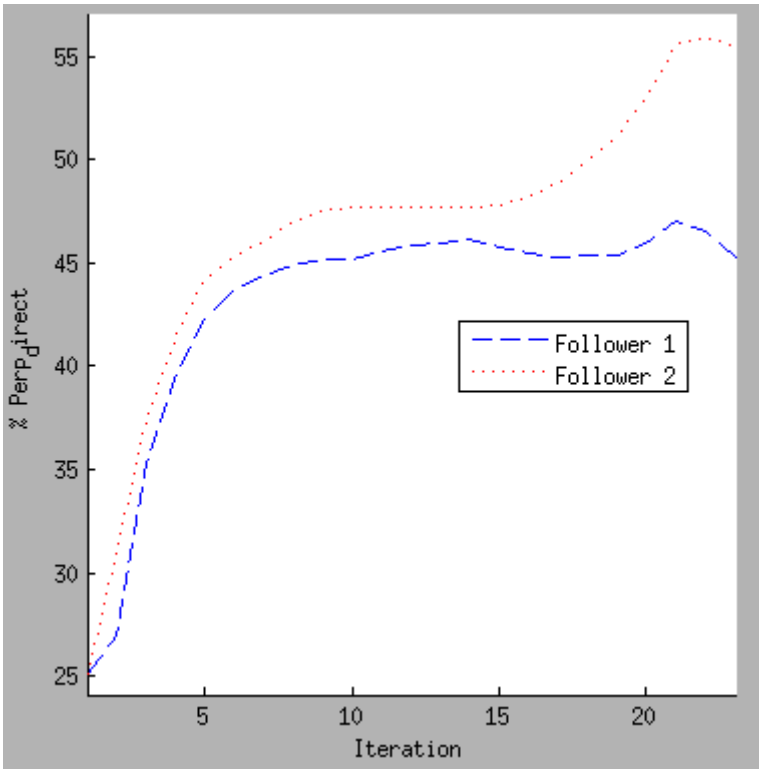


FIGURA 47. "*Mapa de modificación de trayectoria*"

La disposición final de los ejes será la mostrada en la figura 48, donde los dos mapas principales están en el centro permitiendo una mayor visualización en detalle y los resultados alineados en la derecha, resumiendo el proceso.

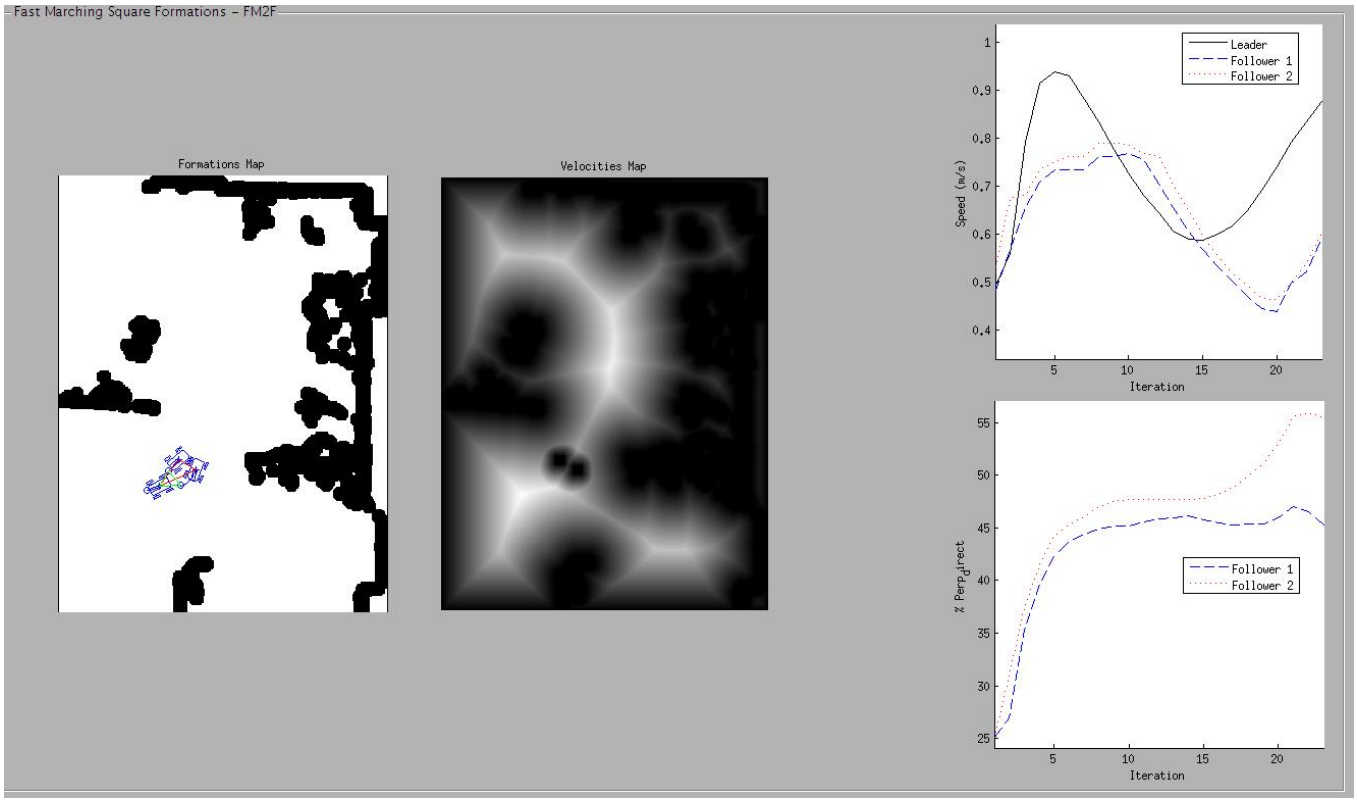


FIGURA 48."*Distribución FM2F*"

Sección 5: Demostración y Conclusión

En este último capítulo, y una vez habiendo explicado en qué consiste este proyecto, además de cómo está formado, solo nos queda comprobar que hemos alcanzado los objetivos que nos habíamos marcado en relación a la interfaz gráfica y los tres algoritmos.

Para una mayor claridad, para cada algoritmo se va a dedicar un apartado con diferentes demostraciones.

Demostración FM2

En esta demostración vamos a cargar un mapa de ensayo proporcionado y uno generado en la interfaz Draw. Se les someterá a dos pruebas el mapa de ensayo y se cargara un mapa generado. En las dos demostraciones del mapa de ensayo se usan valores extremos de saturación para poder entender si realmente se ha cumplido con los objetivos explicados en el marco teórico.

Para poder ejecutar el algoritmo, primero cargamos un mapa disponible en nuestra librería mediante Load Data. Una vez cargado el mapa de obstáculos y el de demostración, ajustamos el valor de saturación con la *slider*. Introducimos los puntos inicial y final con Start&Goal, como se muestra en la figura 49, para posteriormente ejecutar el algoritmo con Compute FM2.

FIGURA 49. "*Punto inicial y final FM2*"

-Mapa de ensayo:

En esta primera demostración (Figura 50), se ha utilizado un 100% de saturación, lo que condiciona la velocidad y la distancia que deja el robot con respecto a los obstáculos.

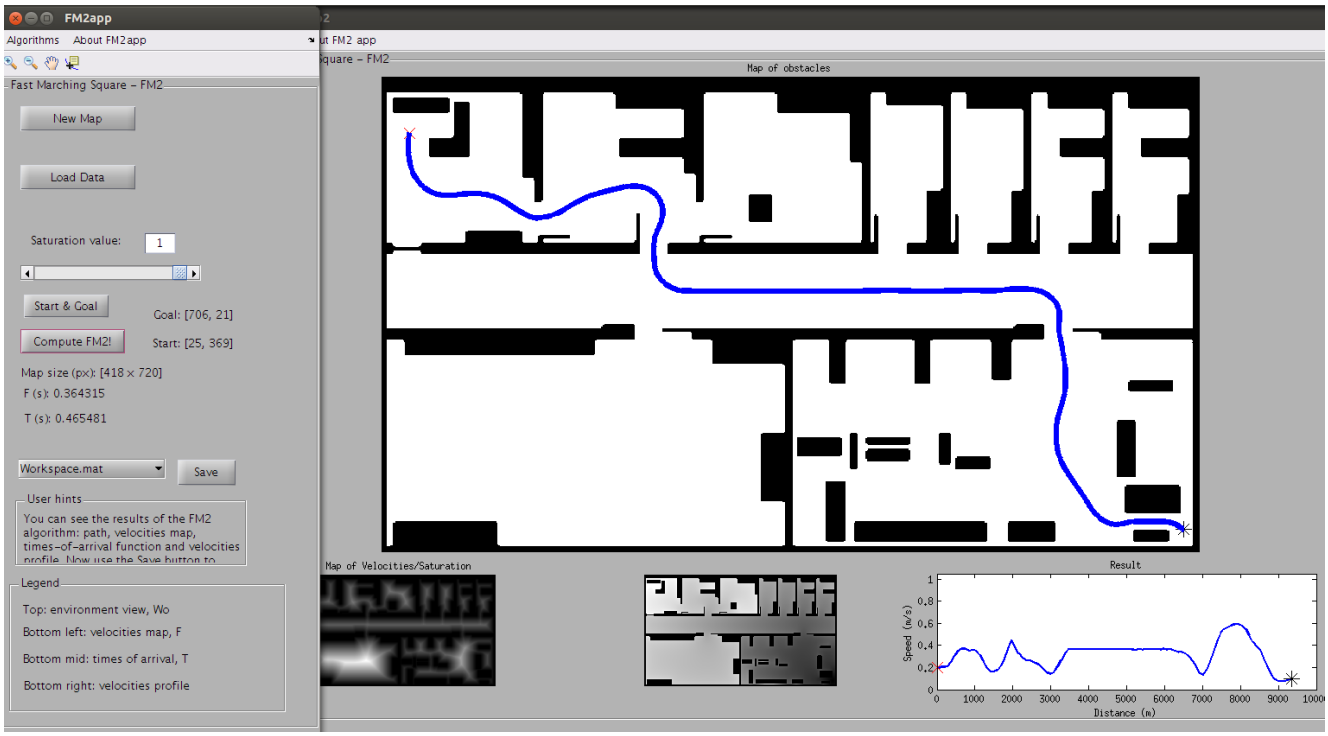


FIGURA 50.”*Demostración FM2 con sat=1*”

Se puede observar que cada vez que se pasa entre dos obstáculos, se hace justo en el medio, manteniendo la distancia de seguridad. En la siguiente demostración, se ha recalculado la misma trayectoria, pero con la saturación a 0% (figura 51).

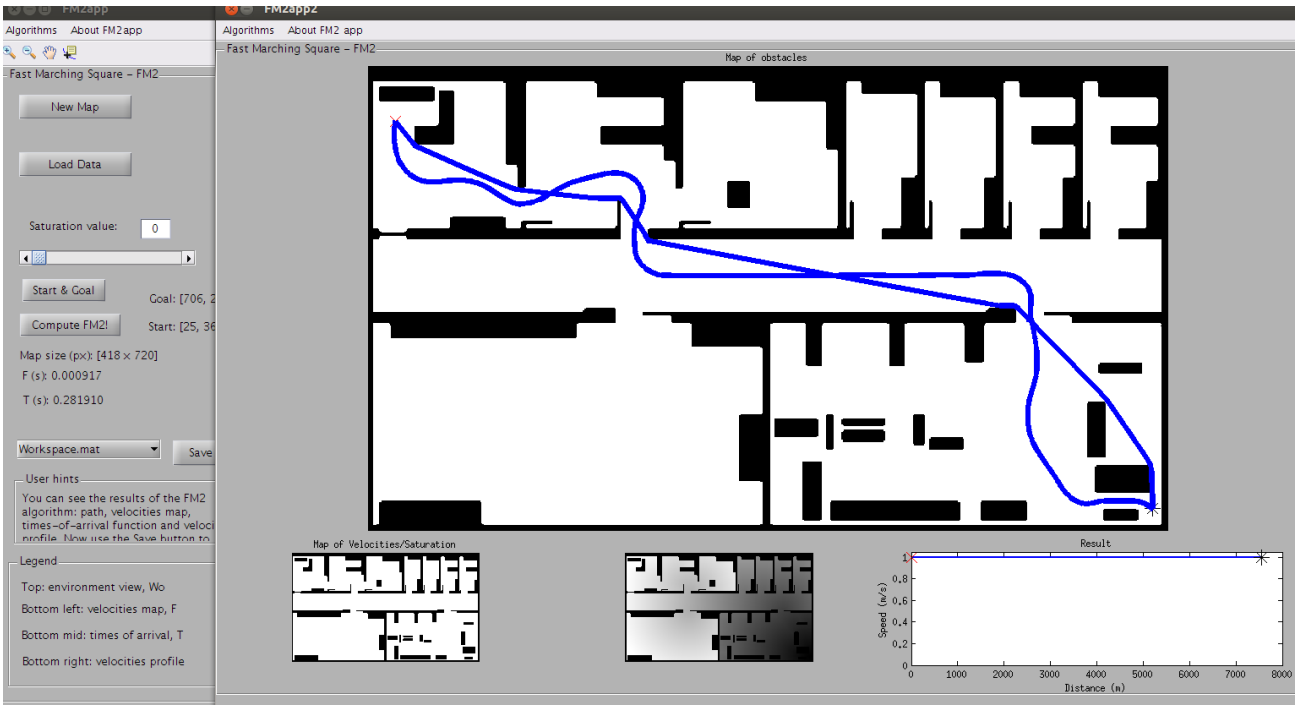


FIGURA 51.”*Demostración FM2 con un valor de sat=0*”

Si comparamos las dos demostraciones, la demostración de sat =0 tiene una mayor velocidad que la otra. Esto es debido a que no hay un margen de seguridad con respecto a los obstáculos del mapa, lo que condicionaría la velocidad. De esta manera, la demostración con sat=1 está sometida a dejar unos márgenes de seguridad con todos los obstáculos, lo que genera una reducción en la velocidad.

Si el experimento adoptara un valor de sat = 0.5, tendríamos una gráfica de resultado como la figura 52, donde se alcanzan distintas velocidades debido a la presencia de la saturación, lo que implica descensos en la curcas o al aproximarse a un obstáculo.

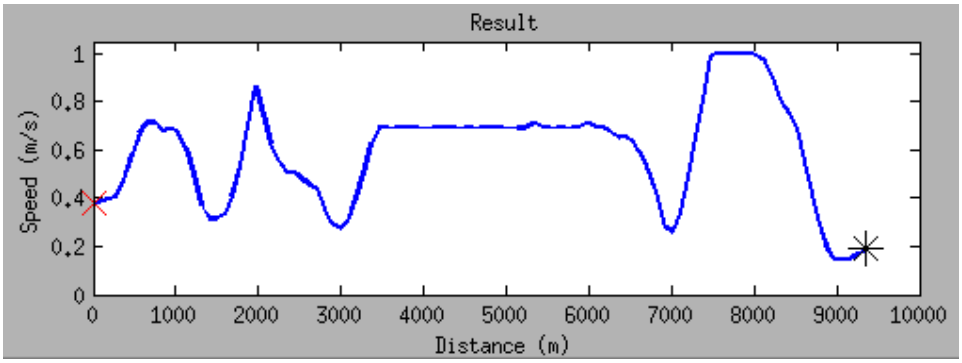


FIGURA 52. "*Resultado FM2 sat = 0.52*"

-Mapa generado: En este caso se ha conseguido cargar un mapa creado en la interfaz Draw y se ha conseguido realizar la demostración con éxito, como se muestra en la figura 53.

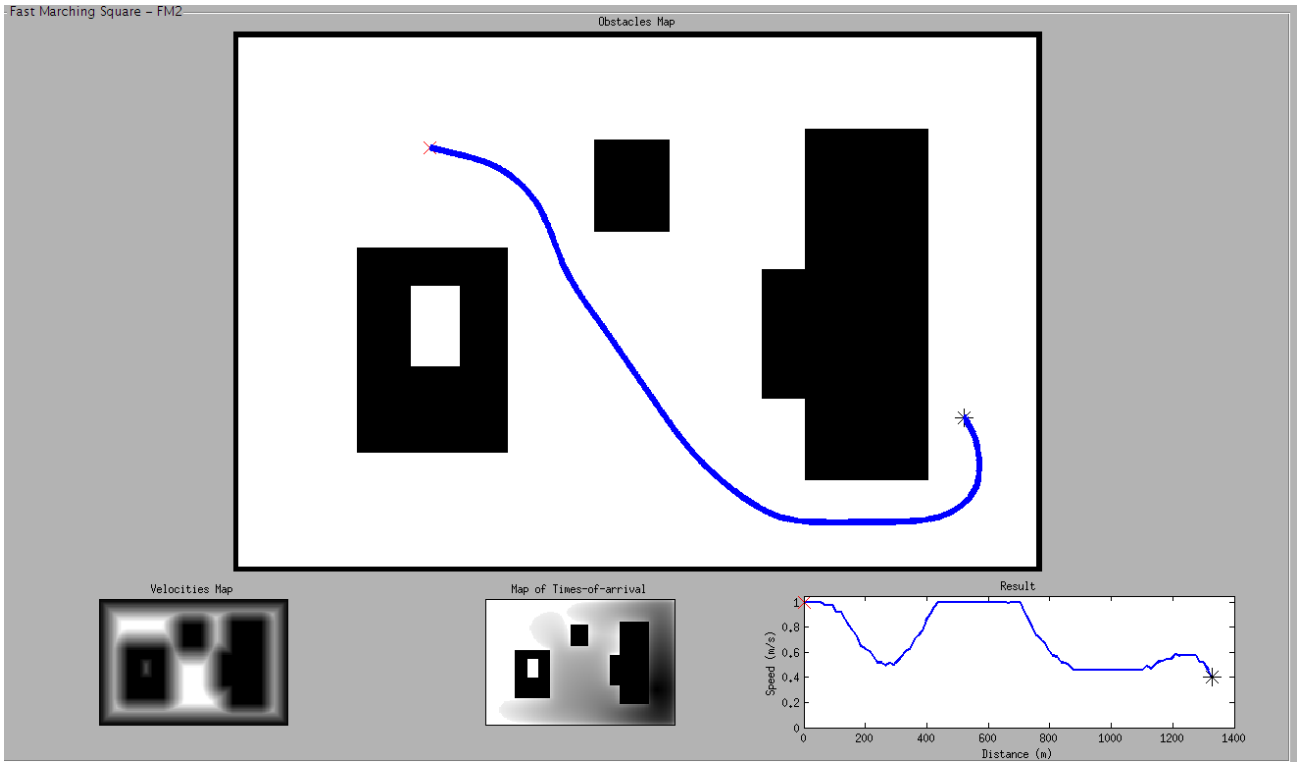


FIGURA 53."*FM2 éxito mapa generado Draw*"

Demostración FML

En esta demostración se van a utilizar también dos mapas. En este caso, vamos variar el valor de AOI, para observar su influencia, puesto que el valor saturación ya ha sido analizado en la anterior demostración. Además se cargar una trayectoria.

Para comenzar, primero cargamos un mapa mediante Load Data. Una vez cargado el mapa, introducimos el número de demostraciones, que para este caso serán 2. Ajustamos los valores de AOI. Con introduce Demos, comenzamos a introducir los puntos de las demostraciones y pulsando un botón distinto al botón izquierdo del ratón, se ejecuta el algoritmo. Al empezar a introducir las demos se mostrara un mensaje como el de la figura 54.

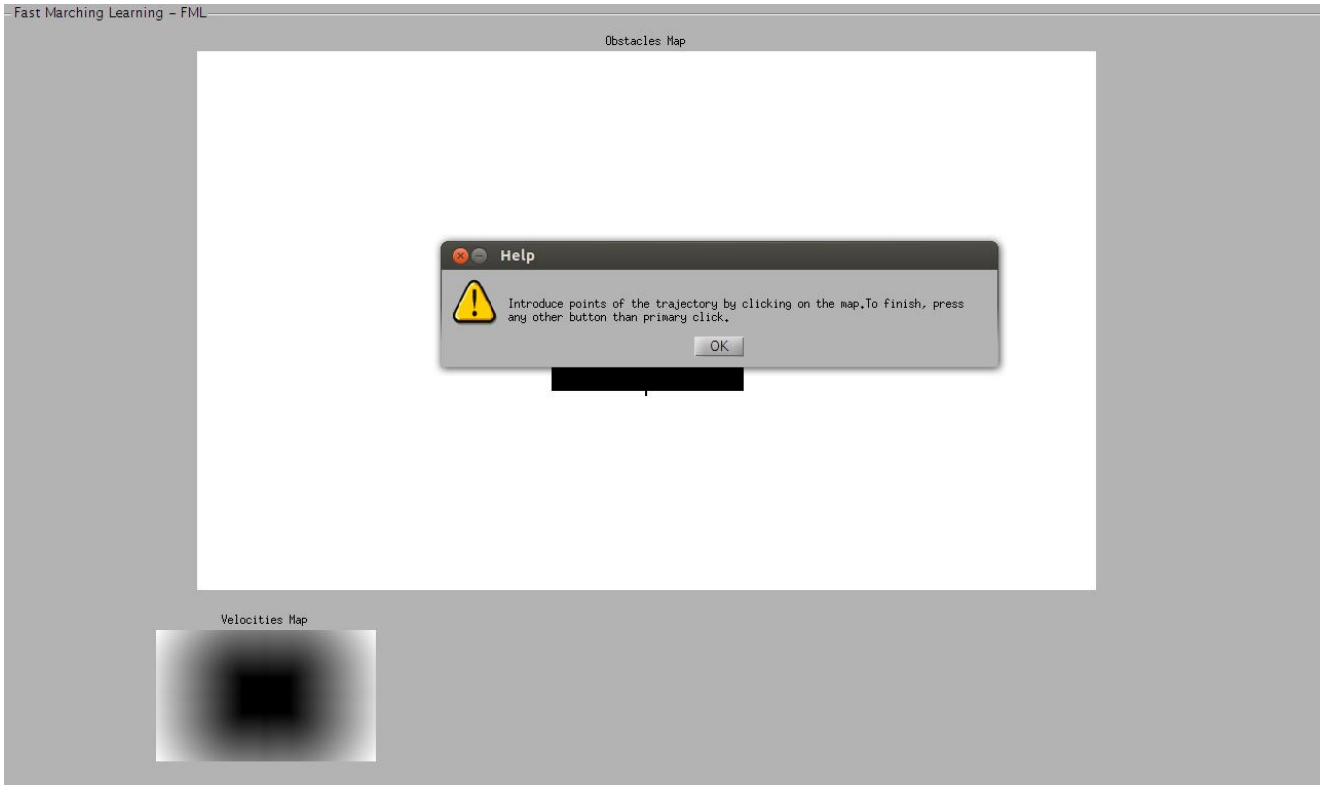


FIGURA 54.”*Introducción demostraciones en FML*”

-Mapa de ensayo:

En la figura 55 se aprecia que la trayectoria final hay una influencia de la saturación, debido a que la trayectoria no sigue la trayectoria de aprendizaje. En este primer ensayo, se ha utilizado un AOI mínimo.

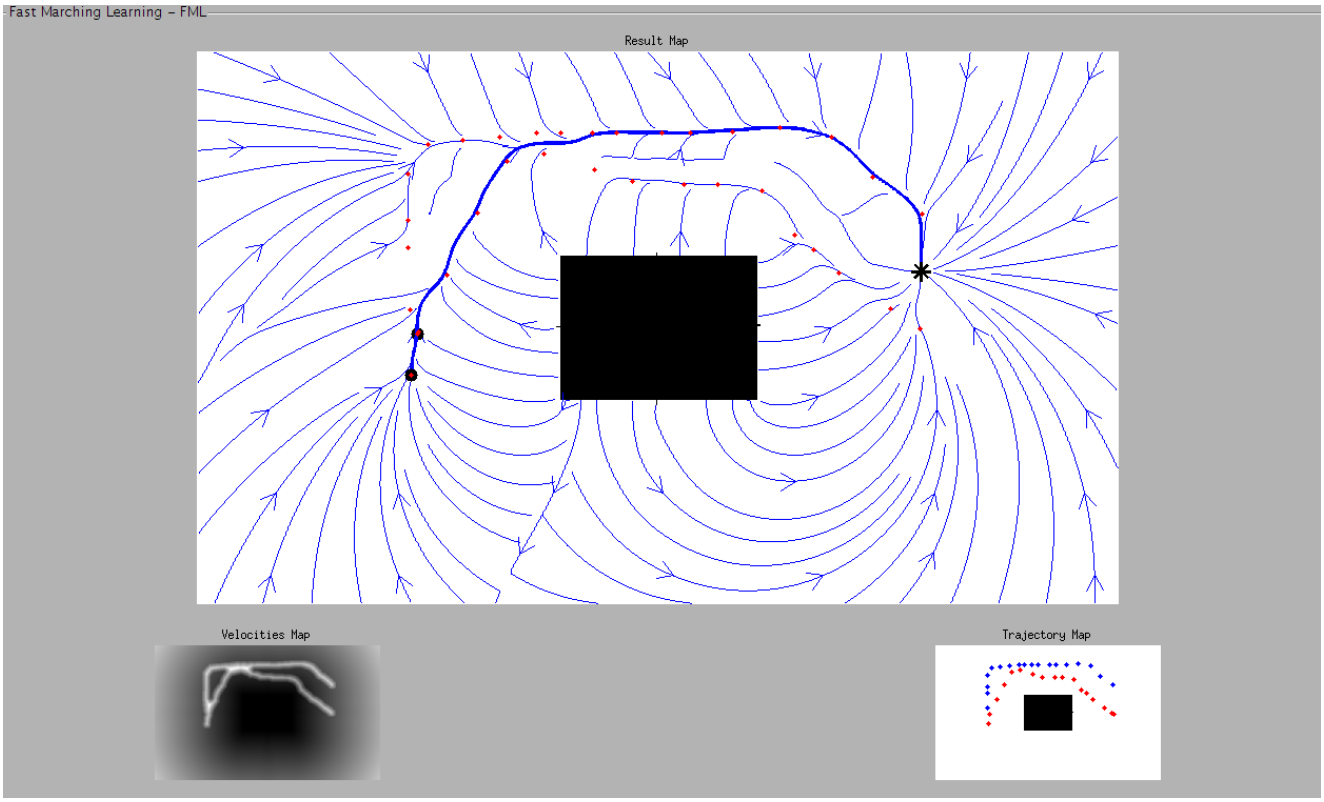


FIGURA 55."*FML con 2 demostraciones y sat=1 y AOI = 10*"

Por el contrario, en la figura 56, al aplicar un valor de saturación más pequeño, la trayectoria sigue la trayectoria de aprendizaje.

El último ensayo, representado en la figura 57, se utiliza un AOI de valor 65. Este valor dilata los puntos introducidos, haciendo que la trayectoria pase entre las trayectorias de aprendizaje.

En la figura 58 se muestran dos imágenes correspondientes a la demostración de que se pueden guardar trayectorias. En la imagen 1 se muestra el mapa del cual se guarda la trayectoria. En la imagen 2 se muestra la trayectoria cargada en otro mapa. Al utilizar esta función, se deben utilizar mapas de tamaño similar, si no, ocurre lo mostrado en esas dos imágenes.

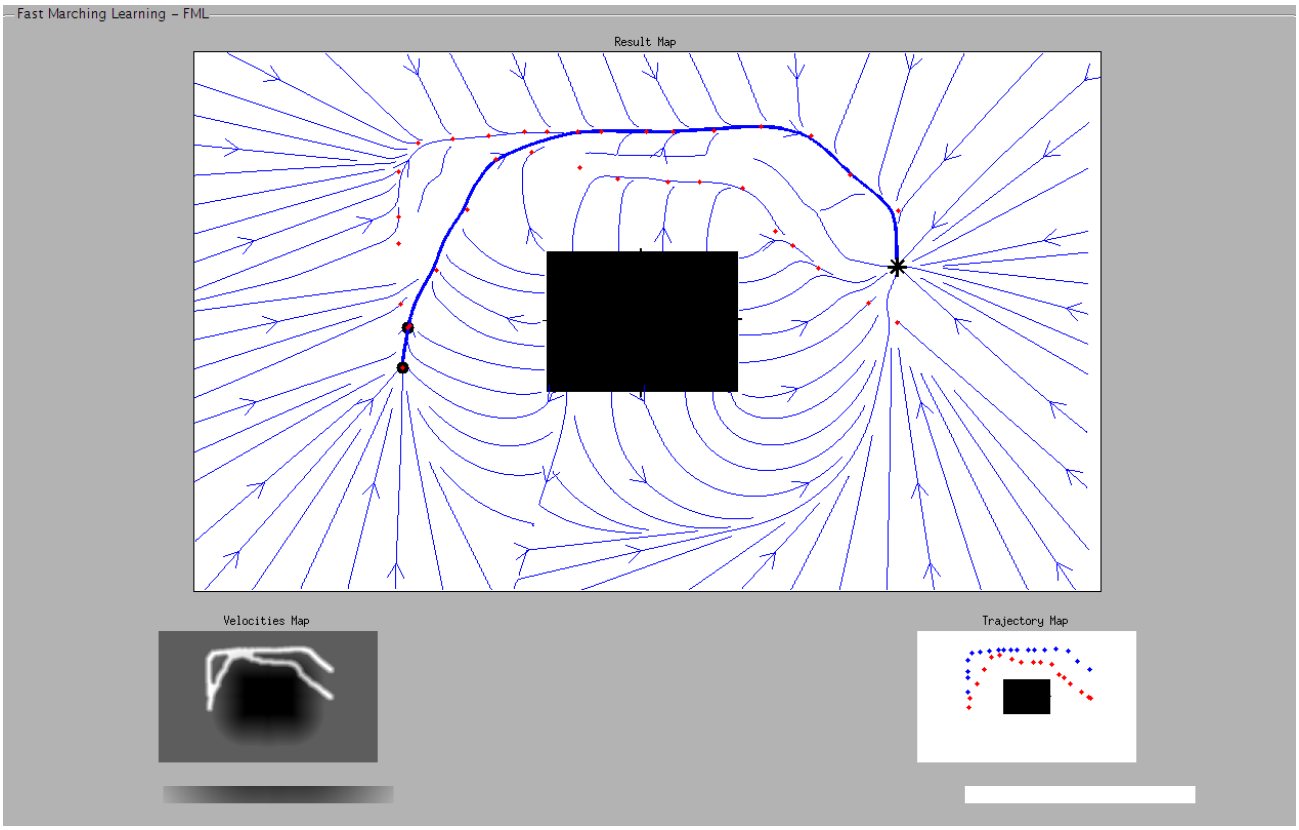


FIGURA 56.”*FML con sat=0.6 y AOI =10*”

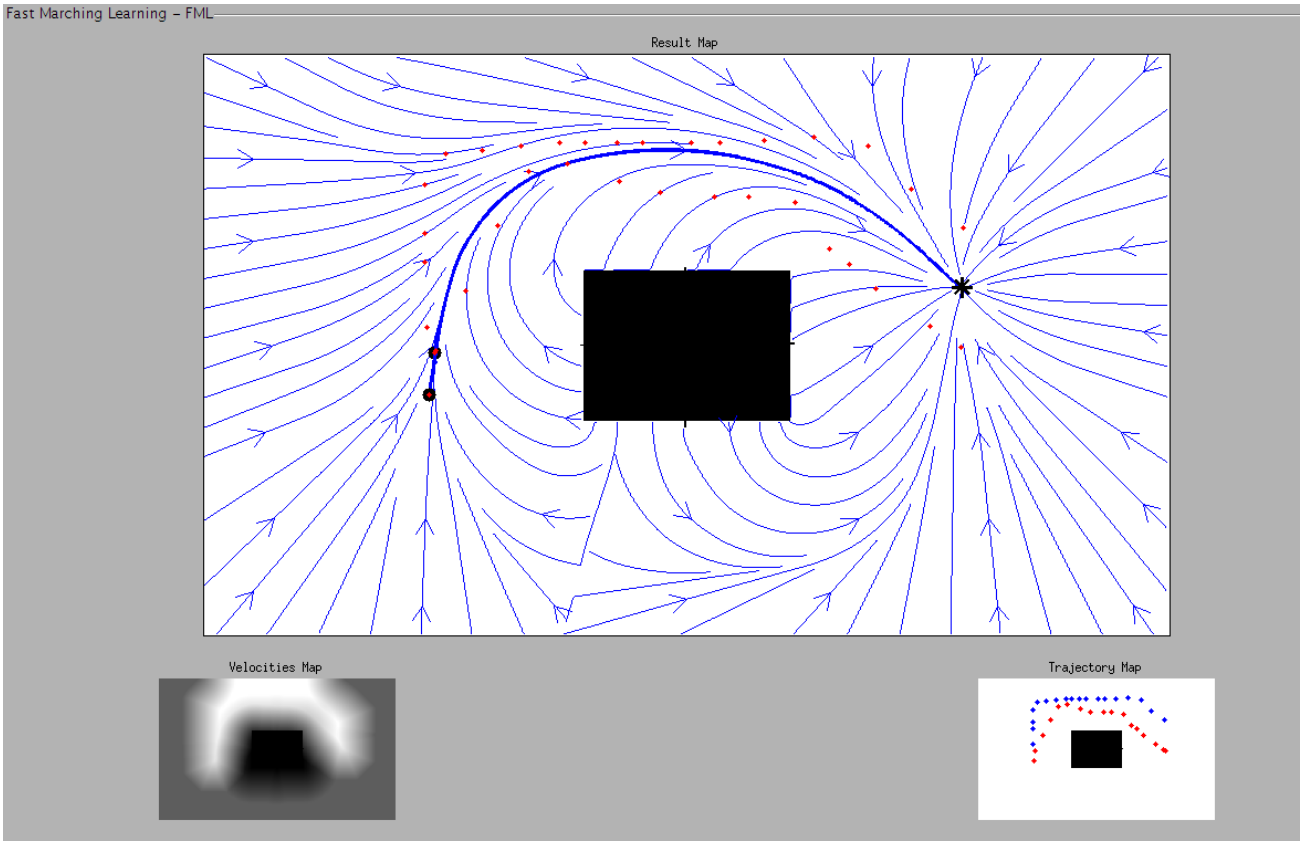
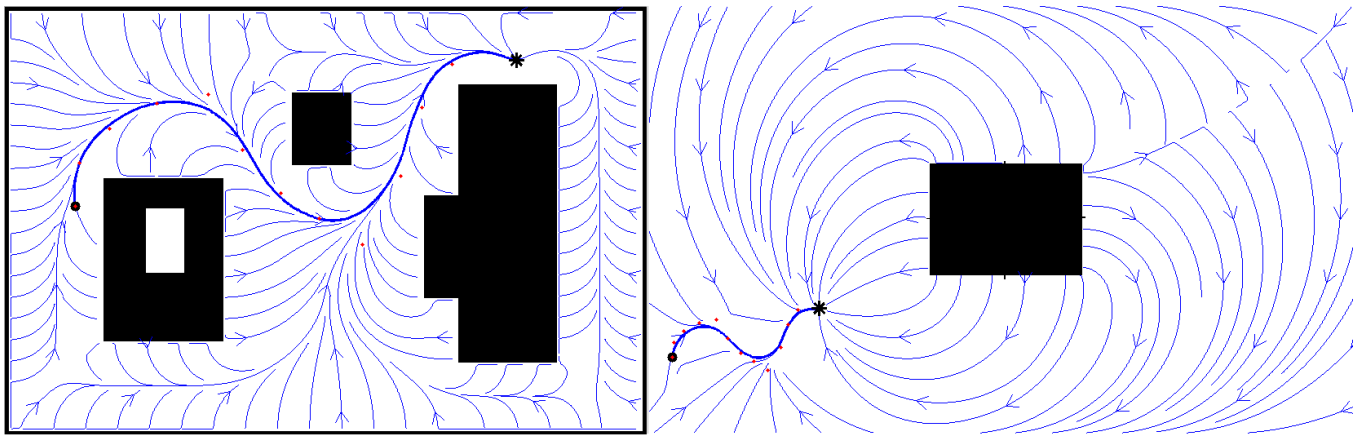


FIGURA 57.”*FML con sat=0.6 y AOI =65*”



(1) (2)

FIGURA 58."*FML cargar trayectoria*"

Demostración FM2F

En esta última demostración se analizan 2 configuraciones de parámetros con el mapa de ensayos. De esta manera se probara la diferencia que se generan.

Primera demostración. La configuración seguida en esta demostración es:

- Sat =1: condiciona la distancia de seguridad y también condicionara el resultado en la gráfica del eje fm2fdist, puesto que obliga a modificar la trayectoria de los seguidores para no chocarse y mantener la distancia de seguridad.
- Uncertainty=0.6 y 1: al disminuir el valor aumenta la incertidumbre alrededor de los seguidores, de manera que queda representada en el mapa de saturación, aumentando el área de incertidumbre de los seguidores (figura 59). En la figura 60, se reduce la incertidumbre al aumentar el valor, lo que hace más pequeños puntos representativos y las áreas de incertidumbre.
- Dist=10 y 11: esta distancia hace que la formación sea más compacta. Con una formación mayor, habría una formación más extensa, pero también tendrán los robots que modificar más la trayectoria en caso de llegar a un obstáculo.
- Steps = 100 y 10: este valor hace que se ejecute el algoritmo más rápido, requiere un cálculo menos preciso. En caso de disminuir el valor, aumenta el tiempo de calculo.

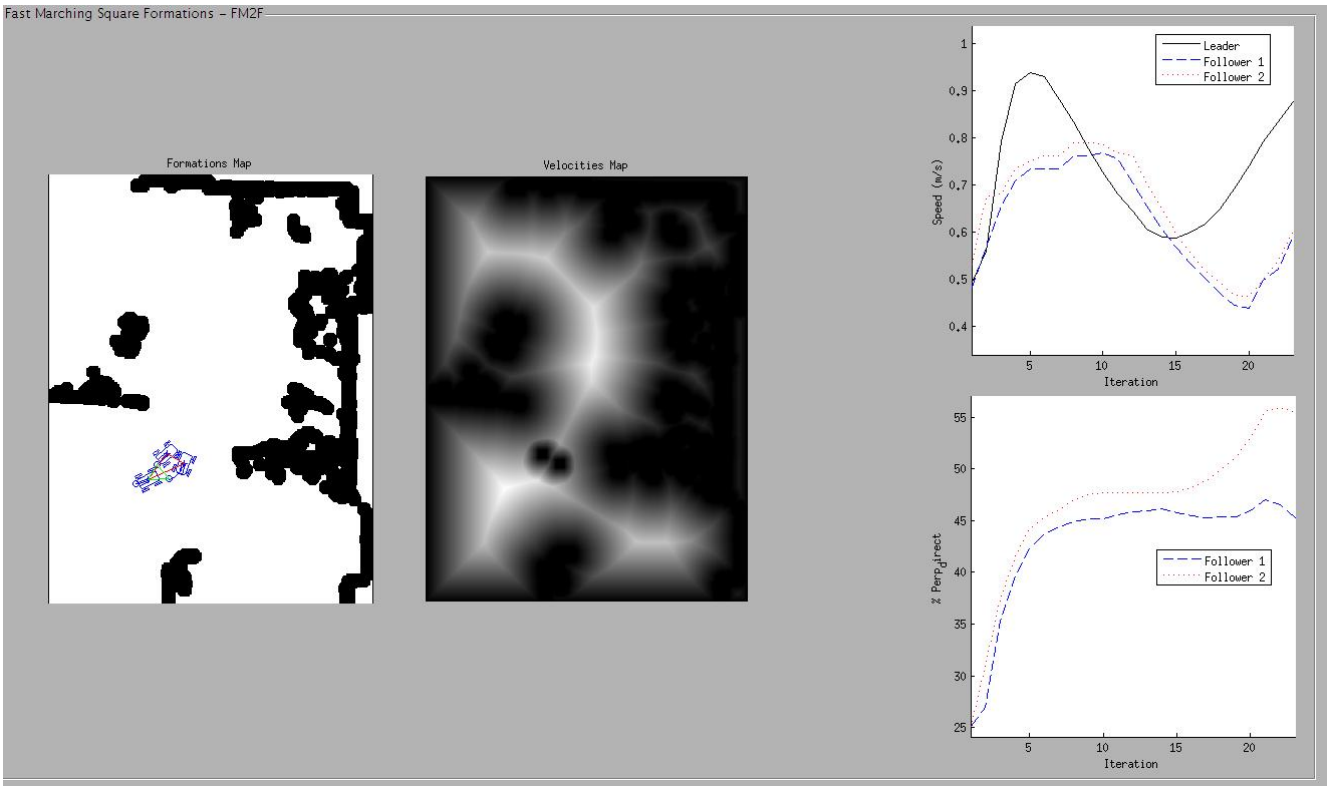


FIGURA 59. "*FM2F con sat=0.9 uncertainly=0.65, dist=10 y steps = 100*"

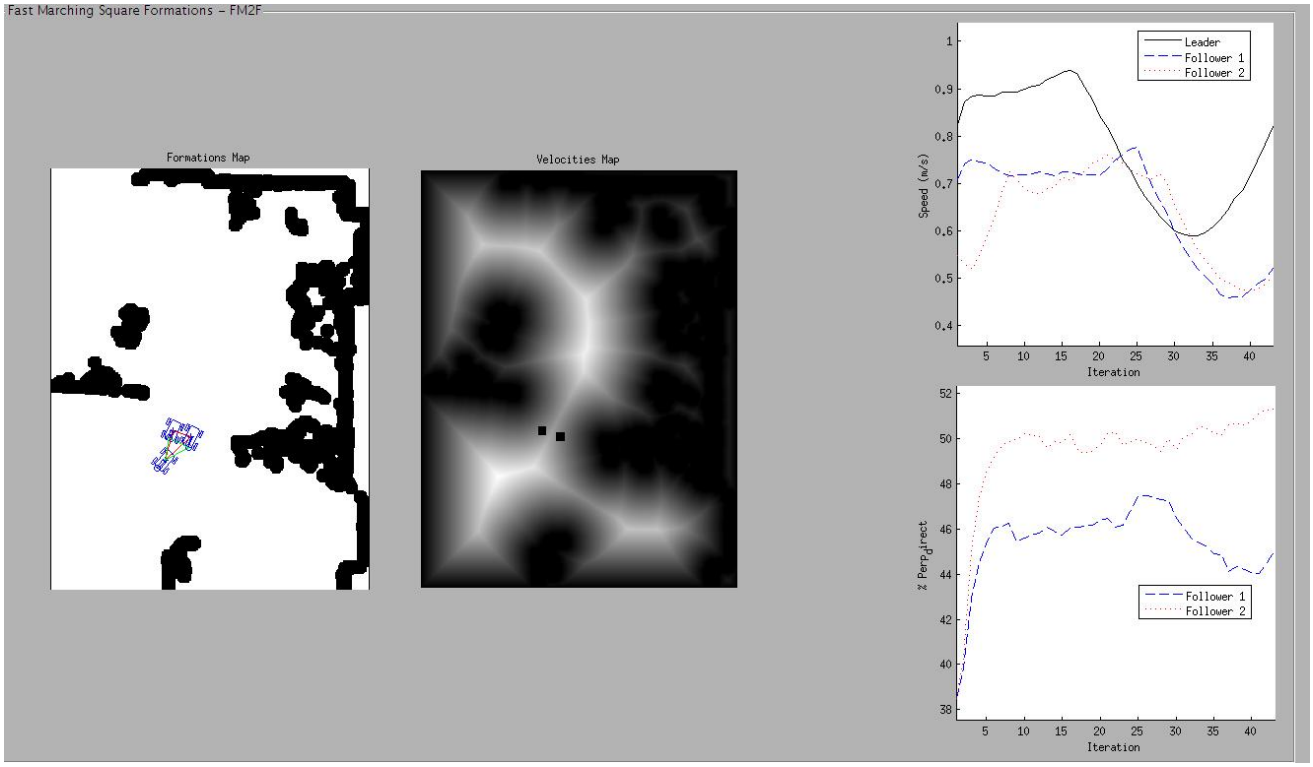


FIGURA 60."*FM2F con sat=0.9, uncertainly =1, dist= 11 y steps =10*"

Finalizado el proyecto y una vez que se han realizado las demostraciones, se puede afirmar que se ha conseguido integrar con éxito los algoritmos de planificación de trayectorias FM2, FML y FM2F.

Para conseguir esta integración, se ha necesitado adquirir competencias sobre la planificación de trayectorias y sobre los diferentes métodos para poder obtener una trayectoria. Además, se han adquirido competencias sobre los parámetros de saturación, área de influencia, incertidumbre, distancia de seguidores, etc.

Para hacer posible la integración, también se han adquirido competencias sobre el estudio y el diseño de interfaces gráficas con GUIDE, tales como los diferentes elementos y la manera de programar en el programa MATLAB.

Se ha conseguido crear una interfaz para poder diseñar los mapas necesarios para poner a prueba las demostraciones.

También se ha conseguido almacenar los espacios de trabajo de los tres algoritmos, poder guardar imágenes de los resultados de los tres y poder almacenar los puntos que conforman la trayectoria en el algoritmo FML

Por lo tanto, se puede afirmar que se han alcanzado los objetivos propuestos al principio de esta memoria.

Sección 6: Anexo

6.1 Presupuesto

En esta sección se analizan los costes y el valor del proyecto, separándolo en tres secciones:

- Coste de personal.
- Coste material.
- Total.

6.1.1 Costes de personal

Este apartado de costes estará dividido en: diferencia de salario entre el director de proyecto e ingenieros y las horas que se dedican al proyecto.

En la tabla 1 se dividirá el proyecto en 4 apartados:

- Planteamiento: Estudio y comprensión del algoritmo y diseño de interfaz gráficas utilizando la herramienta de MATLAB, GUIDE.
- Desarrollo del proyecto: desarrollo del algoritmo e interfaz, así como la investigación de nuevos elementos sobre la interfaz.
- Test: se realizan las pruebas necesarias del algoritmo, comprobando el funcionamiento correcto.
- Memoria: redacción de la memoria, incluyendo la posterior corrección.

Proyecto	Desglose (horas)	Director de proyecto	Ingeniero
PLANTEAMIENTO	Comprensión algoritmo	0	44
	Diseño herramienta	15	30
DESARROLLO	Desarrollo algoritmo	4	280
	Investigación	3	143
	Desarrollo interfaz	7	100
TEST	Prueba de algoritmos	10	65
MEMORIA	Redacción	0	120
	Corrección	6	25
TOTAL	Total horas		
	Salario/hora (€)	50	30
	Total individual (€)	2250	24210
	Total (€)		26460

TABLA 1. "*Costes de personal*"

6.1.2 Coste material

En esta sección del presupuesto se incluye todo el coste de material necesario para el desarrollo del proyecto. En este caso, como se refleja en la tabla x, se incluye el coste de hardware empleado para la realización, coste de software empleado en la realización del proyecto y coste oficina.

En este caso el portátil tiene un plazo de amortización de tres años y ha sido utilizado durante 7 meses en el proyecto. Para el coste del software se han empleado alternativas de carácter gratuito. El único coste de oficina será el internet, necesario para los procesos de planteamiento, desarrollo y test. En la tabla 2 se recogen los gastos materiales.

Tipo de Coste		Coste (€)	Coste de proyecto (€)
Hardware	Ordenador portátil	750	145.84
Software	Software	0	0
Oficina	Internet	29 €/mes	203
Total			348.84

TABLA 2."*Coste de material*"

6.1.3 Total

En este último apartado del anexo de presupuesto se sumaran los costes anteriormente mencionados junto con los costes de carácter indirecto (20%) y el IVA (21%). Se incluye un apartado subtotal donde no se incluye IVA. En la tabla 3 se recogen los datos mencionados.

Tipo de coste	Coste (€)
Personal	26460
Material	348.84
Indirecto (20%)	2361.77
Subtotal	29170.61
IVA (21%)	5629.86
Total	34800.47

TABLA 3."*Coste total*"

Sección 7: Referencias

[1] A. Valero, J.V. Gómez, S. Garrido and L. Moreno, **The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories**, IEEE Robotics and Automation Magazine, Vol. 20, No. 4, 2013. Impact factor (2012): 2.484, Q1

[2] J.V. Gómez, A. Lumbier, S. Garrido and L. Moreno, **Planning Robot Formations with Fast Marching Square including Uncertainty Conditions**, Robotics and Autonomous Systems, Vol. 61, No. 2, pp. 137–152, 2013. http://dx.doi.org/10.1016/j.bbr.2011.03.031 Impact factor (2012): 1.156, Q2

[3] A. Valero, J.V. Gómez, S. Garrido and L. Moreno **Deterministic, Globally Stable Motion Learning with Fast Marching Square**, Robotics and Autonomous Systems, 2013.

[4] Introducción a MATLAB GUIDE.
https://www.dspace.espol.edu.ec/bitstream/123456789/10740/11/MATLAB_GUIDE.pdf

[5] Robótica (ult. mod. 20 de junio 2014]. *En Wikipedia, la enciclopedia libre*. Disponible en: http://es.wikipedia.org/wiki/Robot.

[6] J.V.Gomez, *Consulta material Fast Marching*.
http://sgpsproject.sourceforge.net/JavierVGomez/index.php/Main_Page

[7] Principios de diseño de una interfaz gráfica
http://www.letrak.com.co/alejandro/material/web/Principios-Disenno-Delphi.pdf