Trabajo Fin de Máster

Implementación de técnicas de modelado, planificación y control para la coordinación de sistemas multi-robot en entornos complejos para la ejecución de tareas de atrapamiento

Implementation of modeling, planning and control techniques for the coordination of multi-robot systems in complex enviroments for the execution of entrapment tasks

Autor

Carlos Renau Morales

Directores

Eduardo Montijano Muñoz

Carlos Sagüés Blázquiz

Escuela de Ingeniería y Arquitectura

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

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

D./Dª. CARLOS RENAU MORALES,

con nº de DNI 20910851X en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) MÁSTER, (Título del Trabajo) IMPLEMENTACIÓN DE TÉCNICAS DE MODELADO, PLANIFICACIÓN Y CONTROL PARA LA COORDINACIÓN DE SISTEMAS MULTI-ROBOT EN ENTORNOS COMPLEJOS PARA LA EJECUCIÓN DE TAREAS DE ATRAPAMIENTO,

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

Zaragoza, 17 de Septiembre de 2018

Fdo: CARLOS RENAU MORALES

IMPLEMENTACIÓN DE TÉCNICAS DE MODELADO, PLANIFICACIÓN Y CONTROL PARA LA COORDINACIÓN DE SISTEMAS MULTI-ROBOT EN ENTORNOS COMPLEJOS PARA LA EJECUCIÓN DE TAREAS DE ATRAPAMIENTO

RESUMEN

La idea de disponer de un sistema multi-robot que se mueva por un entorno de forma coordinada plantea numerosas ventajas para los seres humanos, teniendo aplicación directa en tareas de exploración, de vigilancia o de búsqueda y rescate, en extinción de incendios, en operaciones militares o en la monitorización de la vida silvestre. En este contexto, es importante disponer de técnicas y algoritmos que permitan resolver los problemas de planificación, asignación de tareas y coordinación del movimiento propios de los sistemas multi-robot.

Con la finalidad de aportar soluciones a estos problemas, fundamentalmente en tareas de atrapamiento coordinado, este trabajo final de máster tiene como objetivo principal el desarrollo de estrategias de captura cooperativa de agentes evasores en entornos complejos utilizando un equipo de robots. Para ello, es necesario disponer, en primer lugar, de un modelo del entorno sobre el cual poder aplicar las estrategias de captura. Dado que en la captura pueden aparecer uno o varios evasores, y dado que el entorno puede ser convexo o no convexo y estar o no libre de obstáculos, se abren varias variantes a la hora de abordar este problema.

El proyecto parte de un artículo de investigación que aborda la captura cooperativa de evasores en entornos cerrados convexos libres de obstáculos, implementando con éxito y desde cero las técnicas propuestas en él. Dado que la premisa de que el entorno sea convexo lo aleja de su implementación en entornos reales, se extienden estas técnicas de captura a entornos complejos y se desarrollan nuevas estrategias de atrapamiento.

El correspondiente estudio del estado del arte permite abordar el modelado de entornos complejos, desarrollando e implementando por cuenta propia una estrategia de modelado orientada a tareas de atrapamiento y que se sustenta principalmente en la teoría de grafos. Las estrategias de captura en entornos cerrados complejos son elaboradas de acuerdo al modelado proyectado en este trabajo, y son implementadas y evaluadas inicialmente frente a un único evasor. Dichas estrategias también se extienden al caso de múltiples evasores, logrando con éxito la captura en un tiempo finito.

Con la finalidad de evaluar las diferentes estrategias de captura propuestas, se realiza un ensayo estadístico de las mismas frente a una lógica de evasión manual y una programada. Por último, se realiza una prueba de concepto con un equipo real de robots.

Índice general

1. INTRODUCCIÓN 15

1.1. Motivación **15**

1.2. Objetivos y problemas abordados **16**

1.3. Alcance **17**

1.4. Metodología y herramientas aplicadas **17**

1.5. Estructura de la memoria **18**

2. CAPTURA COOPERATIVA DE MÚLTIPLES EVASORES EN ENTORNOS CERRADOS CONVEXOS LIBRES DE OBSTÁCULOS 19

2.1. Formulación del problema **19**

2.2. Minimización del área segura de un único evasor **20**

2.3. Minimización del área segura de múltiples evasores **20**

2.4. Diagrama de Voronoi en entornos cerrados convexos **20**

2.5. Implementación **21**

2.6. Resultados **21**

2.7. Conclusiones **22**

3. MODELADO DE ENTORNOS COMPLEJOS 23

3.1. Elaboración del grafo de trayectorias **24**
3.1.1. Procesado del mapa binario 24
3.1.2. Generación de trayectorias libres de colisiones 25

3.2. Elaboración del grafo regiones **26**
3.2.1. Procesado del mapa binario 26
3.2.2. Generación de regiones convexas 26

3.3. Vinculación de grafos **27**

4. CAPTURA COOPERATIVA DE UN EVASOR EN ENTORNOS CERRADOS COMPLEJOS 29

4.1. Formulación del problema 29

4.2. Localización de agentes en el grafo 30

4.3. Asignación de rutas 30

4.4. Estrategia de atrapamiento 31
4.4.1. Atrapamiento directo 32
4.4.2. Atrapamiento predictivo 32
4.4.3. Atrapamiento por minimización del área segura 32

4.5. Resultados y conclusiones 33

5. CAPTURA COOPERATIVA DE MÚLTIPLES EVASORES EN ENTORNOS CERRADOS COMPLEJOS 35

5.1. Formulación del problema 35

5.2. Asignación de objetivos 35
5.2.1. Formulación del problema 36
5.2.2. Resolución del problema 36
5.2.3. Actualización de objetivos 37

5.3. Resultados y conclusiones 37

6. EXPERIMENTACIÓN 39

6.1. Lógica de evasión programada 39

6.2. Ensayo estadístico de las estrategias de atrapamiento 40
6.2.1. Ensayo automatizado 41
6.2.2. Ensayo manual 41
6.2.3. Coste computacional 42
6.2.4. Conclusiones 42

6.3. Implementación en ROS 43
6.3.1. Introducción a ROS 43
6.3.2. Implementación en ROS de los algoritmos de captura 43
6.3.3. Resultados y conclusiones 44

7. CONCLUSIONES Y MÁRGENES DE AMPLIACIÓN 47

7.1. Trabajo realizado 47

7.2. Trabajo futuro 48

8. REFERENCIAS 49

ANEXOS

ANEXO I – ESTADO DEL ARTE 53

1. Introducción **53**

2. Pursuit-evasion problems **54**

3. Modelado del entorno para la generación de trayectorias **55**

4. Algoritmos de búsqueda **55**

5. Asignación de tareas **56**

ANEXO II – HERRAMIENTAS APLICADAS 57

1. Diagramas de Voronoi **57**

2. Grafo euclídeo no dirigido y matriz de adyacencia **58**

3. Triangulación de Delaunay **59**

4. Algoritmo de Dijkstra **60**

5. Programación lineal: Método Simplex **60**

ANEXO III – DIAGRAMA DE VORONOI EN ENTORNOS CERRADOS CONVEXOS 61

1. Función voronoi_boundary_2D **62**

2. Función intersect_2D **63**

ANEXO IV – ELABORACIÓN DEL GRAFO DE TRAYECTORIAS 65

1. Procesado del mapa binario **65**

2. Eliminación de rectas no válidas **66**

3. Filtrado de rectas inconexas **66**

4. Fusionado de rectas **68**

ANEXO V – ELABORACIÓN DEL GRAFO DE REGIONES 69

1. Resultados **69**

ANEXO VI – FUNCIÓN VORONOI_2P 71

Índice de Imágenes

Imagen 1 – Centro logístico de Amazon robotizado. 15
Imagen 2 - Espacio de parámetros para modelos de búsqueda autónoma. 16
Imagen 3 - Diagrama de Voronoi de 8 puntos en un espacio 2D. 19
Imagen 4 - Estrategia de minimización del área segura en un entorno convexo con 3 perseguidores y un evasor. 20
Imagen 5 - Argumentos de entrada y salida de la función 'voronoi_bounday_2D'. 20
Imagen 6 - Captura cooperativa de un evasor en un entorno cerrado convexo. 21
Imagen 7 - Captura cooperativa de múltiples evasores (representados con cruces) en un entorno cerrado convexo. 22
Imagen 8 - Generación manual del mapa binario de una vivienda a partir de su vista en planta. 23
Imagen 9 - Condición de Delaunay. 23
Imagen 10 - Triangulación de Delaunay de 10 puntos. 23
Imagen 11 – Ejemplo de procesado del mapa binario de entrada. 24
Imagen 12 - Lógica de detección de puntos esquina exterior, puntos esquina interior y puntos línea borde. 24
Imagen 13 - Efecto de la distancia de seguridad sobre las trayectorias generadas. 25
Imagen 14 - Efecto de eliminar las rectas inconexas de acuerdo a un parámetro de filtrado. 25
Imagen 15 – Izqda.: Diagrama de Voronoi en bruto. Dcha.: Diagrama de Voronoi post-procesado. 26
Imagen 16 - Izqda.: Procesado del mapa binario. Dcha.: Post-procesado de regiones convexas. 27
Imagen 17 - Vinculación de grafos. A cada nodo región (x) se le asocia el nodo trayectoria (*) visible más cercano. 27
Imagen 18 - Localización de agentes en el grafo. 30
Imagen 19 - Pseudocódigo del código de asignación de rutas. 30
Imagen 20 - Asignación de rutas por peaje. 31
Imagen 21 - Definición de una región convexa en un entorno complejo no convexo. 32
Imagen 22 - Extensión de la estrategia de minimización del área segura a entornos complejos. 33
Imagen 23 - Captura cooperativa de un evasor en un entorno complejo bajo la estrategia de atrapamiento predictivo. 34
Imagen 24 – Ineficacia en entornos complejos de la estrategia de minimización del área segura. 34
Imagen 25 - Asignación de objetivos. 37
Imagen 26 - Captura cooperativa de múltiples evasores (representados como *) en un entorno complejo. 38
Imagen 27 – Simulación de captura con 3 perseguidores y 1 evasor automatizados. 39
Imagen 28 - Entornos de ensayo de algoritmos. Izqda.: Entorno1. Dcha.: Entorno2. 40
Imagen 29 - Resultados del ensayo automatizado en el Entorno1. 41
Imagen 30 - Resultados del ensayo automatizado en el Entorno2. 41
Imagen 31 - Resultados del ensayo manual en el Entorno1. 41
Imagen 32 - Resultados del ensayo manual en el Entorno2. 42

Imagen 33 - Coste computacional de las estrategias de atrapamiento. 42
Imagen 34 - Esquema de funcionamiento de ROS..... 43
Imagen 35 – Arquitectura de la implementación realizada en ROS, mostrada para un único robot. 44
Imagen 36 - Visualización de resultados con Rviz de la captura de 1 robot evasor (representado en verde) realizada en un entorno real con un equipo real de robots. 44
Imagen 37 - Modelado en forma de grafos del entorno sobre el que se ha realizado el ensayo con robots reales. 45
Imagen 38 - Robots empleados en el experimento..... 45
Imagen 39 - Espacio de parámetros para modelos de búsqueda autónoma. 54
Imagen 40 - Grafo de visibilidad..... 55
Imagen 41 - Diagrama de Voronoi..... 55
Imagen 42 - Diagrama de Voronoi aplicado a 22 jugadores sobre un campo de fútbol. 57
Imagen 43 - Grafo en el que los nodos representan ciudades y las aristas la distancia en km entre nodos. 58
Imagen 44 - Izqda.: Grafo dirigido. Dcha.: Grafo no dirigido. 58
Imagen 45 - Matriz de adyacencia asociada a un grafo no dirigido. 59
Imagen 46 - Triangulación de Delaunay de 10 puntos. 59
Imagen 47 - Argumentos de salida de la función ‘voronoi’ que incorpora Matlab R2015. 61
Imagen 48 - Argumentos de salida de la función 'voronoi_boundary_2D'. 61
Imagen 49 – Secuencia lógica de obtención del diagrama Voronoi delimitado para distintos entornos cerrados. 62
Imagen 50 – Ejemplos de procesado del mapa binario de entrada..... 65
Imagen 51 - Efecto de la distancia de seguridad sobre las trayectorias generadas. 66
Imagen 52 - Post-procesado final del diagrama de Voronoi en entornos irregulares..... 67
Imagen 53 - Post-procesado final del diagrama de Voronoi en entornos irregulares sin rectas inconexas..... 67
Imagen 54 - Efecto de eliminar todas las rectas inconexas en un entorno con caminos abiertos. 67
Imagen 55 - Filtrado ajustable de rectas inconexas en un entorno con caminos abiertos. 68
Imagen 56 - Post-procesado final del diagrama de Voronoi en un entorno cuadrado (izqda.) y en uno circular (dcha.). 68
Imagen 57 – Grafo de regiones convexas en entornos de geometría mayormente rectangular. . 69
Imagen 58 – Grafo de regiones convexas en un entorno marcadamente circular. 69
Imagen 59 - Grafo de regiones convexas en entornos de geometría irregular. 70

1. Introducción

1.1. Motivación

Imaginemos por un momento un futuro en el que no es necesario poner vidas humanas en juego a la hora de llevar a cabo actividades tan necesarias en la sociedad como son las operaciones militares, la extinción de incendios o la lucha contra el crimen. Esta idea puede llegar a ser una realidad gracias al desarrollo de los sistemas multi-robot, sistemas formados por múltiples robots móviles autónomos, capaces de interactuar con su entorno y con un objetivo en común. Un ejemplo muy claro de sistema multi-robot es el formado por los robots logísticos de Amazon que trabajan en conjunto y de forma autónoma con el objetivo de agilizar el abastecimiento de pedidos (Imagen 1).

Imagen 1 – Centro logístico de Amazon robotizado.

Los sistemas multi-robot presentan mayor robustez, flexibilidad y capacidad de trabajo que los sistemas mono-robot. No obstante, estas ventajas sólo se ponen de manifiesto cuando se implementan mecanismos de cooperación apropiados que resuelven los tres problemas básicos de los sistemas multi-robot: la planificación de tareas, la asignación de tareas y la coordinación del movimiento.

Este trabajo final de máster nace con la motivación de aportar nuevas técnicas y algoritmos que permitan resolver, sin una complejidad excesiva, los problemas derivados de trabajar con un sistema multi-robot en entornos reales. La temática en torno a la cual se desarrolla la solución es el atrapamiento coordinado de agentes hostiles, pues este tipo de algoritmos tienen aplicación directa en los ejemplos mencionados anteriormente.

Los problemas de persecución-evasión[1], dada su gran utilidad para estudiar problemas de planificación del movimiento, han sido analizados en versiones muy diversas (Imagen 2) y desde dos perspectivas muy distintas: la conservadora y la probabilística. Además, la resolución de estos problemas se puede plantear desde dos enfoques distintos, bien mediante una descripción con ecuaciones diferenciales o bien usando técnicas combinacionales, es decir, hacer una representación del entorno y usar esta representación para la resolución del problema.

[1] Presentados en más detalle en el anexo I – cap. 2.

En este proyecto, el atrapamiento coordinado en entornos complejos es abordado asumiendo conocimientos sobre el comportamiento de los evasores y mediante un enfoque combinacional, pues esto permite tratamientos más sencillos con un menor coste computacional. La solución aportada hace uso de técnicas de modelado de entornos enfocadas a la generación de un mapa de trayectorias, de algoritmos de búsqueda para asignar caminos de mínima distancia entre dos puntos del mapa, y de métodos de programación lineal para abordar el problema de la asignación de tareas. [2]

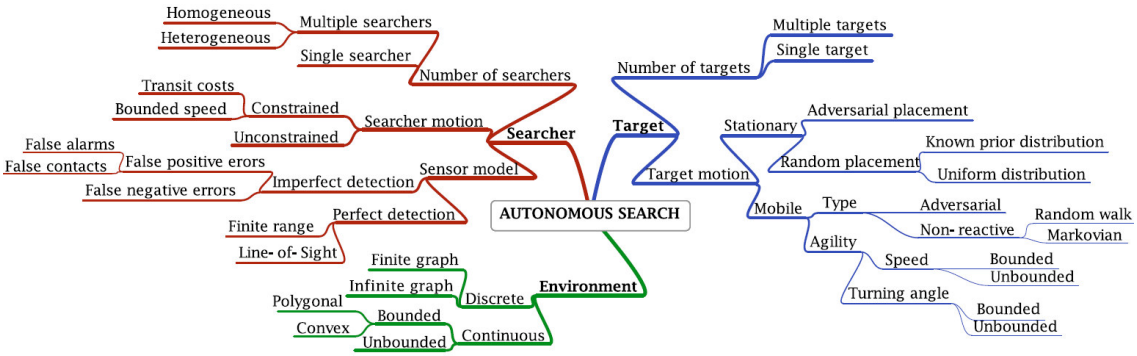


Imagen 2 - Espacio de parámetros para modelos de búsqueda autónoma.

1.2. Objetivos y problemas abordados

El objetivo principal del proyecto es el desarrollo de estrategias de captura cooperativa de agentes evasores en entornos complejos utilizando un equipo de robots.

Para ello, es necesario disponer, en primer lugar, de un modelo del entorno sobre el cual aplicar posteriormente las estrategias de captura. Dado que en la captura pueden aparecer uno o varios evasores, y dado que el entorno puede ser convexo o no convexo y estar o no libre de obstáculos, se abren varias variantes a la hora de abordar este problema.

En particular, en este trabajo se plantean los siguientes objetivos específicos:

- Implementación de un algoritmo para la captura cooperativa de un evasor en entornos cerrados convexos libres de obstáculos.
- Implementación de un algoritmo para la captura cooperativa de múltiples evasores en entornos cerrados convexos libres de obstáculos.
- Modelado de entornos cerrados complejos, es decir, entornos convexos y no convexos con obstáculos de geometría regular e irregular.
- Desarrollo e implementación de estrategias para la captura cooperativa de un evasor en entornos cerrados complejos.
- Desarrollo e implementación de estrategias para la captura cooperativa de múltiples evasores en entornos cerrados complejos.
- Ensayo estadístico de las estrategias de captura propuestas.
- Realización de una prueba de concepto con un equipo real de robots.

[2] En el anexo I – caps. 3-5 se lleva a cabo un breve estudio del estado del arte de estas materias.

1.3. Alcance

Este proyecto ha partido del artículo de investigación [1] que aborda la captura cooperativa de evasores en entornos cerrados convexos libres de obstáculos. Se han implementado, en Matlab y desde cero, las técnicas propuestas en el artículo, logrando con éxito la captura cooperativa de múltiples evasores. Para poder aplicar estas técnicas, ha sido necesario crear una función para delimitar la teselación de Voronoi [7] a una región convexa cerrada. También se extienden las técnicas del artículo a entornos complejos y se desarrollan nuevas estrategias de atrapamiento.

El correspondiente estudio del estado del arte ha permitido abordar el modelado de entornos complejos, desarrollando e implementando en Matlab por cuenta propia una estrategia de modelado que se sustenta en formalismos matemáticos ya conocidos e implementados.

Las estrategias de captura en entornos cerrados complejos han sido elaboradas de acuerdo al modelado proyectado en este trabajo, y han sido implementadas en Matlab y evaluadas frente a un único evasor. Dichas estrategias también se han extendido al caso de múltiples evasores, logrando con éxito y de forma efectiva la captura coordinada de todos ellos en un tiempo finito.

Con la finalidad de evaluar las diferentes estrategias de captura propuestas, se ha realizado un ensayo estadístico de las mismas tanto de forma manual como automatizada. Para este último caso, ha sido necesario idear también una estrategia de evasión programada. Por último, se ha realizado una prueba de concepto con un equipo real de robots.

1.4. Metodología y herramientas aplicadas

Los problemas que plantea este proyecto son tratados de manera incremental, estudiando inicialmente una versión simplificada del problema en la que la captura ocurre en entornos convexos sin obstáculos. Posteriormente se trata la versión compleja en la que son necesarias técnicas de modelado del entorno y algoritmos de planificación y control más avanzados.

De los diversos formalismos matemáticos que existen en el modelado de entornos, se ha optado por una modelización en forma de grafo [6] dada su sencillez computacional y la posibilidad de utilizar la teoría de grafos en las tareas de atrapamiento.

Por tanto, para cada entorno se definen dos grafos: uno que constituye un mapa de trayectorias libre de colisiones, y otro que constituye un conjunto de regiones convexas. El primero se obtiene de la teselación de Voronoi [7], pues de esta forma se maximiza la distancia entre trayectorias y obstáculos. El segundo se obtiene de la triangulación de Delaunay [8] por su sencillez y eficacia para entornos de geometría muy variada.

En cuanto a la planificación de la captura, ésta se computa sobre el grafo de trayectorias bajo un enfoque discreto, haciendo uso de la teoría de grafos. Para obtener el camino de mínima distancia entre un nodo origen y un nodo destino se emplea el algoritmo de Dijkstra [9], mientras que la asignación de objetivos en capturas con múltiples evasores se resuelve con el método Simplex [10]. Por último, en la asignación de trayectorias, con el fin de cubrir las vías de escape del evasor, se ha diseñado un algoritmo denominado en este proyecto como método del peaje.

Los algoritmos propuestos en este documento han sido evaluados en Matlab en su fase de desarrollo. Posteriormente, en una segunda fase se ha propuesto una implementación más realista sobre robots reales haciendo uso del entorno de trabajo ROS [11].

1.5. Estructura de la memoria

En el capítulo 1 se ha realizado una introducción al proyecto, justificando su interés, especificando los objetivos que persigue, los objetivos finalmente alcanzados y la metodología empleada para ello.

En el capítulo 2 se aborda la captura cooperativa de evasores en entornos cerrados convexos libres de obstáculos, presentando las técnicas de captura descritas en el artículo de investigación [1], su implementación para el caso de uno y de múltiples evasores, y los resultados obtenidos en cada caso.

En el capítulo 3 se aborda el problema de modelado de entornos complejos, desarrollando una estrategia de modelado sustentada en formalismos matemáticos muy usados en este campo.

El capítulo 4 desarrolla la captura cooperativa de un único evasor en entornos cerrados complejos, abordando problemas como la localización de agentes en el grafo, la asignación de rutas y la estrategia de atrapamiento. También lleva a cabo una evaluación cualitativa de los resultados.

En el capítulo 5 se extiende la estrategia de captura anterior al caso de múltiples evasores, abordando para ello el problema de la asignación de objetivos.

El capítulo 6 recoge, por un lado, los resultados obtenidos en el ensayo estadístico de las diferentes estrategias de atrapamiento, realizado para entornos y condiciones diferentes y bajo una lógica de evasión tanto programada como manual. En segundo lugar, y a modo de prueba de concepto, presenta la implementación llevada a cabo con robots reales.

Por último, en el capítulo 7 se exponen las conclusiones extraídas de la realización de este trabajo, presentando las dificultades encontradas en su desarrollo, así como las posibles líneas de trabajo futuras.

Se complementa la información desarrollada en esta memoria con seis anexos. El anexo I recoge brevemente el estado del arte de las materias involucradas en la captura cooperativa en entornos complejos. El anexo II presenta los formalismos y herramientas aplicadas en el desarrollo del trabajo. El anexo III presenta la función elaborada para delimitar los diagramas de Voronoi a una región convexa cerrada, necesaria para la implementación de las técnicas descritas en el artículo de investigación [1]. Los anexos IV y V desarrollan en más detalle las técnicas usadas en este proyecto para el modelado de entornos complejos. Finalmente, el anexo VI presenta la función elaborada para obtener el diagrama de Voronoi de dos puntos en un entorno cerrado convexo, usada en los capítulos 4 y 5.

2. Captura cooperativa de múltiples evasores en entornos cerrados convexos libres de obstáculos

El proyecto comienza con la implementación y simulación, en Matlab, de la estrategia de captura propuesta en el artículo de investigación [1], en su versión centralizada en un espacio 2D, Esta estrategia asegura la captura cooperativa de múltiples evasores usando múltiples perseguidores en un entorno cerrado convexo libre de obstáculos. Los perseguidores desconocen la estrategia de evasión de sus rivales, pero pueden garantizar la captura siguiendo una estrategia de minimización del área segura de los evasores, la cual hace uso de los diagramas de Voronoi (Imagen 3). Estos diagramas hacen una descomposición de un espacio métrico en regiones, asociada a la presencia de objetos, de tal forma que a cada objeto se le asigna como región el espacio métrico más cercano a él que a ningún otro objeto.

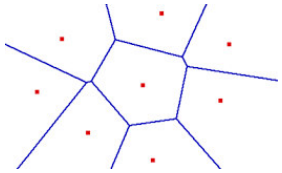


Imagen 3 - Diagrama de Voronoi de 8 puntos en un espacio 2D.

2.1. Formulación del problema

Sea un grupo de $n_p \in \mathbb{N}$ agentes perseguidores y $n_e \in \mathbb{N}$ agentes evasores, en un entorno cerrado convexo $Q \subset \mathbb{R}^2$, denotando los puntos contenidos en Q como q. Sea $x_p^i \in Q$ con $i \in \{1, \dots, n_p\}$ las posiciones de los perseguidores, sea $x_e^j \in Q$ con $j \in \{1, \dots, n_e\}$ las posiciones de los evasores, y sean u_p y u_e las acciones de control que mueven a los agentes por el entorno sujetas a la misma velocidad máxima $v_{máx}$:

$$\begin{cases} \dot{x}_p^i = u_p^i, & \left|\left|u_p(t)\right|\right| \leq v_{máx} \\ \dot{x}_e^j = u_e^j, & ||u_e(t)|| \leq v_{máx} \end{cases}$$

El objetivo de los perseguidores es capturar a todos los evasores. Se considera que un evasor j es capturado en un tiempo t_c cuando, siendo $r_c > 0$ el radio de captura, se cumple:

$$\min_{i \in n_p} ||x_e^j(t_c) - x_p^i(t_c)|| < r_c$$

Se define el área segura de un evasor, A_j, como el conjunto de puntos al que dicho agente j puede llegar antes que cualquier otro agente. Para agentes con velocidades máximas iguales, esta definición se corresponde con la definición de la teselación de Voronoi:

$$V_j = \{q \in Q \mid \ ||q - x^j|| \leq ||q - x^i||, \ \forall i\}$$

$$A_j = \int_{V_j} dq$$

2.2. Minimización del área segura de un único evasor

Se demuestra en [2] que, dirigiendo a los perseguidores al centroide de la frontera Voronoi compartida con el evasor, C_{bi}, se consigue una minimización del área segura de este último, garantizando así en última instancia la captura del evasor en un tiempo finito.

$$u_p^i = \frac{C_{bi} - x_p^i}{\left\| C_{bi} - x_p^i \right\|}$$

Para un espacio $\mathbb{R}^2$, C_{bi} se corresponde con el punto medio de la recta Voronoi compartida entre perseguidor y evasor. En la Imagen 4 se muestra ejemplificada esta estrategia de minimización del área segura en un entorno convexo con tres perseguidores y un evasor.

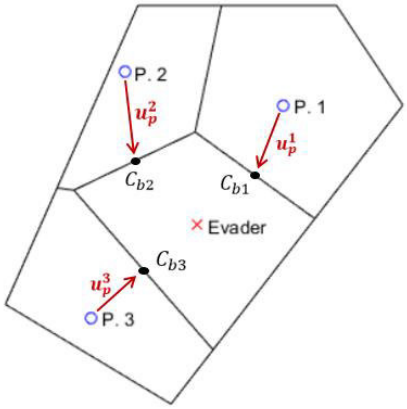


Imagen 4 - Estrategia de minimización del área segura en un entorno convexo con 3 perseguidores y un evasor.

2.3. Minimización del área segura de múltiples evasores

El principio de minimización del área segura empleado para el caso de un único evasor puede extenderse a múltiples evasores. No obstante, es importante observar que no es posible minimizar simultáneamente el área segura de todos los evasores, sino que es necesario incluir una fase de asignación de objetivos bajo un criterio de cercanía, de modo que los perseguidores lleven a cabo una minimización del área segura de su evasor más cercano.

De esta forma la captura cooperativa de múltiples evasores no es más que un tratamiento en paralelo de capturas cooperativas de un único evasor.

2.4. Diagrama de Voronoi en entornos cerrados convexos

A la hora de implementar esta estrategia en Matlab, ha sido necesario elaborar una función (Imagen 5) para delimitar el diagrama de Voronoi a un entorno cerrado convexo, pues la función nativa de Matlab que calcula la partición de Voronoi no permite considerar un entorno delimitante.

```
function [points_in, vecinos, regiones] = voronoi_boundary_2D(points, bound)
```

Imagen 5 - Argumentos de entrada y salida de la función 'voronoi_bounday_2D'.

La función creada en este proyecto recibe como argumentos de entrada las coordenadas de los puntos para los que se desea trazar el diagrama y de los puntos que definen el contorno cerrado. Como argumentos de salida devuelve los puntos para los que se computa Voronoi, la matriz de adyacencia, y una matriz multidimensional columna en la que cada fila constituye una matriz que almacena las rectas Voronoi que delimitan a un punto, así como con qué vecino comparten cada frontera.

Esta función procesa los argumentos de entrada para asegurar, por un lado, que el contorno sea convexo y, por otro lado, que los puntos introducidos estén contenidos en dicho entorno convexo. A continuación, computa con la función *voronoi* de Matlab el diagrama Voronoi asociado a dichos puntos, a partir del cual se elaborará el diagrama Voronoi delimitado. El algoritmo delimitante es bastante complejo y se ha desarrollado en varias fases. Una explicación más detallada de la función queda recogida en el Anexo III.

2.5. Implementación

La implementación de la estrategia de captura de minimización del área segura consiste en materializar los siguientes bloques lógicos:

1. Obtención del diagrama Voronoi delimitado asociado a los n agentes contenidos en Q.
2. Asignación de objetivos colindantes bajo un criterio de distancia mínima.
3. Cálculo de la dirección de avance según la estrategia de minimización del área segura.

2.6. Resultados

La Imagen 6 muestra los resultados obtenidos en simulación de una captura cooperativa de un único evasor en un entorno cuadrado convexo siguiendo la estrategia de minimización del área segura. Se comprueba la correcta implementación del algoritmo, garantizando la captura del evasor en un tiempo finito indistintamente de la situación inicial de los agentes y del tipo de geometría convexa del entorno.

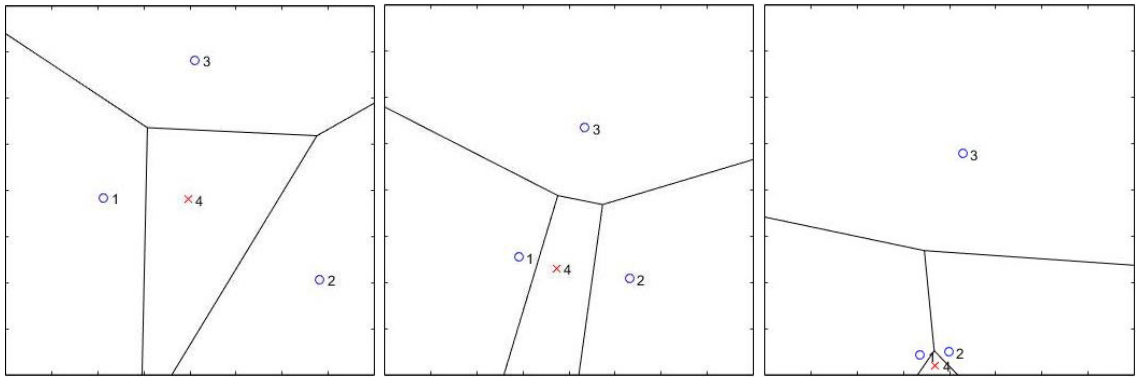


Imagen 6 - Captura cooperativa de un evasor en un entorno cerrado convexo.

Resultados igual de satisfactorios se obtienen al extender la estrategia al caso de múltiples evasores. La Imagen 7 muestra el transcurso de una captura en un entorno convexo cuadrado con $n_p = 5$ y $n_e = 4$, estando los evasores representados con cruces.

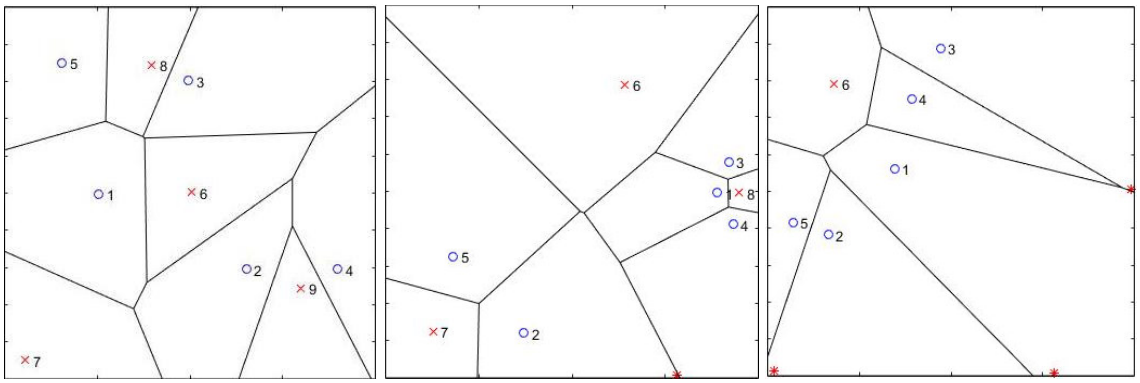


Imagen 7 - Captura cooperativa de múltiples evasores (representados con cruces) en un entorno cerrado convexo.

2.7. Conclusiones

La estrategia de minimización del área segura asegura matemáticamente la captura de todos los evasores en un tiempo finito. No obstante, la premisa de que el entorno sea convexo lo aleja de su implementación en entornos reales. A continuación, este proyecto se propone extender esta estrategia a entornos complejos, así como desarrollar nuevas estrategias de atrapamiento.

3. Modelado de entornos complejos

Para poder abordar la captura en entornos complejos, es necesario obtener en primer lugar un modelo del entorno. El proceso de obtención de este modelo va a depender de la información que se tenga sobre el entorno.

En este proyecto se supone como dato de entrada una matriz binaria que distingue zonas transitables y no transitables (Imagen 8). El tamaño de esta matriz depende de la resolución espacial que se desea asociar a un bit. La obtención del mapa binario puede ser manual, p. ej. a partir del plano planta del entorno, o automática, p. ej. mediante un equipo de cámaras aéreas.

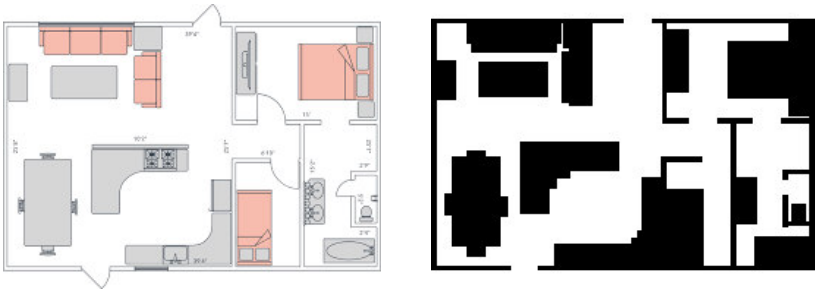


Imagen 8 - Generación manual del mapa binario de una vivienda a partir de su vista en planta.

Se ha optado por un modelado en forma de grafo pues ello permite emplear la teoría de grafos en tareas de captura. Este modelado debe permitir, por un lado, planificar trayectorias libres de colisiones y, por otro, dividir el entorno en regiones convexas para poder extender así la estrategia de minimización del área segura. Por tanto, el modelado se compone de dos grafos, uno de trayectorias y otro de regiones convexas:

- Para la elaboración de trayectorias existen diversas técnicas, siendo las más destacadas el uso de cuadrículas, los grafos de visibilidad y el diagrama de Voronoi. De entre estas, se ha escogido la última porque maximiza la distancia entre trayectorias y obstáculos, además es una herramienta con la que ya se ha trabajado en la fase anterior del proyecto.
- Para la generación de regiones convexas, la técnica empleada debido a su sencillez computacional y su eficacia frente a geometrías muy dispares ha sido la triangulación de Delaunay.

La triangulación de Delaunay (Imagen 10) genera, para un conjunto dado de vértices, una red de triángulos conexa y convexa que cumple la condición de Delaunay (Imagen 9): la circunferencia circunscrita de cada triangulo no contiene ningún otro vértice de otro triángulo.

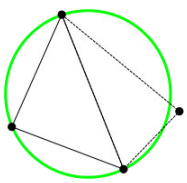


Imagen 9 - Condición de Delaunay.

Imagen 10 - Triangulación de Delaunay de 10 puntos.

En el Anexo II, capítulos 1, 2 y 3, se introducen con más detalle los formalismos matemáticos que se emplean en este proyecto para el modelado de entornos complejos.

3.1. Elaboración del grafo de trayectorias

Este grafo recoge las trayectorias que los robots deben seguir para desplazarse por el entorno sin colisionar con este. Los nodos del grafo están constituidos por los vértices de las rectas que definen las trayectorias, mientras que las aristas representan la unión entre dos nodos conectados por una recta, con un peso igual a su longitud.

3.1.1. Procesado del mapa binario

Dado que las trayectorias se pretenden generar mediante el diagrama de Voronoi, es necesario representar los obstáculos únicamente mediante puntos contorno (Imagen 11), de modo que, situando estos puntos contorno estratégicamente, la teselación de Voronoi permita extraer un conjunto de trayectorias que recorran el entorno sorteando los obstáculos.

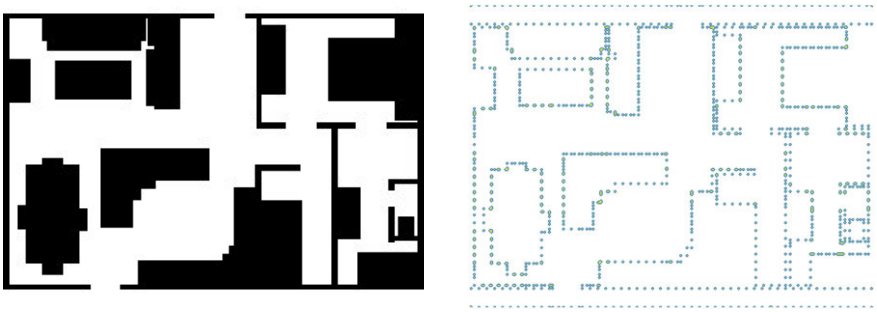


Imagen 11 – Ejemplo de procesado del mapa binario de entrada.

El primer paso consiste en la extracción de los puntos contorno de los obstáculos mediante el análisis de los valores adyacentes a cada bit de la matriz binaria. Como se observa en la Imagen 12, 1 punto libre representa un punto esquina interior, 2 o 3 puntos libres representan un punto línea borde, y 5 puntos libres representan un punto esquina exterior.

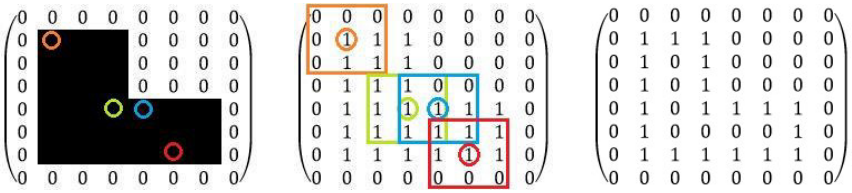


Imagen 12 - Lógica de detección de puntos esquina exterior, puntos esquina interior y puntos línea borde.

A continuación, con fines de reducir el coste computacional de los módulos siguientes, se lleva a cabo un filtrado de puntos definiendo, en función de la regularidad del entorno y la resolución espacial de la matriz binaria, un valor de mínima distancia que debe existir entre dos puntos contiguos línea borde. Conforme el valor de esta distancia aumenta, el coste computacional disminuye, pero lo hace en detrimento de la suavidad y precisión del grafo.

Finalmente, el tercer y último paso consiste en hacer un duplicado de los puntos contorno obtenidos hasta ahora. Este duplicado se realiza añadiendo puntos obstáculo en la dirección vertical y horizontal de cada punto en el instante en el que esta dirección se encuentra con un borde de la matriz o con un obstáculo. De esta forma se maximiza la distancia entre trayectorias y obstáculos, así como se mejora la suavidad del grafo.

3.1.2. Generación de trayectorias libres de colisiones

La generación de trayectorias libres de colisiones se lleva a cabo elaborando, en primer lugar, el diagrama de Voronoi sobre el conjunto de puntos contorno. Este diagrama, obtenido con la función *voronoi* de Matlab, necesita de un post-procesado que se compone de tres fases:

1. Eliminación de rectas que se encuentran de las zonas no transitables a una distancia menor que la distancia de seguridad definida. Esta distancia de seguridad viene dada en pixeles, de modo que su equivalencia en unidades métricas depende de la resolución espacial de la matriz binaria de entrada. El efecto que tiene esta distancia de seguridad sobre el grafo de trayectorias puede verse en la Imagen 13.
2. Eliminación de rectas inconexas de acuerdo a un parámetro de filtrado ajustable. En entornos que presentan geometrías que no son marcadamente verticales u horizontales, el diagrama que se obtiene presenta un número notable de trayectorias que no aportan información útil. El parámetro de filtrado limita el número máximo de rectas inconexas de una misma rama que se pueden eliminar de forma consecutiva. El efecto de este filtrado se muestra en la Imagen 14.
3. Fusionado de rectas. Gran parte de los algoritmos empleados en la planificación de la captura o la asignación de objetivos tienen un coste computacional proporcional al tamaño del grafo de trayectorias, por lo que es importante trabajar con grafos de número de nodos reducido. El proceso de fusionado se realiza en dos etapas. La primera consiste en el fusionado de rectas verticales y horizontales que pueden ser representadas mediante una única recta de mayor tamaño. La segunda es más compleja y está basada en el análisis de la diferencia de ángulo entre una recta y la recta resultante de fusionar esta con su contigua.

En el Anexo IV se explica con mayor profundidad cada una de estas fases.

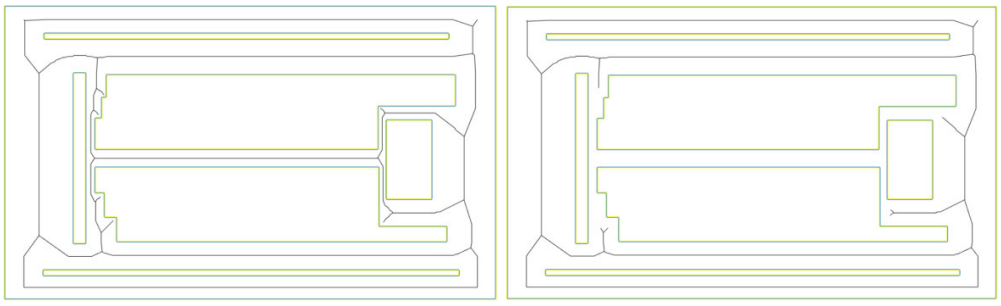


Imagen 13 - Efecto de la distancia de seguridad sobre las trayectorias generadas.

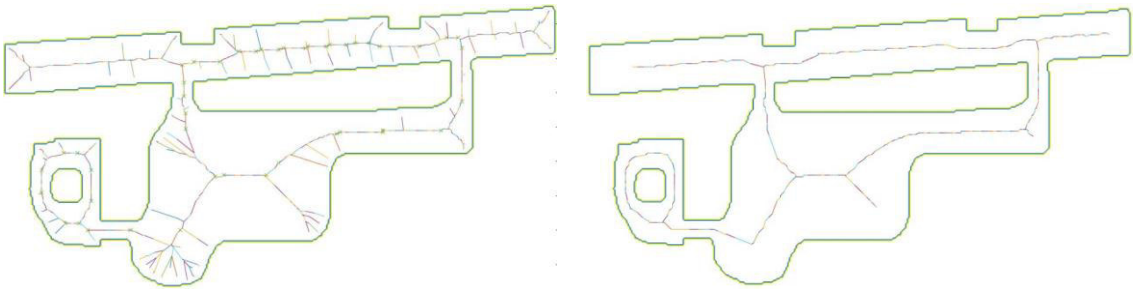


Imagen 14 - Efecto de eliminar las rectas inconexas de acuerdo a un parámetro de filtrado.

Como se ve en la Imagen 15, partiendo de un diagrama de Voronoi en bruto que, en el caso mostrado de ejemplo, contiene 3241 rectas, el post-procesado logra reducir el diagrama a un total de 95 rectas. Este diagrama post-procesado constituye el grafo de trayectorias.

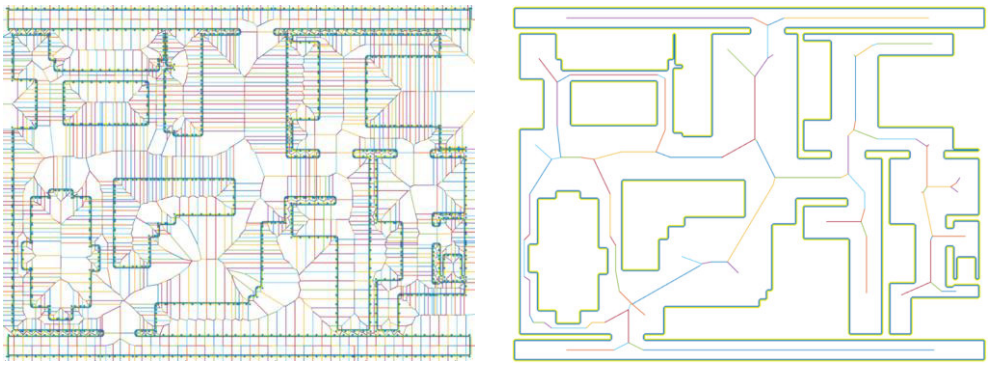


Imagen 15 – Izqda.: Diagrama de Voronoi en bruto. Dcha.: Diagrama de Voronoi post-procesado.

3.2. Elaboración del grafo regiones

Para poder localizar adecuadamente a los agentes en el entorno, así como extender la estrategia de captura por minimización del área segura a entornos complejos con obstáculos, es necesario particionar éste en regiones convexas sobre las que posteriormente se pueda computar de forma local el diagrama de Voronoi.

Para la elaboración del grafo regiones se ha empleado la triangulación de Delaunay. Los nodos del grafo están constituidos por los centroides de las regiones, mientras que las aristas representan la colindancia entre dos regiones, con un peso igual a la distancia entre sus centroides.

3.2.1. Procesado del mapa binario

Al igual que se hacía para el grafo de trayectorias, el primer paso consiste en la extracción de los puntos contorno de los obstáculos mediante la detección de puntos esquina exterior, puntos esquina interior y puntos línea borde, definiendo para estos últimos un valor de mínima distancia de separación.

A continuación, se lleva a cabo un filtrado de puntos haciendo uso de una máscara de convolución cuyo tamaño depende de la regularidad del mapa y la resolución espacial de la matriz binaria de entrada. Consiste en una matriz de elementos nulos salvo el central, la cual hace un barrido sobre la matriz de puntos contorno, actualizando ésta en el instante en que detecta un punto contorno.

Finalmente se hace un duplicado de los puntos que hacen frontera el borde de la matriz, siempre y cuando su distancia a este borde sea menor que un valor definido.

3.2.2. Generación de regiones convexas

La generación de regiones convexas se consigue computando la triangulación de Delaunay sobre los puntos contorno. Al igual que en el caso anterior, el grafo resultante necesita de un post-procesado, esta vez más sencillo, que consiste en eliminar aquellas regiones cuyo centroide se encuentre sobre un obstáculo.

La Imagen 16 muestra el grafo de regiones convexas resultante para el entorno empleado de ejemplo en este capítulo. En el Anexo V se muestran y se comentan los resultados obtenidos para

distintos tipos de entornos, observando que el grafo se vuelve menos uniforme y ordenado conforme la geometría del entorno se hace más irregular. No obstante, este hecho no aumenta el coste computacional de la captura, pues los bloques más pesados como la asignación de objetivos y la planificación de trayectorias se ejecutan sobre el grafo de trayectorias.

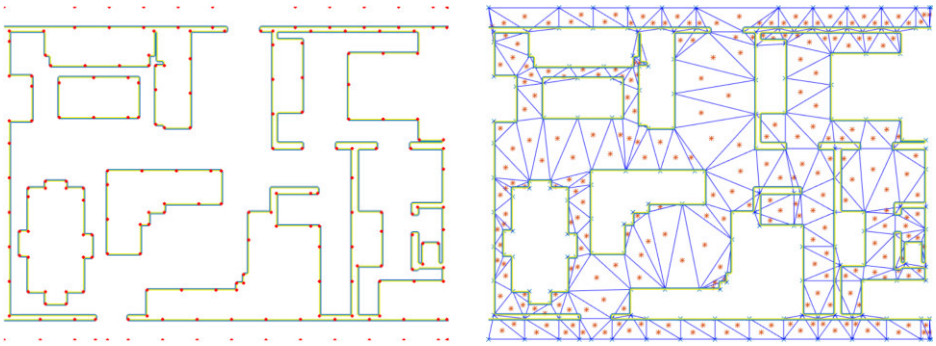


Imagen 16 - Izqda.: Procesado del mapa binario. Dcha.: Post-procesado de regiones convexas.

3.3. Vinculación de grafos

En las tareas de atrapamiento, indistintamente de las estrategias de captura, la localización de agentes se computa sobre el grafo de regiones, mientras que la planificación de la captura se hace sobre el grafo de trayectorias. Por lo tanto, es necesario llevar a cabo una vinculación entre grafos, de modo que a cada nodo región le corresponda un nodo trayectoria.

Esta vinculación de grafos se materializa en forma de vector, de modo que para cada nodo región, se almacena el nodo trayectoria visible más cercano a él. De este modo, la vinculación es unilateral y sólo permite migrar del grafo regiones al grafo trayectorias.

A modo de ejemplo, considérese el entorno mostrado en la Imagen 17, modelado por un grafo de trayectorias de 14 nodos (representado en rojo) y un grafo de regiones convexas de 47 nodos (representado en azul). La vinculación de grafos consistirá en asignar para cada uno de los 47 nodos región, el nodo trayectoria visible más cercano de los 14 nodos trayectoria existentes. Esta vinculación se materializa en un vector de 47 elementos en el que el valor $j \in [1,14]$ del elemento $i \in [1,47]$ indica que al nodo región i se le ha asociado el nodo trayectoria j. Esta vinculación es unilateral, pues mientras que sólo existe un nodo trayectoria asociado a cada nodo región, existen varios nodos región asociados a un mismo nodo trayectoria.

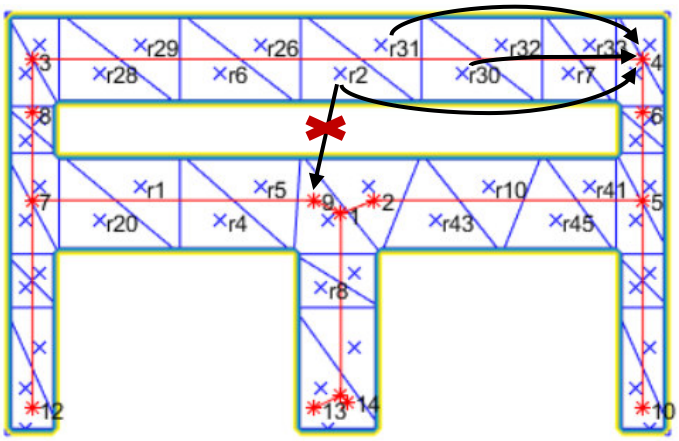


Imagen 17 - Vinculación de grafos. A cada nodo región (x) se le asocia el nodo trayectoria (*) visible más cercano.

4. Captura cooperativa de un evasor en entornos cerrados complejos

Para lograr la captura coordinada en entornos cerrados complejos, es necesario abordar fundamentalmente tres aspectos: la localización de agentes en los grafos, la asignación de rutas y la estrategia de atrapamiento.

El primero de ellos consiste en asignar, a cada agente, un nodo región y un nodo trayectoria en función de su localización en el entorno. El segundo consiste en obtener, para cada agente perseguidor, un conjunto de nodos trayectoria que definen una ruta hacia un agente evasor. El tercero se refiere a la lógica de movimiento de los perseguidores una vez tienen contacto visual con el evasor, para la cual se han propuesto tres estrategias**.**

Indistintamente de la estrategia de atrapamiento, los datos de entrada del algoritmo son la matriz binaria del entorno, el grafo de trayectorias, el grafo de regiones convexas y la posición de los agentes en el entorno. Como dato de salida se devuelve la acción de control que se debe aplicar a cada agente perseguidor.

4.1. Formulación del problema

Sea un grupo de $n_p \in \mathbb{N}$ agentes perseguidores y $n_e \in \mathbb{N}$ agentes evasores, con $n_e = 1$, en un entorno cerrado complejo $Q \subset \mathbb{R}^2$, con $Q_T \cup Q_{NT} = Q$, siendo Q_T y Q_{NT}, respectivamente, el subconjunto transitable y no transitable de Q.

Sea el grafo de trayectorias $G_T = (N_T,\ A_T)$ un grafo euclídeo no dirigido, con $N_T \in Q_T$ el conjunto de nodos y A_T el conjunto de aristas formado por pares de nodos adyacentes, y sea el grafo de regiones convexas $G_R = (C_R,\ A_R)$ otro grafo euclídeo no dirigido, con $C_R \in Q_T$ el conjunto de centroides de dichas regiones y A_R el conjunto de aristas formado por pares de centroides de regiones adyacentes. Sean además ADY_T y ADY_R las matrices de adyacencia asociada a dichos grafos y $RtoT$ el vector de dimensión C_R que, para cada nodo región, almacena el nodo trayectoria visible más cercano a él.

Sea $x_p^i \in Q_T$ con $i \in \{1, \dots, n_p\}$ las posiciones de los perseguidores y sea $x_e \in Q_T$ la posición del evasor. Sean u_p y u_e las acciones de control que mueven a los agentes por el entorno sujetas a la misma velocidad máxima $v_{máx}$.

El objetivo de los perseguidores es capturar al evasor. Se considerará que el evasor es capturado en un tiempo t_c cuando, siendo $r_c > 0$ el radio de captura, se cumpla:

$$\min_{i \in n_p} ||x_e(t_c) - x_p^i(t_c)|| < r_c$$

4.2. Localización de agentes en el grafo

La localización de agentes en el grafo se lleva a cabo a partir de la posición de cada agente, comprobando, en primer lugar, en qué región del grafo G_R está contenido cada uno de ellos. Una vez han sido asociados a un centroide del conjunto C_R, se asocian a un nodo del grafo G_T por medio del vector $RtoT$. La Imagen 18 ejemplifica este proceso.

Expresado matemáticamente, la localización de un agente k en el grafo consiste en:

1. Localización en G_R hallando ${C_R}^k \in C_R$ tal que $x^k \in {C_R}^k$.
2. Localización en G_T como ${N_T}^k = RtoT({C_R}^k)$.

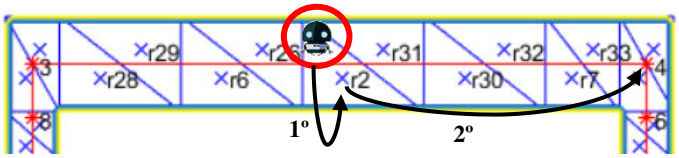


Imagen 18 - Localización de agentes en el grafo.

4.3. Asignación de rutas

Este módulo asigna, a cada agente perseguidor, un conjunto de nodos que definen una ruta hacia el evasor. Con el objetivo de que las rutas cubran, en conjunto, las posibles vías de escape del evasor, se ha diseñado en este proyecto una técnica que se ha denominado *método del peaje*. Éste método, que se explica a continuación, hace uso a su vez del algoritmo de Dijkstra, el cual determina el camino de menor coste entre un nodo origen y un nodo destino de un grafo. Este algoritmo cuenta con una extensa documentación y funciones que lo implementan, por lo que no ha sido necesaria su programación en este trabajo.

El método del peaje se trata de un algoritmo voraz, y está constituido por los siguientes bloques lógicos. En la Imagen 19 se muestra, en pseudocódigo, una posible implementación del mismo.

1. Algoritmo de Dijkstra para los n_p agentes perseguidores.
2. De las rutas recién calculadas, asignar la de menor distancia.
3. Aplicar peaje: Distorsionar el peso de las aristas que componen la ruta recién asignada.
4. Algoritmo de Dijkstra para los k perseguidores que no tienen trayectoria asignada.
5. Si $k > 1$, volver a 2.

Al distorsionar el peso de las aristas que ya componen una ruta, el algoritmo de Dijkstra devuelve, siempre que sea posible, caminos alternativos que todavía no han sido recorridos por un agente. De este modo se cubren las posibles vías de escape del evasor.

```
ADY_T_Modif = ADY_T;
Mientras haya agentes sin ruta asignada:
    Para cada agente perseguidor "i":
        Si no tiene ruta asignada:
            Dijkstra (ADY_T_Modif, N_T(evasor), N_T(i));
        Fin Si
        Asignar_Ruta (min(Distancias_rutas_sin_asignar));
        Actualizar_ADY_T_Modif ((Nueva_ruta_asignada, Peaje);
    Fin Para
Fin Mientras
```

Imagen 19 - Pseudocódigo del código de asignación de rutas.

La Imagen 20 muestra el efecto que tiene en la captura aplicar el método del peaje en la asignación de rutas, consiguiendo que los perseguidores avancen hacia el evasor cubriendo las posibles vías de escape del mismo.

El coste computacional de esta estrategia de asignación de rutas por peaje evoluciona de manera cuadrática con el número de perseguidores, pues la cantidad de veces que se ejecuta el algoritmo de Dijkstra responde a la siguiente expresión: $\sum_{i=1}^{n_p} i$.

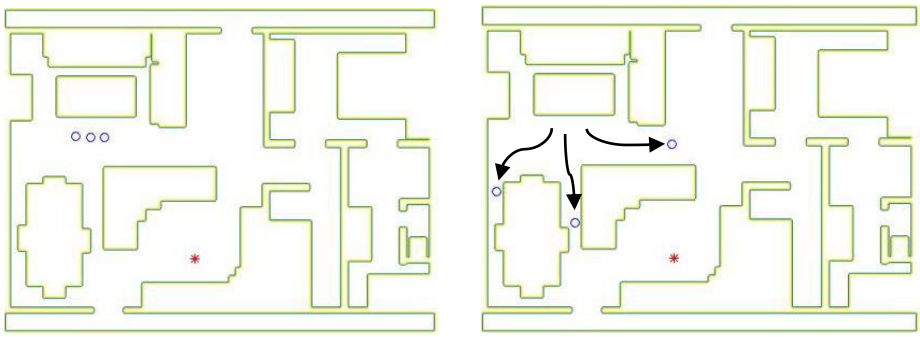


Imagen 20 - Asignación de rutas por peaje.

La asignación de rutas se ejecuta cada vez que, debido al movimiento del evasor por el entorno, cambia el nodo trayectoria que localiza al evasor en el grafo trayectorias, el cual actúa como nodo destino en el algoritmo de Dijkstra.

El nodo origen que se considera para cada perseguidor dependerá de si este tiene o no contacto visual con el evasor:

- Si no existe visibilidad, el nodo origen será el nodo trayectoria fijado por el perseguidor como dirección de avance en ese instante.
- Si existe visibilidad, el nodo origen será el nodo trayectoria vinculado a la región en la que se encuentra el perseguidor. En este caso, la ruta calculada será considerada sólo para el método del peaje, pero no en la lógica de movimiento del perseguidor.

Cuando un perseguidor pierda el contacto visual con el evasor, se le asignará inicialmente la trayectoria de mínima distancia al evasor computando el algoritmo de Dijkstra sobre la matriz ADY_T. Además, se fijará como dirección de avance el nodo más cercano al evasor de entre los dos más cercanos al perseguidor.

4.4. Estrategia de atrapamiento

La lógica de movimiento de los perseguidores es distinta en función de si estos tienen o no visibilidad con el evasor. Mientras el evasor esté fuera del campo de visión del perseguidor, este último avanzará, secuencialmente, hacia los nodos trayectoria que componen la ruta que le conduce hacia el evasor. Cuando un perseguidor tenga contacto visual con el evasor, ignorará la ruta que le sea asignada e iniciará la estrategia de atrapamiento propiamente dicha, de la cual se hace, a continuación, tres propuestas.

4.4.1. Atrapamiento directo

Esta técnica de atrapamiento consiste en avanzar en línea recta hacia la posición actual en la que se encuentra el evasor. La acción de control, u_p, viene dada por la siguiente expresión:

$$u_p^i = \frac{x_e - x_p^i}{\|x_e - x_p^i\|}$$

4.4.2. Atrapamiento predictivo

Esta estrategia radica en la idea de avanzar, no hacia la posición actual del evasor, sino hacia donde se predice que estará el evasor en un futuro. La acción de control, u_p, viene dada por la siguiente expresión:

$$u_p^i = \frac{x_{e,pred} - x_p^i}{\|x_{e,pred} - x_p^i\|}$$

Donde:

$$x_{e,pred} = x_e + u_{e,pred} \cdot K_{PRED}$$

$$u_{e,pred} = \frac{x_e - x_{e,last}}{\|x_e - x_{e,last}\|}$$

El parámetro que ajusta el nivel de predicción, K_{PRED}, toma distintos valores en función de la distancia que separa a los agentes.

$$\begin{cases} K_{PRED} = k \cdot r_c & si \quad \|x_e - x_p^i\| > k \cdot r_c \\ K_{PRED} < r_c & si \quad \|x_e - x_p^i\| \leq k \cdot r_c \end{cases}, \quad k \in \mathbb{N}$$

4.4.3. Atrapamiento por minimización del área segura

Esta estrategia es una extensión a entornos cerrados complejos de la estrategia de minimización del área segura implementada previamente en la captura coordinada en entornos cerrados convexos.

Para poder aplicarse en entornos complejos, es necesario definir, dentro del mismo, una región convexa sobre la que posteriormente se pueda computar Voronoi. Esto se consigue fusionando las regiones convexas que conectan, en G_R, al evasor con los perseguidores que tienen visibilidad con él, tal y como se ejemplifica en la Imagen 21.

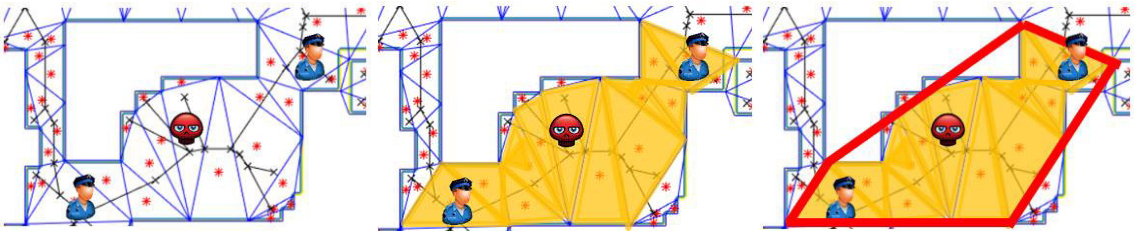


Imagen 21 - Definición de una región convexa en un entorno complejo no convexo.

Cuando el diagrama de Voronoi se elabora para tres o más agentes, puede ocurrir que haya perseguidores que no compartan ninguna de sus fronteras Voronoi con el evasor. En este caso, para los perseguidores que no sean colindantes con el evasor se computará la partición de Voronoi considerando únicamente al perseguidor y al evasor en la región convexa obtenida anteriormente. Para ello, dado que la función *voronoi* de Matlab trabaja con un mínimo de tres agentes, ha sido necesario crear una función, denominada *voronoi_2P,* que calcule la partición de Voronoi para dos puntos en un entorno cerrado convexo (Ver Anexo VI).

Por lo tanto, para el caso de dos agentes o agentes que no colinden con el evasor se emplea la función *voronoi_2P*, mientras que para el resto de casos se usa la función *voronoi_boundary_2D* ya desarrollada en la captura en entornos cerrados convexos y explicada en el Anexo III.

La dirección de avance del perseguidor viene dada por el centroide de la frontera Voronoi compartida con el evasor, C_{bi}.

$$u_p^i = \frac{C_{bi} - x_p^i}{\left\| C_{bi} - x_p^i \right\|}$$

En la Imagen 22 se muestra esta estrategia de minimización del área segura en entornos complejos aplicada a dos perseguidores y un evasor. Es importante observar que, dado que ahora las fronteras de la región convexa sobre la que se computa Voronoi pueden traspasarse, la eficacia que se obtenga con esta estrategia puede ser muy distinta a la obtenida anteriormente en entornos cerrados convexos.

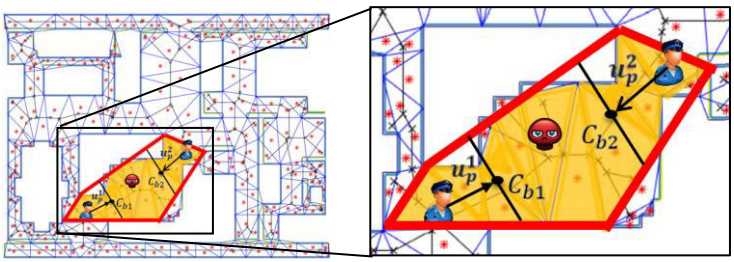


Imagen 22 - Extensión de la estrategia de minimización del área segura a entornos complejos.

4.5. Resultados y conclusiones

Ensayando por simulación los algoritmos de captura cooperativa de un único evasor en diversos entornos complejos se ha observado que, empleando la estrategia de atrapamiento predictivo y siempre y cuando el número de agentes perseguidores permita cubrir las posibles vías de escape que ofrece el entorno, se logra con éxito la captura del evasor en un tiempo finito.

Tras el análisis cualitativo de los resultados, se han observado diferencias apreciables en el grado de destreza de cada una de las estrategias de atrapamiento, las cuales se comentan a continuación. La evaluación cuantitativa por ensayo estadístico se lleva a cabo en el capítulo 6.

- Bajo la estrategia de atrapamiento predictivo se hace imposible sortear de frente a un perseguidor, por lo que tener cubiertas las vías de escape del evasor es garantía de captura (Imagen 23).
- La estrategia de atrapamiento directo se asemeja a la estrategia anterior en espacios estrechos, pero se vuelve menos efectiva en zonas amplias en las que es posible sortear de cara a un perseguidor.

- La estrategia de minimización del área segura resulta ser la menos efectiva. Dado que ahora las fronteras de la región convexa sobre la que se computa Voronoi pueden traspasarse (Imagen 24), la lógica de movimiento que antes aseguraba la minimización del área segura del evasor deja de ser válida, facilitando la labor de huida del evasor.

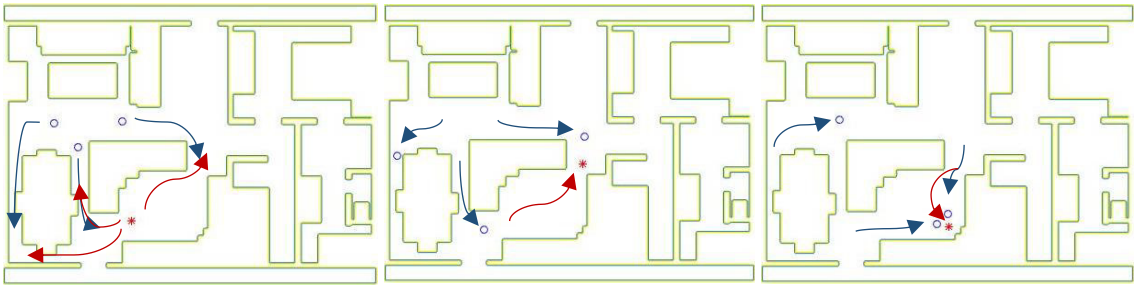


Imagen 23 - Captura cooperativa de un evasor en un entorno complejo bajo la estrategia de atrapamiento predictivo.

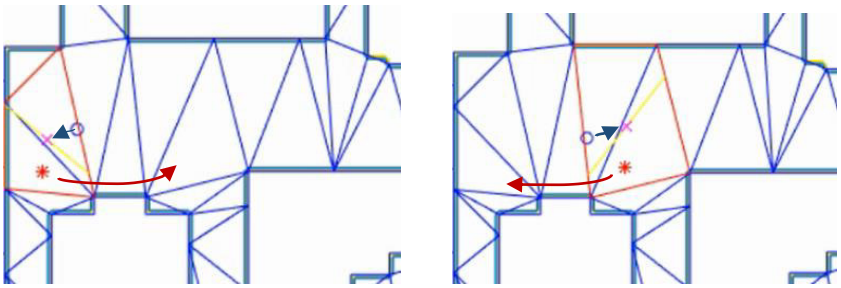


Imagen 24 – Ineficacia en entornos complejos de la estrategia de minimización del área segura.

5. Captura cooperativa de múltiples evasores en entornos cerrados complejos

Si se desea extender la metodología de captura presentada en el capítulo anterior a capturas con múltiples evasores, es necesario considerar, además de la localización de agentes en los grafos, la asignación de rutas y la estrategia de atrapamiento, un cuarto aspecto: la asignación de objetivos.

La asignación de objetivos pretende adjudicar, a cada perseguidor, un evasor sobre el que computar la captura cooperativa junto al resto de perseguidores que tengan adjudicado el mismo evasor. De este modo, la captura cooperativa de múltiples evasores no es más que un tratamiento en paralelo de capturas cooperativas de un único evasor.

5.1. Formulación del problema

Sea el problema formulado en el capítulo 4.1. de la memoria, extendido al caso de múltiples evasores, con $n_e \geq 1$, siendo $x_e^j \in Q_T$ con $j \in \{1, \dots, n_e\}$ las posiciones de los evasores.

Se considerará que un evasor j es capturado en un tiempo t_c cuando, siendo $r_c > 0$ el radio de captura, se cumpla:

$$\min_{i \in n_p} ||x_e^j(t_c) - x_p^i(t_c)|| < r_c$$

5.2. Asignación de objetivos

La asignación de objetivos constituye un problema de programación lineal, conocido como problema de transporte, en el cual se debe minimizar el coste de abastecimiento a una serie de puntos demanda (agentes evasores) a partir de un grupo de puntos oferta (agentes perseguidores), teniendo en cuenta los distintos precios de envío (distancias) de cada punto oferta a cada punto demanda.

Las soluciones a los problemas de transporte se sustentan tradicionalmente en un conjunto de métodos para resolver problemas de programación lineal conocidos como el método Simplex, siendo por tanto el empleado en este proyecto.

La herramienta Matlab integra una función para este fin que resuelve problemas de programación entera, denominada `intlinprog`, que internamente implementa el método Simplex. El problema queda definido por los parámetros siguientes, explicados en el capítulo 5.2.1.

```
x = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub)
```

$$\rightarrow \quad \min_x f^T x \text{ subject to } \begin{cases} x(\text{intcon}) \text{ are integers} \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub. \end{cases}$$

5.2.1. Formulación del problema

Sea $X = \mathcal{M}_{n_p x n_e} \in \{0,1\}$ la matriz de asignación de objetivos, con $X_{ij} = 1$ simbolizando que el perseguidor i tiene asignado el evasor j. Sea $C = \mathcal{M}_{n_p x n_e}(\mathbb{R})$ la matriz de costes, siendo C_{ij} la distancia mínima entre el nodo del perseguidor i y el nodo del evasor j en el grafo G_T.

Se define la función objetivo como el sumatorio de las distancias mínimas que separan, en G_T, a cada perseguidor i de cada uno de los evasores j:

$$f(X) = \sum_{i=1}^{n_p} \sum_{j=1}^{n_e} C_{ij} \cdot X_{ij}$$

Cada uno de los agentes perseguidores debe tener asignado un evasor, lo que introduce una restricción de tipo 1:

$$\sum_{j=1}^{n_e} X_{ij} = 1, \quad \forall\, i \in \{1, \dots, n_p\}$$

Además, cada agente evasor tiene limitada la cantidad de agentes perseguidores asignables, lo que introduce una restricción de tipo 2. Sea $P_{máx} \in \mathbb{N}$ la cantidad máxima de agentes perseguidores asignables a un mismo evasor:

$$\sum_{i=1}^{np} X_{ij} \leq P_{máx}, \quad \forall\, j \in \{1, \dots, n_e\}$$

De modo que el problema de asignación de objetivos queda definido como sigue:

$$\min_X \sum_{i=1}^{n_p} \sum_{j=1}^{n_e} C_{ij} \cdot X_{ij} : \begin{cases} \sum_{j=1}^{n_e} X_{ij} = 1 \,, \forall\, i \\ \sum_{i=1}^{np} X_{ij} \leq P_{máx} \,, \forall\, j \\ X_{ij} \in \{0,1\} \end{cases}, \quad i \in \{1, \dots, n_p\} \quad j \in \{1, \dots, n_e\}$$

5.2.2. Resolución del problema

El problema de asignación recién formulado se trata de un problema de programación entera y ha sido resuelto por el Método Simplex empleando la función de Matlab *intlinprog*.

De acuerdo al tipo de argumentos con los que trabaja esta función, la matriz de costes y la matriz de asignación han tenido que ser vectorizadas, mientras que las restricciones han tenido que ser introducidas con ecuaciones e inecuaciones matriciales.

La función devuelve, en forma de vector, la matriz de asignación de objetivos que minimiza la función objetivo. Ahora ya es posible ejecutar en paralelo varias capturas cooperativas de un único evasor en las que, en cada una, participan los perseguidores que tienen asignado un mismo evasor.

La Imagen 25 muestra, con un sencillo ejemplo, la ventaja de tratar la asignación de objetivos como un problema de programación lineal, pues con ello se consigue que el reparto de evasores favorezca la minimización del tiempo global de captura.

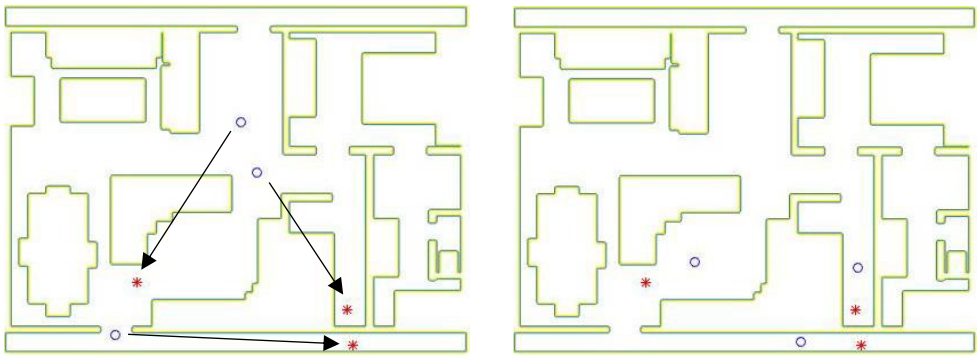


Imagen 25 - Asignación de objetivos.

5.2.3. Actualización de objetivos

Cuando la captura ocurre con evasores no estáticos, la asignación de objetivos debe de actualizarse frecuentemente ya que, debido al movimiento de los evasores por el entorno, la solución óptima que minimiza la función objetivo puede variar durante el transcurso de la captura.

La actualización de objetivos puede ejecutarse cada un periodo dado de tiempo, cada cierta distancia recorrida o bien cada vez que un agente evasor vea modificada su localización en el grafo. La elección del criterio de actualización va a depender mucho del tipo de entorno, la capacidad computacional del sistema y la dinámica de la captura. En este caso concreto se ha optado por definir un margen de tiempo teniendo en cuenta los aspectos mencionados.

5.3. Resultados y conclusiones

La captura cooperativa de múltiples evasores en entornos complejos se ha evaluado bajo la estrategia de atrapamiento predictivo, pues es la que mejor desempeño ha demostrado en capturas de un único evasor, y bajo una lógica de evasión automatizada que ha resultado ser insuficiente, por lo que los ensayos realizados sólo han permitido una evaluación cualitativa.

Pese a que en las simulaciones realizadas se ha logrado la captura de todos los evasores en un tiempo finito, también debido a una lógica de evasión sencilla, la asignación de objetivos manifiesta un reparto de perseguidores ineficiente que bajo una lógica de evasión manual podría no lograr la captura en un tiempo finito. Esta limitación en la lógica de asignación de objetivos puede deberse a los siguientes motivos:

- Dado que la captura cooperativa de múltiples evasores no es más que un tratamiento en paralelo de capturas de un único evasor, es importante que la cantidad mínima de perseguidores asignados a un evasor, P_{min}, sea tal que se puedan cubrir sus vías de escape. No obstante, dado que un evasor también puede quedar desatendido, se llega a una restricción no lineal que añade mucha complejidad al problema de programación lineal.

$$\boxed{\sum_{i=1}^{np} X_{ij} = 0} \quad \text{ó} \quad \boxed{P_{mín} \leq \sum_{i=1}^{np} X_{ij} \leq P_{máx}}, \qquad \forall\, j \in \{1, \dots, n_e\}$$

- La actualización de objetivos implementada en este trabajo, al estar poco refinada, ocasiona un constante cambio en los evasores que son atendidos y los perseguidores que le son asignados, lo que se traduce en un tratamiento paralelo pero inconsistente de capturas de un único evasor.
- La función objetivo se ha definido como el sumatorio de las distancias mínimas que separan a cada perseguidor de cada evasor, sin considerar la posterior distorsión de distancias que realiza el *método del peaje* en la asignación de trayectorias. Por tanto, esta función objetivo podría no estar vinculada al coste temporal de la captura.

Bien ejecutado, tratar la asignación de objetivos como un problema de programación lineal entera permite realizar un reparto inteligente de perseguidores de tal manera que se reduzca el tiempo global de captura. Sin embargo, una ejecución sencilla como la realizada en este proyecto puede llegar a ofrecer peores resultados que los que se obtendrían por medio de una captura secuencial de evasores. En general y como primera aproximación, los resultados son satisfactorios, pero sería conveniente profundizar en los aspectos mencionados anteriormente.

La Imagen 26 muestra los resultados obtenidos en la simulación manual de una captura cooperativa de seis evasores en un entorno complejo empleando tres perseguidores. Los evasores se representan con asteriscos, o cruces si han sido capturados, y los perseguidores con círculos. Se observa como, pese a las limitaciones encontradas en el algoritmo de asignación de objetivos, se logra la captura de todos los evasores en un tiempo finito.

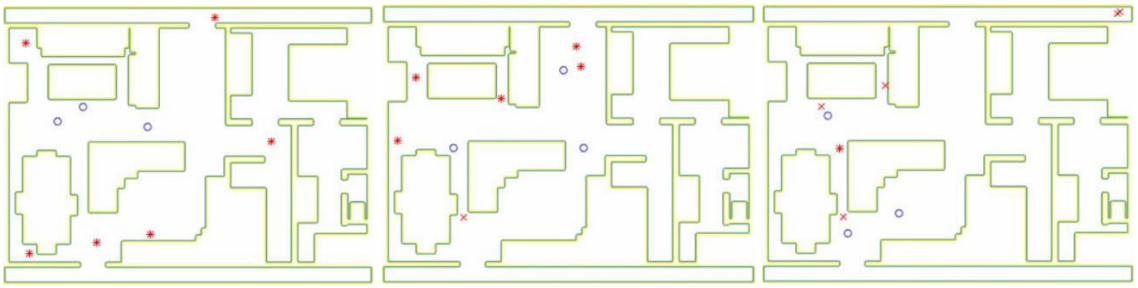


*Imagen 26 - Captura cooperativa de múltiples evasores (representados como *) en un entorno complejo.*

6. Experimentación

En primer lugar, se lleva a cabo un ensayo estadístico de las estrategias de atrapamiento. Los algoritmos se evalúan frente a un único evasor, en entornos y condiciones diferentes, y con una lógica de evasión programada. Posteriormente, dada las limitaciones de esta lógica de evasión, se realiza un segundo ensayo con el evasor controlado manualmente.

En segundo lugar, y a modo de prueba de concepto, se realiza un experimento con robots reales y el entorno de desarrollo ROS.

6.1. Lógica de evasión programada

A la hora de hacer un ensayo automatizado de las distintas estrategias de atrapamiento, ha sido necesario desarrollar una lógica de evasión que permita conducir al evasor hacia un objetivo, sorteando los agentes que tratan de darle captura. La estrategia que a continuación se propone es similar a la empleada en la asignación de trayectorias por el método del peaje vista en el capítulo 4.3. de la memoria.

En primer lugar, considerando que el evasor tiene un alcance de visión infinita, se analiza con qué perseguidores tiene visibilidad. A continuación, se obtiene mediante el algoritmo de Dijkstra, para cada uno de los perseguidores visibles, la ruta de nodos del grafo G_T que les dirige hacia el evasor. Finalmente se modifica, con un valor de peaje, el peso de las aristas que conducen al perseguidor al nodo intermedio de la ruta.

Una vez distorsionado el peso de las aristas que conducen al evasor hacia sus perseguidores y hacia su consecuente captura, únicamente queda aplicar el algoritmo de Dijkstra para generar la ruta del evasor hacia su objetivo. Esta ruta, gracias a la previa distorsión de aristas, sorteará, en la medida de lo posible, los agentes perseguidores.

La Imagen 27 muestra una simulación de captura con tres perseguidores y un evasor en la que este último es dirigido hacia su objetivo por la lógica de evasión de forma acertada.

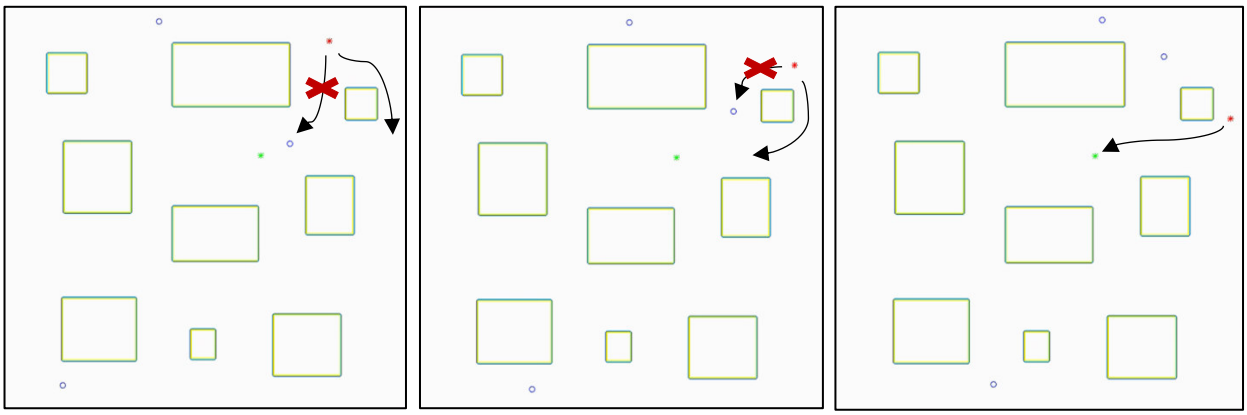


Imagen 27 – Simulación de captura con 3 perseguidores y 1 evasor automatizados.

6.2. Ensayo estadístico de las estrategias de atrapamiento

Las estrategias de atrapamiento se evalúan frente a un único evasor, bajo condiciones diferentes y con una lógica de evasión tanto programada como manual. El estudio se lleva a cabo para los dos entornos mostrados en la Imagen 28.

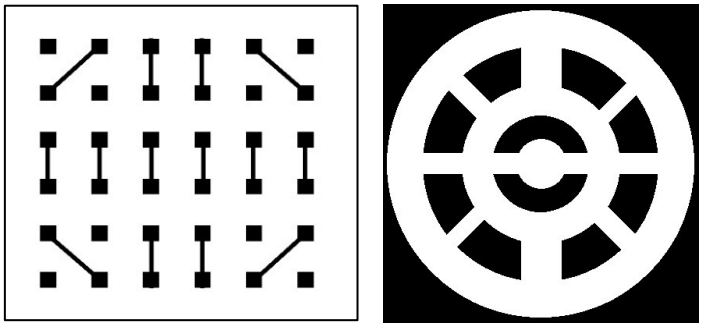


Imagen 28 - Entornos de ensayo de algoritmos. Izqda.: Entorno1. Dcha.: Entorno2.

La metodología empleada es la siguiente. Al inicio de cada simulación se sitúa aleatoriamente a los agentes en el grafo G_T y se genera, también de forma aleatoria y en el grafo G_T, cinco premios. El objetivo del evasor es obtener los premios sin ser capturado, mientras que el objetivo de los perseguidores es dar caza al evasor en el menor tiempo posible. Además, el evasor tiene limitada la distancia máxima que puede recorrer sin alcanzar un premio; si la sobrepasa se considera que ha sido capturado y se inicia de nuevo la simulación. Por último, no puede haber más de un premio activo al mismo tiempo.

Como variables independientes, se tiene:

- Entorno.
- Estrategia de atrapamiento.
- Número de agentes perseguidores.
- MR: determina si el evasor tiene (MR=0) o no (MR=1) conocimiento total de la posición de los perseguidores, independiente de que tenga o no contacto visual con ellos.

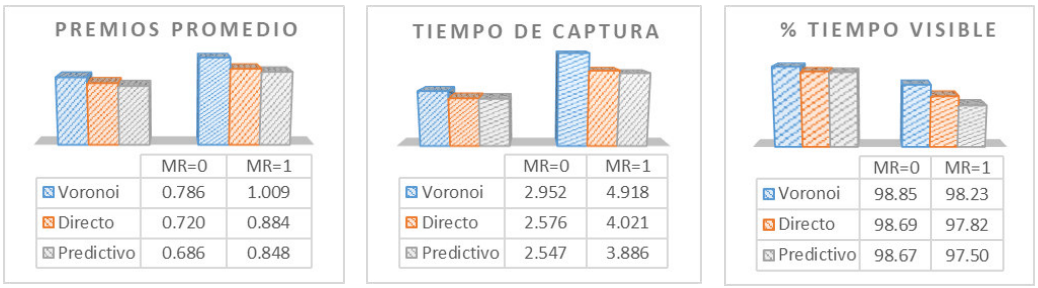
Las variables dependientes que se evalúan en el ensayo son:

- Promedio y desviación típica de la cantidad de premios recogidos.
- Promedio y desviación típica de la distancia total recorrida por el evasor.
- Promedio y desviación típica del tiempo de captura.
- Promedio y desviación típica del tiempo que el evasor es visto por algún perseguidor.
- Porcentaje promedio respecto al tiempo de captura que el evasor es visto.
- Promedio y desviación típica del tiempo de ejecución de la asignación de trayectorias.
- Promedio y desviación típica del tiempo de ciclo de ejecución del algoritmo completo.

Tanto la cantidad de premios promedio recogidos por el evasor, el tiempo que transcurre hasta que el evasor es capturado, o el porcentaje respecto al tiempo de captura que el evasor está siendo visto por al menos un perseguidor visible, son variables dependientes que reflejan la superioridad del evasor frente a los perseguidores. Por lo tanto, dado que la destreza del evasor se supone constante en todas las simulaciones, valores elevados en estos tres parámetros indicarán una menor eficacia de la estrategia de atrapamiento.

6.2.1. Ensayo automatizado

Para cada configuración se ha realizado un total de 2000 simulaciones, mostrando en la Imagen 29 y la Imagen 30 los resultados promediados más relevantes para cada entorno de ensayo.

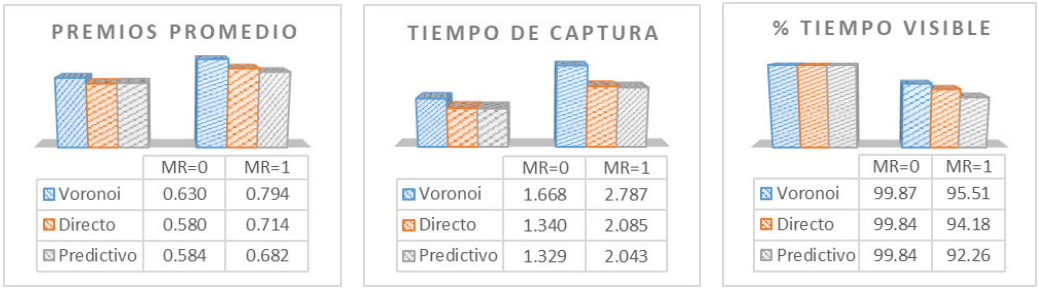


	MR=0	MR=1
Voronoi	0.786	1.009
Directo	0.720	0.884
Predictivo	0.686	0.848

	MR=0	MR=1
Voronoi	2.952	4.918
Directo	2.576	4.021
Predictivo	2.547	3.886

	MR=0	MR=1
Voronoi	98.85	98.23
Directo	98.69	97.82
Predictivo	98.67	97.50

Imagen 29 - Resultados del ensayo automatizado en el Entorno1.



	MR=0	MR=1
Voronoi	0.630	0.794
Directo	0.580	0.714
Predictivo	0.584	0.682

	MR=0	MR=1
Voronoi	1.668	2.787
Directo	1.340	2.085
Predictivo	1.329	2.043

	MR=0	MR=1
Voronoi	99.87	95.51
Directo	99.84	94.18
Predictivo	99.84	92.26

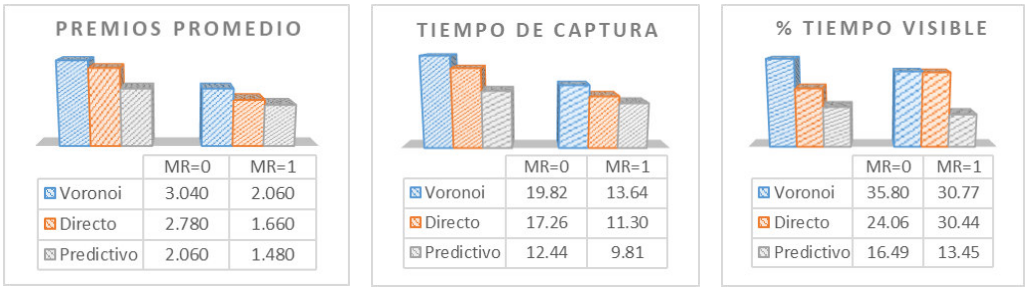
Imagen 30 - Resultados del ensayo automatizado en el Entorno2.

Analizando los resultados del ensayo automatizado, se llega a las siguientes conclusiones:

- Dadas las limitaciones de la lógica de evasión, el ensayo automatizado puede no ser adecuado para la evaluación de los algoritmos. Por este motivo, la realización de un ensayo manual está justificado.
- Contrariamente a lo que cabría esperar, la lógica de evasión es más efectiva cuando ésta sólo tiene en cuenta a los perseguidores con los que el evasor tiene contacto visual (MR=1).
- Indistintamente de MR y del entorno, la estrategia de atrapamiento que demuestra una mayor eficacia es la del atrapamiento predictivo, seguida muy de cerca por el atrapamiento directo. La estrategia de minimización del área segura resulta ser la menos eficaz.

6.2.2. Ensayo manual

Para cada configuración se ha realizado un total de 50 simulaciones. Los resultados promedio más relevantes para cada entorno se muestran en la Imagen 31 y la Imagen 32.

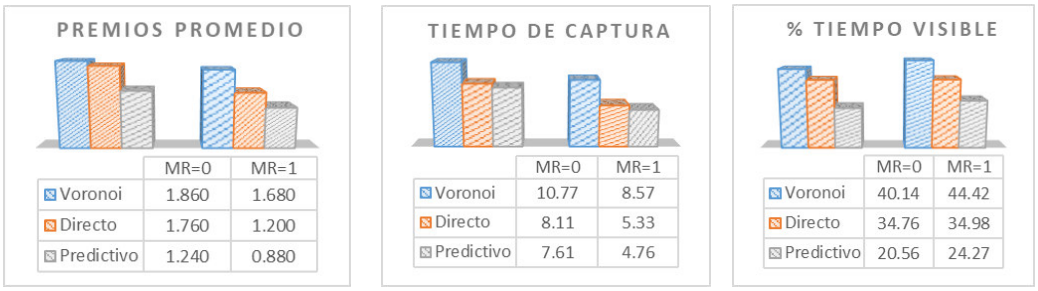


	MR=0	MR=1
Voronoi	3.040	2.060
Directo	2.780	1.660
Predictivo	2.060	1.480

	MR=0	MR=1
Voronoi	19.82	13.64
Directo	17.26	11.30
Predictivo	12.44	9.81

	MR=0	MR=1
Voronoi	35.80	30.77
Directo	24.06	30.44
Predictivo	16.49	13.45

Imagen 31 - Resultados del ensayo manual en el Entorno1.



	MR=0	MR=1
Voronoi	1.860	1.680
Directo	1.760	1.200
Predictivo	1.240	0.880

	MR=0	MR=1
Voronoi	10.77	8.57
Directo	8.11	5.33
Predictivo	7.61	4.76

	MR=0	MR=1
Voronoi	40.14	44.42
Directo	34.76	34.98
Predictivo	20.56	24.27

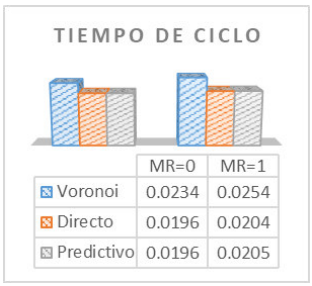
Imagen 32 - Resultados del ensayo manual en el Entorno2.

Analizando los resultados del ensayo manual, se llega a estas otras conclusiones:

- El ensayo manual, debido a un muestreo reducido, puede generar datos falsos que no permitan una valoración cuantitativa de algoritmos, pero sí una valoración cualitativa.
- Como cabía esperar, el evasor es más capaz de evitar la captura cuando conoce la localización de los perseguidores indistintamente de que exista contacto visual (MR=0).
- De acuerdo a lo concluido en el ensayo automatizado, indistintamente de MR y del entorno, la estrategia de atrapamiento predictivo resulta ser la más eficaz, esta vez con una diferencia bastante notable. El atrapamiento por minimización del área segura resulta ser la estrategia menos efectiva.

6.2.3. Coste computacional

Si se analiza el coste computacional de los algoritmos, según se muestra en la Imagen 33, el atrapamiento directo y el predictivo tienen coste idéntico, mientras que el atrapamiento Voronoi tiene un coste computacional algo mayor.



	MR=0	MR=1
Voronoi	0.0234	0.0254
Directo	0.0196	0.0204
Predictivo	0.0196	0.0205

Imagen 33 - Coste computacional de las estrategias de atrapamiento.

6.2.4. Conclusiones

En este apartado se han analizado las diferentes estrategias de atrapamiento, concluyendo que el algoritmo que mejores resultados ofrece es el atrapamiento predictivo, pues éste se anticipa al movimiento del evasor, reduciendo el tiempo de captura y el porcentaje de tiempo visible frente a las otras estrategias. Además, también ha demostrado ser la estrategia de menor coste computacional junto con el atrapamiento directo. El atrapamiento por minimización del área segura ha resultado ser la estrategia menos efectiva.

6.3. Implementación en ROS

Hasta ahora, las estrategias de captura han sido implementadas en Matlab, considerando los robots como puntos en un espacio 2D. Una representación simplificada puede ser suficiente para realizar una evaluación de algoritmos por simulación, pero no es prueba suficiente de su implementabilidad en equipos reales dado que en estos casos existen otras dificultades como la auto-localización del robot en su entorno o el movimiento autónomo del mismo. Con ROS, es posible realizar simulaciones realistas e implementaciones en equipos reales de forma sencilla sin necesidad de abordar estas dificultades.

6.3.1. Introducción a ROS

ROS (Robot Operative System [11]) es un framework usado de manera generalizada en el mundo de la robótica. Esta plataforma funciona en conjunto con un sistema operativo y proporciona una serie de servicios y librerías que facilitan el desarrollo estandarizado de software para robots.

En ROS, los procesos que constituyen un sistema robótico se definen en ejecutables, denominados nodos, los cuales pueden comunicarse entre sí por medio de canales de comunicación, denominados tópicos. Los nodos son independientes entre sí, y todos ellos puede publicar o suscribirse a un tópico indistintamente de que otros nodos publiquen o estén suscritos al mismo tópico. De este modo, los nodos se combinan dentro de un grafo computacional para crear ejecuciones completas, existiendo un nodo maestro que gestiona la comunicación entre procesos, permitiendo el registro y la búsqueda de nodos (Imagen 34).

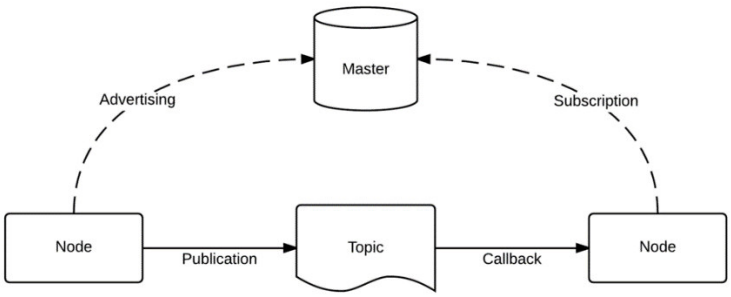


Imagen 34 - Esquema de funcionamiento de ROS.

La implementación en ROS se realiza en este trabajo con ayuda de Stage, un simulador de robots móviles en 2D. Una vez comprobado su funcionamiento a nivel de simulación, se lleva a cabo la implementación en un entorno real y con un equipo real de robots. Es importante mencionar que, a efectos de código, es transparente el que se esté trabajando con un robot real o simulado, pues el simulador emite un tópico sobre la odometría del robot del mismo modo que lo hacen el conjunto de nodos que representan al robot empleado en la prueba real.

6.3.2. Implementación en ROS de los algoritmos de captura

La Imagen 35 esquematiza la comunicación que se establece entre Matlab y un robot cualquiera, ya sea real o simulado en Stage. Se observa que los robots están representados por un nodo de localización y tres nodos de navegación que se encargan de la planificación a alto nivel, el movimiento a bajo nivel y la evitación de obstáculos.

El nodo de localización hace uso de sensores infrarrojos y el algoritmo AMCL (Adaptative Monte Carlo Localization) para estimar la posición y orientación del robot. Esta información es publicada en el tópico *amcl_pose*. Matlab se suscribe a él y utiliza esta información para ejecutar el algoritmo de captura. Las acciones de control calculadas se publican en otro tópico, *matlab_path*, de modo que son recibidas y aplicadas por los nodos de navegación por medio de actuadores.

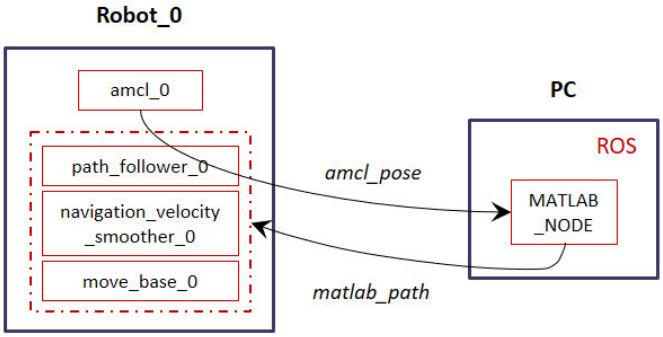


Imagen 35 – Arquitectura de la implementación realizada en ROS, mostrada para un único robot.

A efectos de código, la implementación en ROS hace que ya no sea necesario simular en Matlab la posición y el movimiento de los agentes que intervienen en la captura, sino que ahora esa información se genera externamente por simulación o por un equipo real de robots, de modo que Matlab solicita esa información, ejecuta el algoritmo de captura, y devuelve la acción de control que debe aplicarse a cada agente perseguidor.

6.3.3. Resultados y conclusiones

La Imagen 36 muestra los resultados obtenidos de una captura cooperativa real bajo la lógica de atrapamiento directo de un robot evasor, representado de color verde, en un entorno complejo empleando dos robots perseguidores, representados en rojo y morado. Las imágenes también muestran las trayectorias asignadas a cada perseguidor. Cuando existe contacto visual con el evasor se inicia la lógica de atrapamiento directo, avanzando directamente hacia la posición del evasor.

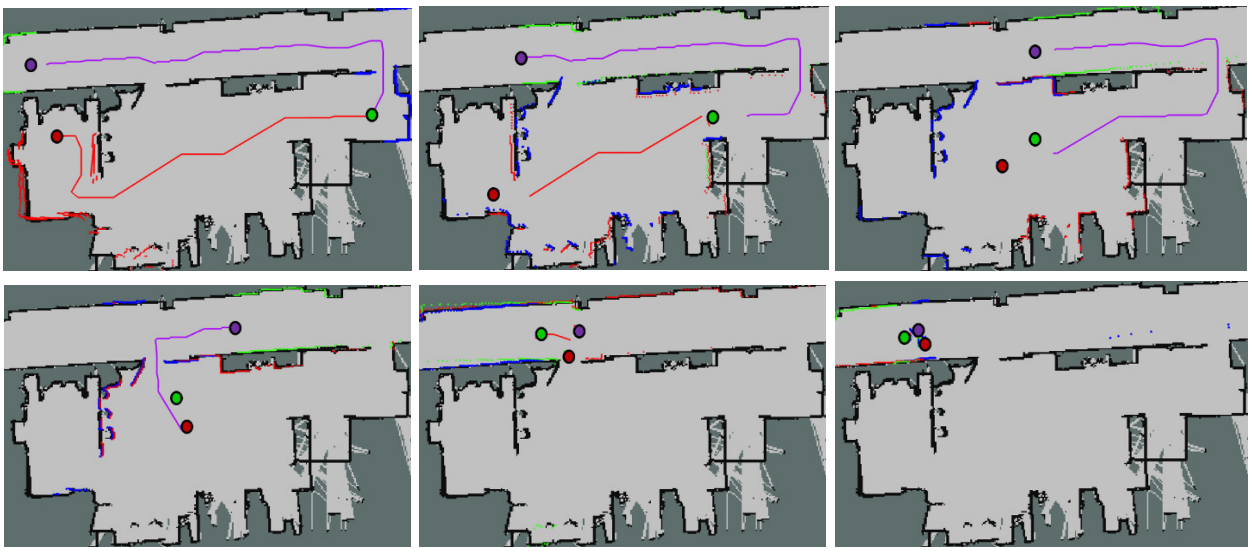


Imagen 36 - Visualización de resultados con Rviz de la captura de 1 robot evasor (representado en verde) realizada en un entorno real con un equipo real de robots.

Este experimento demuestra la viabilidad de implantar en una plataforma real las técnicas de modelado y las estrategias de captura desarrolladas en este trabajo final de máster. Las trayectorias asignadas consiguen el avance efectivo y seguro de los perseguidores hacia el evasor, cubriendo las vías de escape y logrando en última instancia la captura del evasor en un tiempo finito.

El modelo del entorno sobre el que se ha realizado el experimento con robots reales, así como los robots empleados en el experimento, se muestran respectivamente en la Imagen 37 y la Imagen 38.

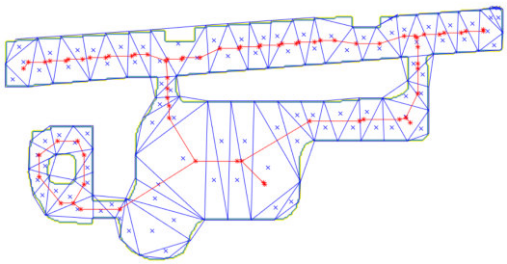


Imagen 37 - Modelado en forma de grafos del entorno sobre el que se ha realizado el ensayo con robots reales.

Imagen 38 - Robots empleados en el experimento.

7. Conclusiones y márgenes de ampliación

7.1. Trabajo realizado

El objetivo principal de este proyecto ha sido el desarrollo de estrategias de captura cooperativa de agentes evasores en entornos complejos utilizando para ello un sistema multi-robot. En base a los resultados obtenidos, se puede afirmar que se ha alcanzado con éxito este objetivo, habiendo logrado con éxito la captura cooperativa de múltiples evasores en diferentes entornos complejos.

El trabajo ha comenzado con la reproducción de las técnicas presentadas en un artículo de investigación que aborda la captura de evasores en entornos cerrados convexos libres de obstáculos mediante una estrategia de minimización del área segura de los evasores, para lo que hace uso de los diagramas de Voronoi. Conseguido esto, se ha visto la limitación que supone la premisa de que el entorno deba de ser convexo, alejándolo de la implementación en entornos reales, por lo que se ha decidido extender la estrategia presentada en el artículo a entornos cerrados complejos, así como desarrollar nuevas estrategias.

Para extender la captura en entornos cerrados complejos, ha sido necesario abordar en primer lugar el modelado de entornos complejos para así obtener un modelo sobre el cual computar posteriormente la estrategia de captura. Tras el correspondiente estudio del estado del arte, se ha desarrollado e implementado una estrategia de modelado en forma de grafo, para así poder aplicar en la captura la teoría de grafos. El modelo está compuesto de dos grafos, uno que constituye un mapa de trayectorias libre de obstáculos elaborado haciendo uso de los diagramas de Voronoi, y otro elaborado haciendo uso de la triangulación de Delaunay que particiona el entorno en regiones convexas.

Abordado el modelado, se ha abordado la captura cooperativa de un único evasor en entornos cerrados complejos, encontrando especial dificultad en la asignación de rutas y en las estrategias de atrapamiento. Para el primero se ha ideado una técnica denominada en este trabajo como *método del peaje* que hace uso del algoritmo de Dijkstra y logra una asignación de rutas que cubre las posibles vías de escape del evasor. Para el segundo, se ha extendido la estrategia de minimización del área segura haciendo uso del grafo de regiones convexas y se han desarrollado nuevas estrategias: el atrapamiento directo y el atrapamiento predictivo.

Ensayando cualitativamente por simulación los algoritmos de captura cooperativa de un único evasor en diversos entornos complejos se ha observado que, empleando la estrategia de atrapamiento predictivo y siempre y cuando el número de agentes perseguidores permitía cubrir las posibles vías de escape que ofrecía el entorno, se logra con éxito la captura del evasor en un tiempo finito, por lo que tener cubiertas las vías de escape del evasor es garantía de captura. La estrategia de minimización del área segura ha resultado ser la menos efectiva dado que ahora las fronteras de la región convexa sobre la que se computa Voronoi pueden traspasarse, por lo que la lógica de movimiento que antes aseguraba la minimización del área segura del evasor deja de ser válida, facilitando la labor de huida del evasor.

La captura cooperativa de múltiples evasores en entornos complejos ha requerido abordar el problema de asignación de objetivos, resuelto como un problema de programación lineal entera por medio del método Simplex. Se ha observado que, pese a lograr la captura de todos los evasores

en un tiempo finito, la asignación de objetivos manifiesta un reparto de perseguidores ineficiente que bajo una lógica de evasión manual podría no lograr la captura en un tiempo finito.

Se ha realizado un ensayo estadístico de los algoritmos para evaluar las diferentes estrategias de atrapamiento, concluyendo que el atrapamiento predictivo es la estrategia que ofrece mejores resultados, seguida de cerca por el atrapamiento directo, quedando en último lugar la estrategia de minimización del área segura.

Finalmente, se ha realizado una implementación con robots reales haciendo uso de la plataforma ROS, logrando con éxito la captura y demostrando así la viabilidad de implementar sobre plataformas reales las estrategias de captura desarrolladas en este trabajo.

7.2. Trabajo futuro

En la línea de trabajo realizado en este TFM sería interesante profundizar en los siguientes aspectos:

- Optimización de las estrategias de modelado y estudio de nuevas técnicas.
- Asignación de rutas en base al análisis discreto del grafo regiones: Dado que con la estrategia de atrapamiento predictivo tener cubiertas las vías de escape del evasor es garantía de captura, sería interesante abordar la captura bajo un enfoque discreto sobre un grafo de regiones, pudiendo lograr una asignación de rutas más eficaz que la obtenida con el método del peaje.
- Lógica de asignación de objetivos: Se ha observado que esta lógica manifiesta un reparto de perseguidores ineficiente que en determinados casos podría no lograr la captura en un tiempo finito. Como primera aproximación, los resultados son satisfactorios, pero es necesario analizar las causas diagnosticadas en el capítulo 5.3 y ofrecer una solución a todas ellas.
- Descentralización de algoritmos: Las estrategias ideadas has sido implementadas bajo un enfoque centralizado, en el que existe un nodo maestro que recibe la posición de cada robot, ejecuta el algoritmo de captura y actualiza la ley de control de cada agente. La descentralización de algoritmos dotaría a los robots de total autonomía, computando cada agente la estrategia de captura de manera local, trabajando únicamente con la información procedente de sus sensores y de la comunicación que pueda tener con el resto de agentes.

8. Referencias

[1] A. Pierson, Z. Wang and M. Schwager, "Intercepting Rogue Robots: An Algorithm for Capturing Multiple Evaders With Multiple Pursuers," in IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 530-537, April 2017.

[2] H. Huang, Z. Zhou, W. Zhang, J. Ding, D. M. Stipanovic, and C. J. Tomlin, “Safe-reachable area cooperative pursuit,” IEEE Trans. Robot., 2012, submitted for publication.

[3] T. H. Chung, G. A. Hollinger, and V. Isler, “Search and pursuit-evasion in mobile robotics,” Auton Robot (2011) 31.

[4] C. Goerzen, Z. Kong, and B. Mettler, “A survey of motion planning algorithms from the perspective of autonomous uav guidance,” Journal of Intelligent and Robotic Systems, vol. 57, no. 1, pp. 65-100, 2010.

[5] J. K. Kuchar and L. C. Yang, “A review of conflict detection and resolution modeling methods,” IEEE Transactions on Intelligent Transportation Systems, vol. 1, pp. 179–189, 2000.

[6] Trudeau, Richard J. (1993). “Introduction to Graph Theory”. Dover Books.

[7] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara & Sung Nok Chiu (2000). “Spatial Tessellations – Concepts and Applications of Voronoi Diagrams”. John Wiley.

[8] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars (2008). “Computational Geometry: Algorithms and Applications”. Springer-Verlag.

[9] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second ed.). MIT Press and McGraw–Hill. pp. 595–601

[10] Murty, Katta G. (2000) “Linear programming”. John Wiley & Sons Inc.

[11] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R: Wheeler and A. Ng, “ROS: an open-source Robot Operating System,” in ICRA Workshop on Open Source Software, 2009.