UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE

TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

Gestión dinámica de conflictos en entornos con múltiples drones

Paloma Sánchez de la Torre

Junio de 2020

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE

TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

Gestión dinámica de conflictos en entornos con múltiples drones

Autor: Paloma Sánchez de la Torre

Directores: Rafael Casado González

Aurelio Bermúdez Martín

Junio de 2020

“Las cosas no se hacen siguiendo caminos distintos
para que no sean iguales, sino para que sean mejores.”
Elon Musk

Declaración de Autoría

Yo, Paloma Sánchez de la Torre con DNI 54189220-R, declaro que soy el único autor del Trabajo Fin de Grado titulado "Gestión dinámica de conflictos en entornos con múltiples drones" y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 15 de junio de 2020

Fdo.: Paloma Sánchez de la Torre

Resumen

Los drones o UAV (*Unmanned Aerial Vehicle*) son en la actualidad una tecnología muy popular, por sus múltiples posibilidades y aplicaciones. Su uso en estos últimos años ha aumentado de forma exponencial, y esto se debe principalmente al ahorro en costes, a su autonomía, y, por consiguiente, la ausencia de riesgo del factor humano. Gracias a los avances tecnológicos es posible que estos vehículos cubran un sinfín de necesidades actuales y futuras. Es por eso por lo que surge la necesidad de dotar a los UAV de un mecanismo que garantice el éxito de su vuelo, evitando posibles colisiones.

En este trabajo se propone un algoritmo para la detección y evitación de colisiones en escenarios con múltiples drones. El algoritmo propuesto (BBCA, *Bounding Box Collision Avoidance*) está basado en técnicas geométricas, buscando mejorar, por su sencillez, otros métodos presentes en la literatura.

Este algoritmo ha sido desarrollado pensando en escenarios dinámicos, donde los UAV sobrevuelan el espacio aéreo de forma autónoma y deben tomar decisiones en tiempo real. La solución funciona de forma descentralizada, puesto que en el futuro se prevén escenarios congestionados, con cientos o miles de drones compartiendo un mismo espacio aéreo, donde una solución centralizada sería muy compleja o directamente inviable.

Para la visualización y obtención de datos se ha desarrollado una herramienta de análisis a través de MATLAB, que permite ejecutar el algoritmo y graficar cada paso de este a la vez que visualizamos el vuelo de los diferentes UAV involucrados. Los datos obtenidos con esta herramienta nos han permitido el estudio y análisis del comportamiento del algoritmo para simulaciones con dos o más UAV. Además, se ha desarrollado un simulador de espacio aéreo mediante Simulink y MATLAB, que permite la ejecución de escenarios en tres dimensiones.

Agradecimientos

A mi abuela, por apoyarme en todo, hasta
en mis locuras.
A mi familia y a mis amigos, por estar en
los tiempos difíciles.
A mis tutores, Rafael y Aurelio, por el
apoyo y la confianza.

Paloma Sánchez de la Torre
Junio, 2020

Índice de Contenido

Declaración de Autoría iii
Resumen v
Agradecimientos vii

CAPÍTULO 1. Introducción 1
1.1. Motivación 1
1.2. Objetivos 3
1.3. Estructura de la memoria 4

CAPÍTULO 2. Estado del arte 5
2.1. Planteamiento general del problema 6
2.2. Técnicas de evitación de colisiones entre UAV 8
2.3. Algunas técnicas propuestas 10
2.3.1. Artificial Potencial Field (APF) 10
2.3.2. Particle Swarm Optimization (PSO) 11
2.3.3. Collision Cone (CC) 12
2.3.4. Dubins Paths 13
2.3.5. Optimal Reciprocal Collision Avoidance (ORCA) 13
2.3.6. Conclusión 15

CAPÍTULO 3. Herramientas y metodología 17
3.1. Herramientas empleadas 17
3.1.1. Matlab 17
3.1.1.1 Simulink 18
3.1.1.2 App Designer 19
3.1.2. Github 19
3.1.2.1 Git Bash 19
3.1.3. Mendeley 20
3.1.4. Microsoft Teams 20
3.1.5. Trello 21
3.2. Metodología 21

CAPÍTULO 4. Algoritmo propuesto 27
4.1. Aproximación al problema 27
4.2. Criterio para la detección de colisiones 28

4.3. Mecanismo para evitación de colisiones (BBCA) 29
4.3.1. Entradas 33
4.3.2. Velocidad válida 33
4.3.3. Obstáculo de velocidad 35
4.3.4. Velocidad óptima 39
4.4. Escenarios 43

CAPÍTULO 5. Simulador de espacio aéreo 47
5.1. Entidades involucradas en la simulación 47
5.2. Subsistema UAV 48
5.2.1. Dynamics 50
5.2.2. Pilot 51
5.2.3. Radar 53
5.3. Subsistema ATC 53
5.4. Interfaz de usuario 55

CAPÍTULO 6. Experimentos y resultados 57
6.1. Introducción 57
6.2. Estudio con dos UAV 58
6.2.1. Análisis de un escenario realista 62
6.3. Estudio con múltiples UAV 66

CAPÍTULO 7. Conclusiones y propuestas 71
7.1. Conclusiones 71
7.2. Trabajo futuro 72

Bibliografía 75

Anexos 79
A.1. Uso de la herramienta “analizador” 79
A.2. Uso de la herramienta “simulador” 80

Índice de Figuras

Figura 1. UAV 1
Figura 2. UAV de mensajería en DHL 2
Figura 3. Globo aerostático de 1849 5
Figura 4. Sistema Sense, Detect and Avoidance 7
Figura 5. Esquema de APF 10
Figura 6. Ejemplo PSO con dos mínimos diferentes [23] 11
Figura 7. Ejemplo de cono de colisión [24] 12
Figura 8. Ejemplo de Dubins Path [24] 13
Figura 9. Dos agentes virtuales (izquierda), el obstáculo de velocidad truncado VO (centro) y semiplanos libres de colisiones (derecha) [27] 14
Figura 10. Logo de MATLAB 17
Figura 11. Logo de Simulink & MATLAB 18
Figura 12. Logo de GitHub 19
Figura 13. Logo de Mendeley 20
Figura 14. Equipo de trabajo en Microsoft Teams 20
Figura 15. Tablero en Trello del TFG 21
Figura 16. Ciclo de vida Scrum 22
Figura 17. Sprints y fechas realizadas en el proyecto 23
Figura 18. Mecanismo para detectar colisiones 29
Figura 19. Pseudocódigo del algoritmo BBCA 30
Figura 20. Velocidad válida inicial en el analizador (agente 1) 33
Figura 21. Velocidad válida truncada (agente 1 con 3) 34
Figura 22. Obstáculo velocidad en simulador (agente 1 con 3) 36
Figura 23. Semiplano de obstáculo velocidad (agente 1 con 3) 37
Figura 24. Ampliación de obstáculo según vector velocidad (agente 1 con 3) 38
Figura 25. Cálculo de velocidad óptima 39
Figura 26. Ocho puntos de corte posibles 40
Figura 27. Cuatro puntos en esquinas 41
Figura 28. Elección de velocidad óptima 41
Figura 29. Elección del punto con ángulos iguales 43
Figura 30. Simulación del caso ‘ángulos iguales’ 43
Figura 31. Menú de escenarios para simular 44
Figura 32. Entradas para simulación aleatoria 44
Figura 33. Elemento ATC en el simulador 48
Figura 34. Elemento UAV en el simulador 48
Figura 35. Subsistema UAV en Simulink 49
Figura 36. Dynamics en subsistema UAV 50
Figura 37. Máquina de estado de UAV 51
Figura 38. Bloque comunicación de UAV 52
Figura 39. Esquema tratamiento de mensajes en UAV 53
Figura 40. Radar en subsistema UAV 53
Figura 41. Bloque control de ATC 54
Figura 42. Esquema tratamiento de mensajes ATC 54
Figura 43. Interfaz de usuario del simulador 55
Figura 44. Ejemplo de la ejecución del simulador 56
Figura 45. Grados en una circunferencia 59

Figura 46. Media de colisiones empleando los algoritmos direct y BBCA.....60
Figura 47. Media de la distancia recorrida para 2 UAV 60
Figura 48. Incremento en la distancia recorrida para 2 UAV 61
Figura 49. Media de tiempo consumido para 2 UAV 61
Figura 50. Incremento en el tiempo consumido para 2 UAV 62
Figura 51. Colisión entre 2 UAV sin empleo de BBCA 63
Figura 52. Simulación de escenario con 2 UAV empleando BBCA (izquierda) y la desviación realizada en la simulación (derecha).....64
Figura 53. Distancia recorrida por los UAV utilizando BBCA, direct y distancia real 65
Figura 54. Porcentaje de desviación de la ruta inicial por los UAV 65
Figura 55. Tiempo total empelado con BBCA vs direct 65
Figura 56. Media y desviación típica de conflictos para 240 escenarios con diferente cantidad de UAV (área de 5 km x 5 km).....67
Figura 57. Distancia media recorrida por los UAV con los algoritmos BBCA y direct 67
Figura 58. Incremento en la distancia recorrida por los UAV al utilizar BBCA y direct 68
Figura 59. Tiempo medio consumido por los UAV con los algoritmos BBCA y direct.....69
Figura 60. Incremento en el tiempo consumido por los UAV al utilizar BBCA y direct 69
Figura 61. Pasos para el uso de la herramienta “Analizador” 80

Índice de Tablas

Tabla 1. Algoritmo 32
Tabla 2. Acotación de velocidad admisible.....34
Tabla 3. Obstáculo de velocidad cuartiplano 36
Tabla 4. Obstáculo de velocidad semiplano.....37
Tabla 5. Extensión de obstáculo de velocidad 38
Tabla 6. Chequeo de velocidad admisible.....40
Tabla 7. Velocidad a partir del cuadro admisible 42
Tabla 8. Pseudocódigo para generar un escenario aleatorio 45
Tabla 9. Media de colisiones con múltiples UAV.....66

CAPÍTULO 1. INTRODUCCIÓN

A lo largo de esta memoria se refleja el trabajo realizado en el Trabajo Fin de Grado *"Gestión dinámica de conflictos en entornos con múltiples drones"*. En este primer capítulo se exponen los principales ítems que permitan introducir el trabajo de una forma clara y concisa.

1.1. MOTIVACIÓN

La forma en la que nos enfrentamos al mundo los humanos es cambiante. Hemos pasado de realizar tareas de forma manual a una automatización de estas. Y este avance no sería posible sin el apoyo de las nuevas tecnologías.

El mundo, tal y como funciona actualmente, necesita producir, gestionar y almacenar una enorme cantidad de información en todo momento. Esta gestión la han venido realizado los humanos, pero actualmente se emplea la tecnología como medio y apoyo para la realización de innumerables tareas, que no solo permiten una labor ágil sino más eficiente.
Pues bien, hablando de tecnologías, los protagonistas de este proyecto serán los vehículos aéreos no tripulados (UAV, *Unmanned Aerial Vehicle*), comúnmente conocidos como "drones". Son aeronaves no tripuladas que sobrevuelan el espacio aéreo, manejadas remotamente en sus inicios, pero con un alto grado de autonomía en la actualidad. Por otro lado, tendremos como protagonista al software que permitirá la gestión de la que veníamos hablando.

Figura 1. UAV

La motivación del trabajo viene dada por la previsión de que, en un futuro no muy lejano, los drones sean un elemento habitual dentro del espacio aéreo. Y esto se debe a que prestan o prestarán servicios tales como la entrega de paquetería, vigilancia, agricultura, medicina, detección de incendios, obtención de datos o incluso el transporte de pasajeros. De hecho, empresas del sector de la mensajería como Amazon o DHL ya tienen propuestas en este campo.

Figura 2. UAV de mensajería en DHL

El uso de drones en estos últimos años ha aumentado de forma exponencial, y esto se debe principalmente al ahorro en costes, a su autonomía, y, por consiguiente, la ausencia del factor humano. Los avances tecnológicos en campos como la navegación, la dinámica de vuelo, el desarrollo de sensores, las comunicaciones, etc. hacen posible su empleo para distintas aplicaciones y por lo tanto se abre un gran abanico de aplicaciones, cubriendo un sinfín de necesidades actuales y futuras.

En este Trabajo de Fin de Grado (en adelante, TFG) se plantea un escenario en el cual múltiples drones sobrevuelan un área determinada, en base a unas rutas o planes de vuelo preestablecidos. En este contexto, y al igual que sucede en el entorno aéreo “tradicional”, deberán desarrollarse mecanismos para detectar y resolver en tiempo real cualquier colisión entre los drones en vuelo, entendiendo por colisión (o conflicto) una situación en la que dos (o más) de ellos se encuentren a una distancia inferior a un umbral de seguridad.

De esta manera se pretende llegar a un escenario en el cual múltiples drones sobrevuelan un espacio aéreo, tomando decisiones en tiempo real y de forma autónoma, para unos fines concretos, y en ausencia de colisiones.

1.2. OBJETIVOS

El objetivo principal de este trabajo es realizar un primer acercamiento a los mecanismos dinámicos de detección y resolución de colisiones entre drones. En este trabajo nos centramos en la necesidad existente de la evitación de colisiones entre múltiples drones que se encuentran en el espacio aéreo, permitiendo el desplazamiento de estos de forma exitosa, sin llegar a colisionar.

Para alcanzar ese objetivo principal nos hemos planteado una serie de subobjetivos:

1. Estudiar los mecanismos existentes en la literatura para la detección y resolución de conflictos en entornos con múltiples UAV.
2. Desarrollar una herramienta de análisis que permita testear el mecanismo de evitación de colisiones propuesto en este trabajo.
3. Desarrollar una herramienta de simulación que permita probar nuevos mecanismos de evitación de colisiones en escenarios con tres dimensiones.
4. Diseñar un algoritmo sencillo de evitación de colisiones.
5. Llevar a cabo una evaluación comparativa de las prestaciones del nuevo algoritmo. En su caso, proponer mejoras al mismo.

La metodología que emplearemos para lograr nuestros objetivos será fundamentalmente la simulación. Debido al alto coste, el tiempo y riesgo humano que supondría testear los servicios o aplicaciones que ofrecen u ofrecerían los UAV, es necesario buscar una alternativa que permita un ahorro tanto en costes como en tiempo y riesgo. Para ello, nos apoyamos en el potencial ofrecido por las nuevas tecnologías, que permiten el desarrollo e implementación de un simulador/analizador de mecanismos para detección y evitación de colisiones.

1.3. ESTRUCTURA DE LA MEMORIA

Este documento se estructura en diferentes capítulos, con sus correspondientes apartados y subapartados. A continuación, se describen de forma concisa cada uno de ellos:

- **Capítulo 1:** se introduce el proyecto, abordando la motivación de este, los objetivos establecidos y cómo se estructura la memoria.
- **Capítulo 2:** consiste en el estado del arte, en el que se incluyen aquellos aspectos previos al desarrollo, como la normativa aplicable a los UAV y los algoritmos/métodos de evitación de colisiones existentes en la literatura.
- **Capítulo 3:** se mencionan las herramientas empleadas para el desarrollo del proyecto en su conjunto, así como la metodología llevada a cabo.
- **Capítulo 4:** se aborda el mecanismo para la evitación de colisiones propuesto, así como la herramienta de análisis empleada para su testeo.
- **Capítulo 5:** se corresponde con el simulador de espacio aéreo desarrollado, en el que se describe cada componente que lo conforma, así como su comportamiento.
- **Capítulo 6:** se trata del análisis y estudio de los resultados obtenidos en las pruebas realizadas con la herramienta de análisis desarrollada.
- **Capítulo 7:** se describen las conclusiones obtenidas y el trabajo futuro.

CAPÍTULO 2. ESTADO DEL ARTE

La idea de vehículo aéreo no tripulado, UAV o dron, nace hace décadas, cuando se lanzaron los primeros globos aerostáticos no tripulados al cielo cargados de bombas, para misiones de vigilancia militar, aplicaciones fotográficas, etc. Estos primeros usos se sitúan en torno a 1849 principalmente para aplicaciones militares y, desde entonces, este tipo de vehículo ha ido evolucionando hasta lo que hoy conocemos, extendiendo su uso para fines civiles.

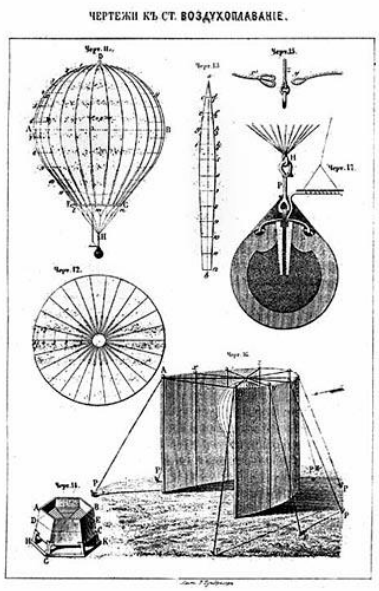

Figura 3. Globo aerostático de 1849

El aumento del empleo de estos vehículos para diversos fines ha hecho que la comunidad científica no solo se centre en la aerodinámica de los drones, en los materiales o en los chipsets de los que se componen, sino que surgen una serie de problemas o desafíos técnicos que quizás son de mayor interés, entre los que se encuentra la detección y evitación de colisiones entre estos vehículos.

Por lo tanto, nace un requisito indispensable para que estos vehículos puedan cubrir las nuevas aplicaciones a las que se enfrentan, la capacidad de evitar colisionarse.
Aunque este auge del uso de los UAV en aplicaciones civiles es relativamente nuevo, ya se

han elaborado pautas o requisitos que regulan el uso de estos en el espacio aéreo, en España lo rige el Real Decreto 1036/2017 [1]. El organismo encargado de regular su uso es la Agencia Estatal de Seguridad Aérea (AESA). Entre las exigencias existentes, caben destacar aquellas que afectan a la resolución de nuestro problema y por tanto deben tenerse en cuenta: No podrán sobrepasar los 120 metros de altura sobre el suelo, volar de día y en buenas condiciones meteorológicas y, la más importante, deberán incluir un sistema de *"Sense and Avoid"*.

Este sistema *"Sense and Avoid"* supone la incorporación de tecnología en los UAV, permitiendo la detección en vuelo de otras aeronaves, así como su entorno y, por lo tanto, realizar maniobras de evasión automáticamente [2].
A continuación, se expresa el planteamiento general del problema, así como un acercamiento a diferentes técnicas de evitación de colisiones entre UAV existentes en la literatura.

2.1. PLANTEAMIENTO GENERAL DEL PROBLEMA

Es evidente que la gestión de los posibles obstáculos que un UAV puede encontrarse durante su vuelo es crucial para el éxito de su misión. La evitación de obstáculos ha sido un problema muy estudiado por la comunidad científica, destacando su aplicación a robots móviles terrestres. En este contexto se han desarrollado diferentes algoritmos que ofrecen una solución más o menos acertada al problema planteado. Los algoritmos de evitación de obstáculos propuestos se apoyan en los diferentes sensores de los que dispone el robot, además de aplicarse, obviamente, a entornos de dos dimensiones. Entre los diferentes algoritmos se encuentran: método de campo potencial apoyado en localización GPS, basados en sensores 2-D LiDAR, mapeo global, programación de ganancia, etc.

Al contrario de lo que sucede con los robots terrestres "tradicionales", en el caso de los UAV nos encontramos con un escenario tridimensional. Algunos de los obstáculos a evitar serán de tipo estático. Es el caso de las elevaciones en el terreno, los edificios, o las zonas de vuelo no permitidas para este tipo de aeronaves. Por otro lado, también existirán obstáculos de tipo dinámico, como las aves u otros UAV con los que se comparte el espacio aéreo. Como ya se ha comentado, en este trabajo nos centraremos en la evitación de colisiones entre múltiples UAV.

La evitación de colisiones (en inglés *collision avoidance*) entre aeronaves es un problema ampliamente explorado en el contexto de la gestión del tráfico aéreo tradicional (ATM, *Air Traffic Management*), donde comúnmente se emplea la expresión "detección y resolución de conflictos" (en inglés *Conflict Detection and Resolution*, o simplemente CDR) [3][4][5]. En este contexto, una colisión (o conflicto) ocurre cuando dos o más aeronaves (tripuladas) en vuelo se sitúan a una distancia inferior a un mínimo prestablecido, conocido como separación mínima.

La separación mínima entre dos aeronaves tripuladas viene dada por la normativa que emana de las distintas autoridades con competencias en espacio aéreo, como es el caso del Documento 4444 de la ICAO (*International Civil Aviation Organization*) sobre gestión del tráfico aéreo [6]. En el caso de los UAV, en el que ya no se habla de ATM, sino de UTM (*Unmanned aircraft system Traffic Management*) [7], no hay una separación mínima estándar, aunque puedan existir regulaciones particulares que la establezcan o en algunos trabajos de investigación se asuman valores concretos.

Hay que indicar que, en el campo que nos ocupa, el de los UAV, además de las expresiones "evitación de colisiones" y "detección y resolución de conflictos" (CDR), numerosos trabajos se refieren a este mismo problema mediante la expresión en inglés "*Sense, Detect and Avoidance*" (o por sus siglas SDA, SAA o DAA). La Figura 4 muestra este tipo de sistemas, donde existen dos funciones fundamentales: una función sensorial que permite adquirir información del entorno y una función de evitación que evalúa el riesgo de una posible colisión, tomando las medidas oportunas, ya sea mediante una llamada al piloto o tomando acciones de forma autónoma.

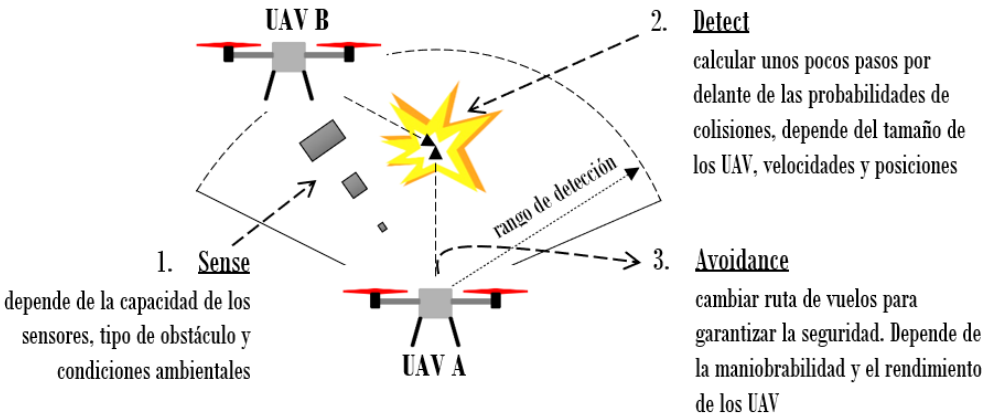

Figura 4. Sistema *Sense, Detect and Avoidance*

En escenarios como el planteado en el CAPÍTULO 1 de esta memoria, con múltiples UAV ofreciendo servicios de distinta naturaleza y compartiendo un mismo espacio aéreo, surgen gran cantidad de desafíos técnicos [8]. Uno de ellos es la planificación de la ruta (o trayectoria) que debe seguir cada uno de esos UAV. Si asumimos además un espacio aéreo relativamente congestionado, con cientos o miles de UAV, es evidente que una estrategia centralizada de planificación de rutas libres de colisiones para todos esos UAV no es viable. Por tanto, en la medida de lo posible nos centraremos en estrategias de evitación de colisiones descentralizadas (o coordinadas), de manera que los UAV puedan detectar y resolver dinámicamente este tipo de situaciones.

Sin ánimo de ser demasiado exhaustivos, en los siguientes apartados de este capítulo presentaremos primero una taxonomía general de las técnicas de evitación de colisiones para entornos con múltiples UAV existentes en la literatura, describiendo después con algo más de detalle algunas de las técnicas más relevantes o populares, indicando en cada caso su adecuación a nuestro problema. En [9] puede encontrarse una revisión completa y actualizada de los mecanismos de evitación de colisiones propuestos en este escenario.

2.2. TÉCNICAS DE EVITACIÓN DE COLISIONES ENTRE UAV

En general, podemos hablar de distintas maneras de abordar el problema de la evitación de colisiones (o CDR) entre UAV en el contexto UTM [10]. Antes de revisarlos hay que dejar claro que se trata de enfoques no excluyentes y que por tanto podrían coexistir, por ejemplo, por motivos de redundancia.

En primer lugar, como se ha comentado, existe la posibilidad de definir un conjunto de trayectorias libres de colisión para todos los UAV antes de que dé comienzo su misión. En este caso, se habla de *"pre-flight CDR methods"*. Este enfoque tiene dos grandes inconvenientes. Por un lado, nos encontramos con el más que previsible coste computacional de la solución. A esto hay que añadir el hecho de que todos estos UAV estarán operados por diferentes proveedores de servicio, lo cual dificultará todavía más una solución centralizada. Dentro del segundo enfoque nos encontramos los métodos que detectan el conflicto entre los UAV durante el vuelo. Se habla entonces de *"in-flight CDR methods"*. Dentro de esta categoría, nos encontramos con métodos centralizados y métodos distribuidos (o descentralizados). En el primer caso, el método debe recalcular un nuevo conjunto de

trayectorias de forma centralizada. Nuevamente, nos encontramos con soluciones con un alto coste computacional. Por otro lado, los métodos distribuidos suelen implicar una comunicación cooperativa entre los UAV, por ejemplo, para intercambiar su posición, velocidad, rumbo, etc. Tradicionalmente se ha considerado que estas estrategias son las más apropiadas en este contexto, y por lo tanto es el tipo de estrategias en el que nos centraremos. El tercer y último enfoque para la evitación de colisiones es lo que en la literatura se conoce como *"Sense and Avoid"*, y está basado en el empleo de técnicas de procesamiento de vídeo y otros sensores abordo del UAV.

En cuanto a técnicas concretas, la bibliografía es realmente amplia, pero destacan por su popularidad un grupo de técnicas geométricas que proceden del campo de los sistemas multi-agente y que se basan en los conceptos de cono de colisión (CC, *Collision Cone*) y obstáculo de velocidad (VO, *Velocity Obstacle*) [11][12]. A partir de estos conceptos se han desarrollado distintas propuestas (VO, RVO, HRVO…) [13][14], siendo ORCA [15] la más popular. De hecho, recientemente ORCA ha sido propuesto como técnica para CDR en aviación civil [16].

Otras técnicas de evitación de colisiones muy populares en este escenario son las basadas en fuerzas. Es el caso de los campos potenciales artificiales (APF, *Artificial Potential Field*, APF) [17], donde el destino del UAV ejerce una fuerza de atracción, pero otros UAV en su trayectoria hacia dicho destino ejercen fuerzas de repulsión.

También existe un grupo de técnicas basadas en la teoría de juegos [18]. Otras técnicas, como la optimización por enjambre de partículas (PSO, *Particle Swarm Optimization*) [19] o las basadas en colonias de hormigas (ACO, *Ant Colony Optimization*) [20] estarían dentro de lo que se conoce como métodos basados en la inteligencia del enjambre (*swarm intelligence*). Finalmente, los algoritmos de tipo evolutivo, como los genéticos, también han sido empleados para dar solución a este problema.

2.3. ALGUNAS TÉCNICAS PROPUESTAS

2.3.1. ARTIFICIAL POTENCIAL FIELD (APF)

Existen métodos basados en la fuerza, donde se modelan grupos de agentes virtuales como un sistema de partículas. Cada partícula ejerce una fuerza sobre las partículas más cercanas, empleándose las leyes de la física para determinar su movimiento.

En la Figura 5, se muestra un esquema del algoritmo APF, en el que se observa como un UAV se mueve frente a unos obstáculos, en función de distintas fuerzas hacia su objetivo.

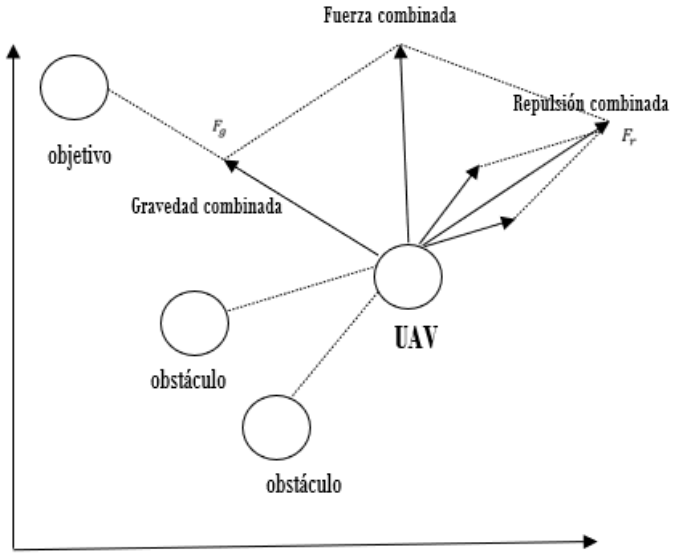

Figura 5. Esquema de APF

Estos métodos se engloban dentro de los conocidos como métodos potenciales, destinados a la planificación de trayectorias. En la literatura pueden encontrarse multitud de propuestas como la presentada en [17], un algoritmo APF optimizado.

Estos métodos se basan en la idea que Khatib introdujo en 1986, donde el agente se mueve atraído en un campo de fuerzas hacia su objetivo final y el resto de los agentes se comportan como obstáculos con fuerzas repulsivas [21].

El problema principal de estos métodos es la existencia de mínimos locales que no permiten avanzar a los agentes hacia su objetivo, además del elevado coste computación de la solución.

2.3.2. PARTICLE SWARM OPTIMIZATION (PSO)

Este método, conocido como optimización por enjambre de partículas (PSO, *Particle Swarm Optimization*), se engloba dentro de los destinados a la planificación de trayectorias. Este tipo de algoritmos se inspiran en el comportamiento de bancos de peces o bandadas de aves, donde existen un grupo de agentes conocidos como partículas que trabajan en conjunto para obtener una solución óptima al problema.

La obtención de la solución óptima se basa en un problema de optimización continuo, donde se computa la distancia de los agentes hacia su destino y esta es comunicada al resto para transmitir los mejores valores obtenidos. Este algoritmo itera a lo largo de generaciones hasta obtener la solución óptima para todo el enjambre. Se puede encontrar un amplio estudio de las aplicaciones de PSO en [22].

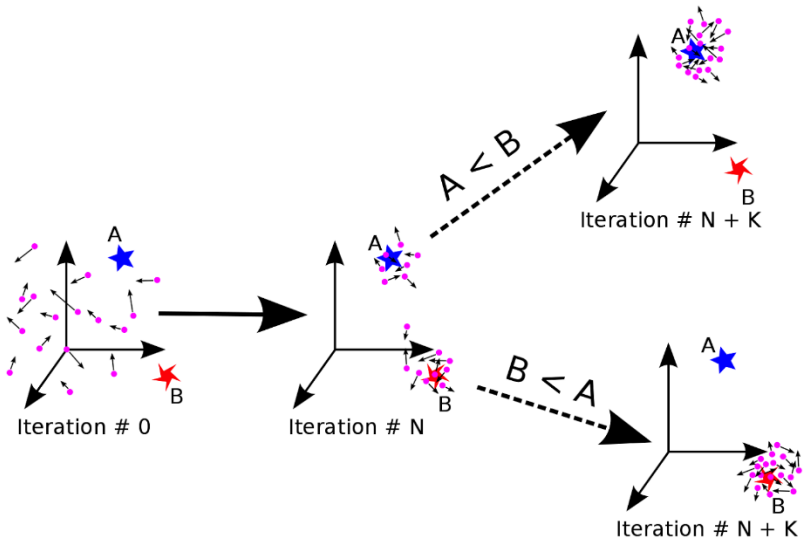

Figura 6. Ejemplo PSO con dos mínimos diferentes [23]

En la Figura 6 se observa como existen dos mínimos locales (A y B), y que al iterar N+K veces se obtienen dos soluciones, donde la solución final será el mínimo global. Debido al coste computacional que requieren los métodos basados en planificación de trayectorias, como es el algoritmo PSO, no son buenos candidatos para aplicaciones en tiempo real con múltiples agentes.

2.3.3. COLLISION CONE (CC)

El método conocido como cono de colisión (CC, *collision cone*) ha sido ampliamente utilizado para la detección y evitación de colisiones entre agentes en movimiento y con trayectorias desconocidas. Este enfoque fue inicialmente propuesto por Chakravarthy y Ghose, que establecieron la probabilidad de colisión entre dos agentes en movimiento en función de sus velocidades [12].

La mejora de este método hace posible su uso en entornos con más de dos agentes, donde se emplea una función de costo como heurística para determinar la seguridad de las posiciones de cada agente. El cono de colisión se calcula y usa para determinar si dos agentes colisionarán, siendo el sistema de dos agentes tratado como un sistema de un único agente y un obstáculo estático. En la Figura 7 se muestra cómo B es el obstáculo estático y A es el agente dinámico con velocidad relativa, $\vec{V}_{rel}$,entre A y B. Por otro lado, se prevé que ocurra una colisión cuando la distancia mínima entre A y B es menor que la distancia de separación ($d_{min} < D_{sep}$).

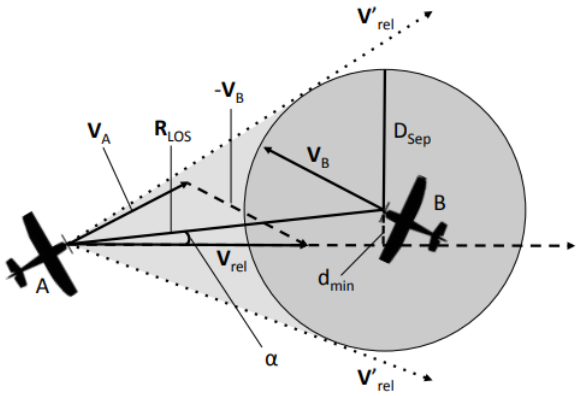

Figura 7. Ejemplo de cono de colisión [24]

Cuando la $\vec{V}_{rel}$ de A es tangencial al área circular de B, marcada como $\overrightarrow{V'}_{rel}$, los agentes se evitan con una desviación mínima a su objetivo. A los rangos de velocidades comprendidas entre los dos vectores $\overrightarrow{V'}_{rel}$ se le conoce como cono de colisión.

El principal problema del uso de este método es su eficiencia en entornos con media-baja densidad, por lo que restringe su uso a escenarios poco poblados. Además, se paga un alto coste computacional a medida que se aumenta dicho escenario. En [24] se ofrece un estudio en profundidad sobre este método.

2.3.4. DUBINS PATHS

Este método fue propuesto por Lester Dubins en 1957, que demostró como el camino más corto se puede formar uniendo líneas tangenciales con arcos circulares [25], tal y como se observa en la Figura 8.

Se demostró que es posible obtener la ruta más corta entre la posición actual de un agente y su objetivo mediante dos tres arcos circulares o mediante los dos circulares y la línea tangencial mencionada anteriormente.

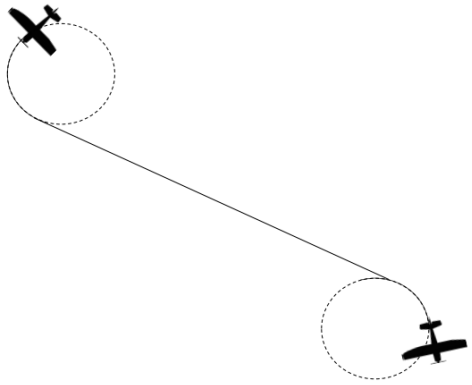

Figura 8. Ejemplo de Dubins Path [24]

Para evitar las colisiones se comprueba la intersección de las rutas de Dubins, y si dos o más caminos se cruzan, se miden las longitudes de dichas rutas hasta su intersección. Simplemente conociendo las velocidades de los agentes se puede determinar si existe colisión y modificar sus rutas lo mínimo posible para evitar el choque [26].

2.3.5. OPTIMAL RECIPROCAL COLLISION AVOIDANCE (ORCA)

Este método conocido como evitación óptima y recíproca de colisión (ORCA, *Optimal Reciprocal Collision Avoidance*), es un método basado en la velocidad. Estos métodos consisten en elegir la velocidad óptima del campo de velocidades válidas, de tal manera que el obstáculo sea evitado.

Este tipo de métodos basados en velocidad ofrecen un mejor rendimiento computacional, prevención de colisiones locales y comportamiento de los distintos agentes virtuales involucrados, frente a los métodos basados en la fuerza. Para ello emplean la velocidad actual de los agentes cercanos para extrapolar la posición futura, y bajo esa predicción se establece la nueva velocidad en función de alguna optimización.

Como se ha comentado, ORCA es un método basado en velocidad, pero con la diferencia de la inclusión de reciprocidad, donde cada agente virtual intenta evitar colisionarse con el otro. De esta manera se observan movimientos más suaves, aunque pueden aparecer otros problemas como cuellos de botella, bloqueos, etc.

Este algoritmo parte de la hipótesis de que cada UAV es independiente y no se comunica con otro, no existe intercambio de información. Cada UAV se encuentra continuamente en un ciclo de detección y actuación. Por lo tanto, cada acción que tome se hará en base a observaciones locales. Básicamente se extrapolan las velocidades observadas, con el fin de estimar las posiciones futuras de los obstáculos. A partir de dicha información se genera el llamado "obstáculo de velocidad" (VO, *Velocity Obstacle*), que incluye aquellas velocidades prohibidas para un agente respecto de otro (Figura 9, centro). Una vez establecido el obstáculo de velocidad, se generan semiplanos (Figura 9, derecha) que permiten definir aquellas velocidades libres de colisiones, donde el agente seleccionará la velocidad preferida (velocidad hacia su objetivo) y no la más cercana a su velocidad actual, puesto que en muchas situaciones se verá muy alejado de su objetivo por evitar una colisión.

Para seleccionar la velocidad óptima de entre las velocidades libres de colisiones, ORCA emplea la técnica de optimización de programación lineal.

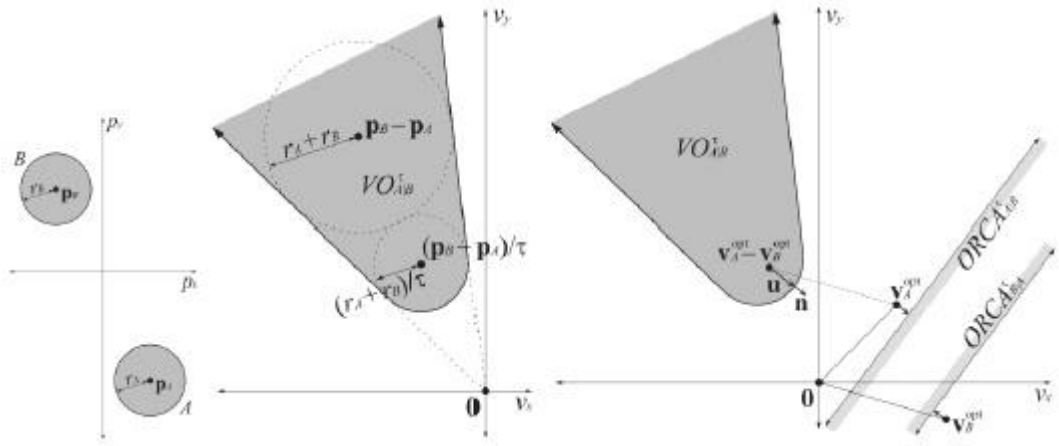

Figura 9. Dos agentes virtuales (izquierda), el obstáculo de velocidad truncado VO (centro) y semiplanos libres de colisiones (derecha) [27]

ORCA garantiza la navegación libre de colisiones cuando los escenarios no son densos [15], de lo contrario, puede no existir una velocidad que cumpla con las restricciones. Cuando no existe una velocidad que cumpla con los requisitos para evitar una colisión, ORCA

selecciona la velocidad más segura posible, que penetre mínimamente dichas restricciones. Por ello, cuando el escenario es denso, se emplea programación lineal tridimensional para el cálculo de la velocidad donde no se puede garantizar que esté libre de colisiones [27].
Los principales problemas de este algoritmo se resumen en: coste computacional medio-alto debido al cálculo de velocidades mediante programación lineal y no poder garantizar un escenario libre de colisiones.

2.3.6. CONCLUSIÓN

Tras el estudio y análisis de los diferentes métodos o algoritmos propuestos en la literatura, se ha determinado que los métodos de mayor interés son los basados en velocidades, por su buena relación entre coste computacional y eficiencia. Además, el algoritmo debe enfocarse en observaciones locales y evitar el empleo de intercambio de información entre los UAV involucrados.

CAPÍTULO 3. HERRAMIENTAS Y METODOLOGÍA

A lo largo de este capítulo se detallan tanto las herramientas como la metodología empleada para el desarrollo de este trabajo.

3.1. HERRAMIENTAS EMPLEADAS

Para poder ofrecer una solución de calidad, que se ajuste a las necesidades que plantea el problema, es necesario realizar un estudio previo de las herramientas que nos permitan desarrollar e implementar dicha solución.

A continuación, se describen las herramientas empleadas en este trabajo, las cuales han sido seleccionadas de forma precisa, teniendo en cuenta principalmente la curva de aprendizaje y la adecuación a las necesidades o requisitos existentes en el proyecto.

3.1.1. MATLAB

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es una plataforma enfocada en resolver problemas científicos y de ingeniería. Ofrece un entorno de desarrollo integrado (IDE, *Integrated Development Environment*), con un lenguaje de programación propio basado en matrices.

Figura 10. Logo de MATLAB

En el momento de iniciar este trabajo, MATLAB se encontraba en su versión R2019b, y, por lo tanto, esta ha sido la versión empleada.

La plataforma de MATLAB es empleada por millones de científicos e ingenieros para analizar y diseñar sistemas y productos. Ofrece un gran número de prestaciones, entre las que destacan: aplicaciones para ajustar curvas, clasificar datos, analizar señales, ajustar

sistemas de control, gráficas para visualizar datos, herramientas para crear aplicaciones con interfaces, herramienta para depuración, etc. [28].

MATLAB es la herramienta principal seleccionada para este proyecto, debido a su potencial, y, en especial, por ofrecer las características necesarias para el desarrollo e implementación de la solución. Entre dichas características se encuentran: la generación de *scripts* que permiten graficar y analizar los datos obtenidos, la implementación de una interfaz que permita la simulación de cada escenario, la representación de modelos para la implementación del simulador mediante diagramas, etc.

El entorno de programación base de MATLAB ha sido empleado para implementar el analizador que se muestra en el CAPÍTULO 4 de esta memoria. A continuación, se exponen dos herramientas que nos ofrece MATLAB y que han sido empleadas para el desarrollo del simulador de espacio aéreo que se detallará a lo largo del CAPÍTULO 5.

3.1.1.1 SIMULINK

Simulink es una herramienta de simulación de modelos de sistemas dinámicos, una *toolbox* especial de MATLAB, que permite un cierto grado de abstracción sobre los fenómenos físicos involucrados. Simulink proporciona un entorno gráfico sobre el que construir nuestros modelos como diagramas de bloques, de una forma muy sencilla.

Figura 11. Logo de Simulink & MATLAB

Simulink dispone de un gran catálogo de librerías de bloques, sobre la que destaca el paquete Stateflow, que permite la simulación de máquinas de estados. Al emplear MATLAB de forma conjunta con Simulink, se está combinando la programación textual y gráfica para el diseño del sistema en un entorno de simulación [29]. El código de MATLAB es incluido en un bloque de Simulink o gráfico de Stateflow.

En definitiva, Simulink servirá como base fundamental para modelar, simular y analizar el sistema dinámico que constituye el escenario de aplicación del presente trabajo. Simulink posibilita la ejecución de miles de simulaciones en paralelo, lo que nos permite el análisis y visualización de los datos en poco tiempo.

3.1.1.2 APP DESIGNER

App Designer es otra de las herramientas interesantes que nos proporciona MATLAB, ya que nos ofrece un IDE sobre el que diseñar interfaces de usuario. Ofrece librerías de componentes que facilitan el desarrollo de forma fácil e intuitiva de la interfaz de una aplicación, como, por ejemplo, botones, listas desplegables, gráficas, etc. [3].

Esta herramienta permitirá desarrollar una pequeña aplicación sobre la que poder ejecutar el simulador aéreo, para así, realizar simulaciones en escenarios personalizados de una forma rápida e intuitiva.

3.1.2. GITHUB

GitHub es un sistema de gestión de proyectos y control de versiones de código. Es una herramienta que nos permite mantener un repositorio en el que almacenar nuestro proyecto, con todas las ventajas que ello conlleva.

Figura 12. Logo de GitHub

El uso de un sistema que nos permita gestionar el proyecto, así como las versiones de este, nos ofrece varias ventajas, entre las que destacan: mantener una copia de seguridad, seguimiento de cambios, trabajo colaborativo, seguimiento de errores, etc.

Por lo tanto, este proyecto será mantenido a través de GitHub en el repositorio almacenado en https://github.com/PalomaSanx/UAVsimulation_TFG.

3.1.2.1 GIT BASH

Es una herramienta que permite manipular y gestionar todo el proceso a realizar con el proyecto. A través de una consola de comandos y de forma rápida, podemos ir actualizando el repositorio [30].

3.1.3. MENDELEY

Mendeley es un gestor de referencias bibliográficas, capaz de extraer de forma automática los metadatos de las distintas webs de interés, así como aquella documentación objeto de ser referenciada en un documento.

Figura 13. Logo de Mendeley

Durante la elaboración de la presente memoria esta herramienta ha sido de gran ayuda en esta labor.

3.1.4. MICROSOFT TEAMS

Microsoft Teams es una plataforma destinada a la comunicación y colaboración que incluye un lugar de trabajo donde poder mantener reuniones remotas, almacenar archivos, colaboración en archivos e integración de aplicaciones, entre otras [31].

Figura 14. Equipo de trabajo en Microsoft Teams

La herramienta permite la creación de equipos de trabajo, tal y como se observa en la Figura 14, desde los que se pueden gestionar fácilmente el conjunto de acciones y archivos destinados a un grupo de personas que trabajan hacia un objetivo común.

Esta herramienta ha sido utilizada a lo largo de tomo el TFG como método de comunicación diario, en el cual almacenar de forma ordenada los archivos del proyecto, así como la colaboración de archivos.

3.1.5. TRELLO

Trello es una herramienta para administración de proyectos. Permite la gestión de tareas a través del registro de actividades con tarjetas virtuales. Esta herramienta dispone de elementos como *checklist*, etiquetas, etc., que permiten gestionar de forma sencilla cada una de las tarjetas incluidas en el proyecto [32].

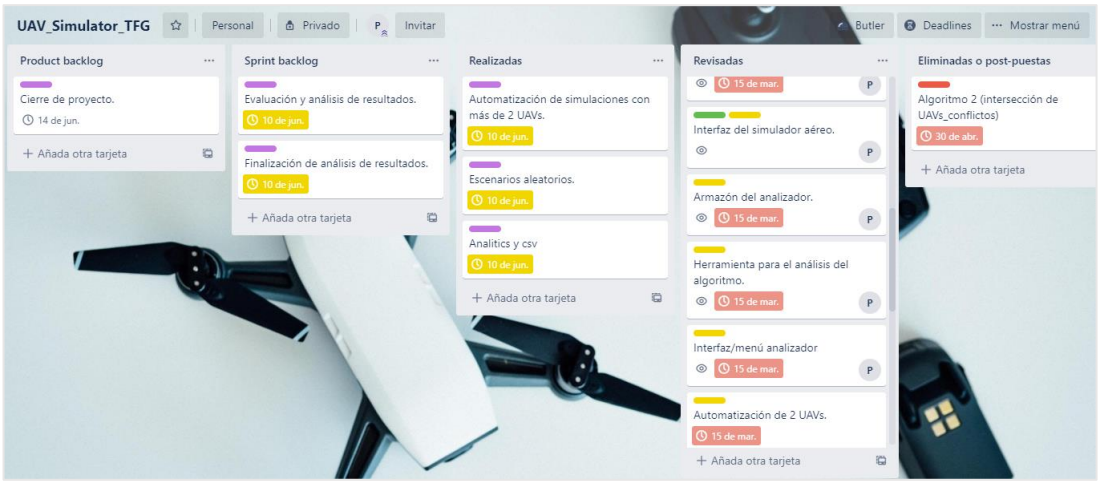

Figura 15. Tablero en Trello del TFG

En la Figura 15 se muestra un ejemplo del uso de esta herramienta en este trabajo. Esta herramienta se ha empleado para incluir de forma simple y rápida cada una de las tareas a lo largo del ciclo de vida del proyecto.

3.2. METODOLOGÍA

Este trabajo se ha llevado a cabo siguiendo una metodología ágil, la metodología Scrum [33]. Una metodología ágil se refiere a métodos que implican un desarrollo iterativo e incremental, donde la incertidumbre es elevada y, por ende, un gran volumen de cambios. Además, el trabajo es realizado de forma colaborativa en equipos multidisciplinarios y autoorganizativos, en el que las decisiones tomadas se realizan de forma conjunta y a corto plazo. Se destinan principalmente a equipos pequeños o medianos, entre 3 y 9 personas. Además, las fases de desarrollo se solapan, dejando atrás la planificación secuencial o en cascada.

Scrum es una metodología ágil para la gestión de proyectos. Trabaja con *sprints* o iteraciones, donde el trabajo suele planificarse cada semana. Al final de cada *sprint* se revisa el trabajo validado de la semana anterior. Tras ello se planifican y priorizan las actividades del próximo *sprint*. La Figura 16 refleja el ciclo de vida de la metodología Scrum, en la que puede observarse los roles que intervienen en el proceso, los eventos y artefactos.

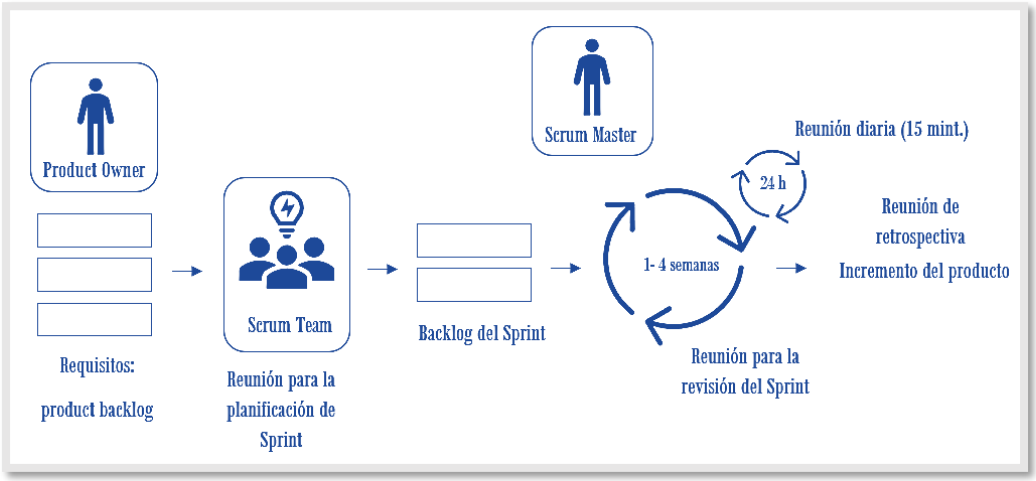

Figura 16. Ciclo de vida Scrum

Se diferencian tres roles:

- *Product Owner*: se encarga de alcanzar y cumplir con todos los requisitos establecidos, desempeñando las tareas necesarias para ello. Es la persona que conoce cómo que se debe desarrollar y la secuencia a seguir en el desarrollo.
- *Scrum Master*: es el responsable de que se sigan las prácticas y valores en el modelo de Scrum. Se encarga de generar y mantener el *Product Backlog* con las distintas tareas, el *Sprint Backlog* con las tareas de cada *sprint* y el propio *sprint*.
- Equipo de desarrollo: se configura por miembros que se encargan de convertir las necesidades del *Product Owner* en un conjunto de funcionalidades del producto final. Son los encargados de desarrollar el sistema.

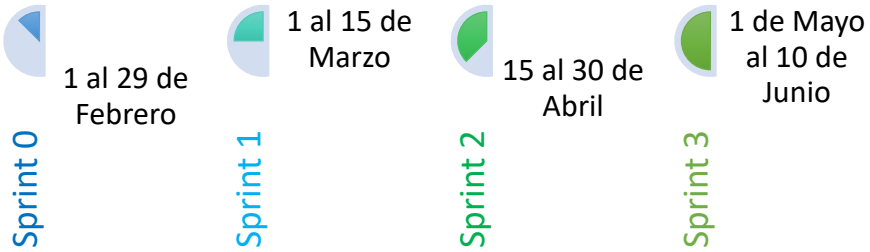

Figura 17. Sprints y fechas realizadas en el proyecto

El equipo de trabajo en este proyecto está compuesto por 3 personas:

- Paloma Sánchez de la Torre tiene el rol de *Scrum Master*.
- El trabajo de *Product Owner* es compartido entre Rafael Casado González, Aurelio Bermúdez Martín y Paloma Sánchez de la Torre.
- Paloma Sánchez de la Torre actúa como equipo de desarrollo.

A lo largo del proyecto se han llevado a cabo un total de cuatro *sprints* de una duración aproximada de un mes. Cada uno de ellos incluye la propia planificación del *sprint* (*Planning Meeting)*, los *scrum* diarios (*Daily Scrum)*, la revisión del *sprint* y su posterior retrospectiva (*Sprint Retrospective)*. La Figura 17 muestra el total y las fechas de los *sprints* realizados. A continuación, se describe cada uno de los *sprints* realizados:

- Sprint 0: en este *sprint* no se parte de cero, puesto que el trabajo ha pasado por una fase previa en la que se establece una propuesta inicial que debe ser validada por la Comisión de TFG para la realización de este. Por lo tanto, en este *sprint* se define el alcance del proyecto, se marcan los objetivos principales, una aproximación a lo que sería y no sería, y se establece el cómo.

  Al finalizar el *sprint* se completó la realización del simulador de espacio aéreo en el que poder ejecutar escenarios en tres dimensiones.
- Sprint 1: con anterioridad se fijó la necesidad de disponer de una herramienta que permita analizar posteriormente el algoritmo propuesto en el TFG. Por lo tanto, es en este *sprint* donde se desarrolla el armazón inicial del analizador, la generación de funciones necesarias como la detección de conflictos y la automatización en la recogida, almacenaje y procesamiento de los datos obtenidos.

Al finalizar el *sprint* se obtiene una herramienta que permite analizar el comportamiento del algoritmo de evitación de colisiones, quedando pendiente la inclusión de dicho mecanismo.

- Sprint 2: las tareas a realizar en este *sprint* consisten en el análisis del comportamiento de otros algoritmos propuestos en la literatura, para obtener una aproximación al problema. En este *sprint* se desarrolla el algoritmo BBCA, además de otras propuestas que finalmente quedan aplazadas para centrarnos en el primer algoritmo.

  Al finalizar el *sprint* se dispone de una herramienta de análisis que incluye el mecanismo de evitación de colisiones y permite la simulación de diferentes escenarios. Además, todos los datos son almacenados de forma ordenada para su posterior análisis.
- Sprint 3: este *sprint* se centra en el análisis y estudio del comportamiento del algoritmo BBCA propuesto. Además, se incluyen *scripts* que permiten la generación automática de un banco de pruebas.

  Al finalizar el *sprint* se obtiene un banco de pruebas extenso sobre el que poder realizar una evaluación final.

La filosofía de trabajo seguida incluye el uso de herramientas colaborativas (como las ya comentadas), reuniones presenciales, reuniones remotas y aplicaciones para la gestión de cambios y versiones, como ya se venía adelantando. Las reuniones mantenidas a lo largo del proyecto se diferencian en:

- Reuniones presenciales: estas han sido llevadas a cabo en la primera mitad de los *sprints*, ya que la pandemia sufrida por el Covid-19 imposibilitó la realización de estas en el resto.
- Reuniones remotas: se llevan a cabo desde la segunda mitad de los *sprints*.
- Reuniones diarias o *Daily Scrum*: estas se llevan a cabo diariamente sin superar los 15 minutos de duración. En ellas se sincronizan las actividades y se establece un plan para las siguientes 24 horas. En estas reuniones se determina en que está trabajando cada miembro, en qué trabajará después y si existe algún problema.
- Reuniones de *sprint* o *Sprint Review*: en estas reuniones se revisa el trabajo completado en el *sprint*. A lo largo del proyecto se han realizado cuatro reuniones,

tres remotas y una presencial, que coinciden con la finalización de cada *sprint*.

- Reuniones de retrospectiva o *Sprint Retrospective:* en estas reuniones se busca una mejora del *sprint* y la planificación del siguiente. A lo largo del proyecto se han mantenido cuatro reuniones de retrospectiva que permiten en la mayoría de los casos obtener una mejora de cara a un trabajo futuro. Estas reuniones se han realizado en su mayoría remotamente y coincidiendo con la finalización de cada *sprint*.

CAPÍTULO 4. ALGORITMO PROPUESTO

En este capítulo se detalla la solución que proponemos para el problema de la gestión de colisiones entre UAV. Para ello se describe la aproximación al problema, el criterio establecido para la detección de colisiones y el mecanismo para la evitación de estos. Por último, se describe la forma en la que pueden generarse o se generan los distintos escenarios.

Esta solución se ha llevado a cabo con el apoyo de la herramienta MATLAB, en la que se ha desarrollado un analizador que permite visualizar la simulación de una forma sencilla, mostrando la trayectoria que sigue cada UAV, la información relevante sobre el algoritmo (velocidades válidas, obstáculos de velocidad, velocidad objetivo, velocidad óptima, etc.), así como la comunicación de mensajes en caso de colisión u otro aspecto relevante. A lo largo del capítulo se irá mostrando la funcionalidad de esta herramienta y en el Anexo A.1 se incluyen una serie de instrucción para su utilización.

4.1. APROXIMACIÓN AL PROBLEMA

Como punto de partida inicial se han definido una serie de criterios que son indispensables para abordar el problema de la detección y evitación de colisiones.

Como se ha indicado en el CAPÍTULO 2, un conflicto o colisión es la pérdida de separación mínima entre dos UAV, es decir, se viola un criterio que define lo que es indeseable. El objetivo principal del mecanismo de detección de colisiones es comunicar un conflicto que va a ocurrir en un futuro. Sería análogo a un sistema que alertase sobre la proximidad a un obstáculo terrestre o advirtiese de otros riesgos como, por ejemplo, el clima. Puede ser necesario tener en cuenta de cara a detectar un conflicto:

- Información del estado actual de los UAV: como por ejemplo sus posiciones y velocidades.
- Un modelo de trayectorias dinámicas, para poder proyectar los estados en el futuro

y predecir si existirá un conflicto. Un ejemplo sería emplear información del estado actual de los UAV para determinar la línea recta de su vector velocidad para un instante concreto.

Por otro lado, será necesario definir un mecanismo de evitación de colisión, es decir, una acción que debe tomar una entidad involucrada en una posible colisión tras la detección de un conflicto, permitiendo que dicho conflicto no llegue a ocurrir. Nuestro mecanismo de evitación empleará la misma información usada en la detección del conflicto para determinar la modificación de la velocidad de cada entidad, así como la actualización de los parámetros involucrados, como podrían ser las posiciones.

El mecanismo se centra en la detección y evitación de colisiones en el plano 2D, asumiendo un escenario en el que todos los UAV sobrevuelan el espacio aéreo a una misma altura, que suponemos reservada a este tipo de aparatos. Obviamente, para evitar colisiones en un espacio tridimensional habría que definir un conjunto de acciones distinto al considerado.

4.2. CRITERIO PARA LA DETECCIÓN DE COLISIONES

Como ya se había comentado con anterioridad, se entiende por colisión la situación en la que dos o más agentes superan un umbral de seguridad preestablecido. Por simplicidad, cada UAV es modelado mediante una circunferencia cuyo radio coincide con dicho margen de seguridad.

En nuestro caso, se establece que una colisión se producirá cuando la distancia existente entre dos UAV supere al doble de dicho radio. Siendo *i* el identificador del UAV que lanza la ejecución de la función de detección de colisión y *j* todos los agentes involucrados. En primer lugar se establece la distancia entre *i* y *j*, asumiendo que sus posiciones en el espacio son P_i*=(x1, y1)* y P_j*=(x2, y2)*.

$$distance(P_i, P_j) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Después se realiza la comprobación, quedando la función de la siguiente manera:

$$distance(P_i, P_j) < 2 * radioUAV \implies collision(i, j)$$

En la Figura 18 puede observarse de forma clara cuando se produce dicha colisión.

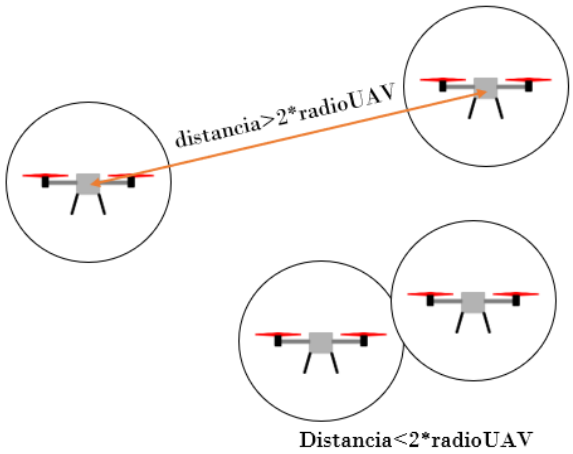

Figura 18. Mecanismo para detectar colisiones

4.3. MECANISMO PARA EVITACIÓN DE COLISIONES (BBCA)

El mecanismo que permite evitar las colisiones entre los distintos UAV que sobrevuelan el espacio aéreo con un determinado fin, como podría ser el de entrega de paquetería a diferentes destinos, es una de las partes más importantes e interesantes de este TFG

Nuestro objetivo es proponer un mecanismo de menor complejidad en cuanto a coste computacional e implementación se refiere respecto a otros algoritmos ya implementados, como el popular algoritmo ORCA, que, como se ha visto en el CAPÍTULO 2, emplea programación lineal para el cálculo de la velocidad óptima de un agente según unas restricciones.

A continuación, se describe la propuesta del algoritmo planteada en este trabajo para la evitación de colisiones, la cual recibe el nombre de "*Bounding Box Collision Avoidance*" (BBCA). Para comenzar, la Figura 19 presenta el pseudocódigo de este mecanismo.

```
input numUAVs: number of virtual agents
input t_step: time step, t_end: end time
input: vel_max: maximun velocity, UAVrad: UAV radio
input UAVpos: UAV position, UAVtarget: UAV target

for t = 1:t_step:t_end
    for i = 1:numUAVs
        get velocity of UAV i
        get green box of velocity valid
        for j = 1:numUAVs
            if i == j
                continue
            end
        get velocity obstacle
        get red box of velocity obstacle(a quarter of the plane)
        move red box according to relative speed
        get distance to red box of side N,S,E and W
        choose better side
        move red box according to better side (horizontal or vertical half plane)
        expand red box to half speed vector
        update green box
        end for
        if empty speed box (green box)
            UAVvelF = half in green box
        else
            UAVvelF = get better velocity in green box
        end if
    end for
    UAVvel = UAVvelF
    for i = 1:numUAVs
        % move the UAV i
        UAVpos(i) = UAVpos + UAVvel(i) * t_step
    end for
    detection collision
end for
```

Figura 19. Pseudocódigo del algoritmo BBCA

Este código es ejecutado por cada UAV en cada paso de simulación. De forma resumida, el mecanismo recibe las posiciones iniciales, las posiciones finales, la velocidad máxima y el radio de todos los UAV en vuelo. El algoritmo supone que cada UAV conoce de antemano esta información. Por otro lado, previamente se han establecido otros parámetros, como el área del escenario, el tiempo entre pasos de simulación, el intervalo para recalcular la navegación o el tiempo de estabilización de la velocidad.

En el bucle externo, por cada paso en la simulación el algoritmo debe escoger dentro del rango de velocidades válidas del UAV la mejor, en un proceso iterativo de refinamiento en el que va haciendo uso de la información disponible sobre el resto de UAV. Esto lo realiza generando y actualizando dos elementos:

- Velocidades válidas (mostradas en el analizador como una caja de color verde). Inicialmente su tamaño se corresponde con la velocidad máxima para el UAV y posteriormente es truncada (en ese proceso iterativo) en función de la restricción del obstáculo de velocidad que proporciona cada uno de los UAV.
- Obstáculo de velocidad (mostrado como una caja de color rojo). Este supone la restricción de velocidad para un UAV frente a otro. Permite truncar la velocidad válida de forma que ambos UAV no colisionen.

Cuando se dispone de la velocidad válida truncada en función del obstáculo de velocidad, se calcula la velocidad óptima en base al "método de los doce puntos". Este método se detalla más adelante, pero básicamente selecciona el punto óptimo entre doce posibles, para posteriormente obtener la velocidad "futura" o velocidad en el paso siguiente. La velocidad "futura" es empleada para el cálculo de la aceleración, que se usa para actualizar la velocidad actual de los UAV, permitiendo así su movimiento.

Además, en cada paso de simulación, se comprueba si existen conflictos (empleando el criterio definido en el apartado 4.2) y qué UAV ha llegado a su destino. Por lo tanto, el mecanismo realiza principalmente estas tres funciones, en cada paso de simulación y para cada UAV:

- Obtener la caja de velocidad válida.
- Obtener la caja del obstáculo de velocidad.
- Obtener la velocidad óptima.

A continuación, se expresa formalmente en la Tabla 1 el algoritmo BBCA:

Sea $\mathbf{p} = [p_x\ p_y] \in \mathbb{P}$ la posición de un UAV
Sea $\mathbf{v} = [v_x\ v_y] \in \mathbb{V}$ la velocidad de un UAV
Sea $\mathbf{w} = [w_x\ w_y] \in \mathbb{P}$ una posición de destino
Sea el conjunto $\mathbb{A} = \{\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n\}$, donde cada elemento $\mathbf{a} = [\mathbf{p}\ r\ \mathbf{v}\ \mathbf{w}] \in \mathbb{A}$ es el vector de estado en el que se encuentra un UAV
Sea r el radio de seguridad común a todos los UAVs
Sea $v_{\max}$ la velocidad crucero común a todos los UAVs
Sea τ el intervalo de ejecución del algoritmo de navegación de los UAVs

Tabla 1. Algoritmo

	$\mathbf{v} = \text{BBCA}(\mathbf{a}_1, r, v_{\max}, \tau)$
01	$\left[\mathbf{p}_1\ \mathbf{v}_1\ \mathbf{w}_1\right] = \mathbf{a}_1$
02	$[G_{\text{N}}\ G_{\text{S}}\ G_{\text{E}}\ G_{\text{W}}] = [v_{\max}\ -v_{\max}\ v_{\max}\ -v_{\max}]$
03	$\forall \mathbf{a}_2 \in \mathbb{A}, \mathbf{a}_1 \neq \mathbf{a}_2$ do
04	$\left[\mathbf{p}_2\ \mathbf{v}_2 \sim\right] = \mathbf{a}_2$
05	$\mathbf{O} = \dfrac{[\mathbf{p}_2 - \mathbf{p}_1 \quad 3r]}{\tau}$
06	$\mathbf{R} = \text{QPlane}(\mathbf{O})$
07	$\mathbf{R} \mathrel{+}= \mathbf{v}_2 \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$
08	$\mathbf{L} = \left(\mathbf{v}_1 \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} - \mathbf{R}\right) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
09	$b = \max(\mathbf{L})$
10	$\mathbf{R} = \text{SPlane}(b, \mathbf{L}, \mathbf{R})$
11	$\mathbf{R} = \text{Extend}(b, \mathbf{L}, \mathbf{R})$
12	$\mathbf{G} = \text{Truncate}(b, \mathbf{L}, \mathbf{G}, \mathbf{R})$
13	if $\text{Empty}(\mathbf{G})$
14	$\mathbf{v} = \left[\dfrac{G_{\text{W}} + G_{\text{E}}}{2} \quad \dfrac{G_{\text{S}} + G_{\text{N}}}{2}\right]$
15	else
16	$\mathbf{v} = \text{VSelect}(\mathbf{G})$
17	endif
18	enddo

En resumen, descomponemos el UAV en sus componentes (1). A continuación, se inicializa el cuadro verde de velocidades válidas (2). Para cada UAV en las proximidades (3), enumeramos sus componentes (4), generamos el obstáculo de velocidad circular que este representa (5). Traducimos dicho obstáculo a un cuartiplano (medio semiplano) (7-20) que denominamos cuadro rojo (6), y lo desplazamos en función de su propia velocidad (7). A continuación, obtenemos la distancia del vector velocidad a los respectivos lados del cuartiplano (8) donde un valor negativo indicaría la cantidad de penetración en el mismo. Elegimos mejor opción el lado del cuartiplano que está más alejado del vector velocidad (9); y ampliamos el cuartiplano para obtener un semiplano (vertical u horizontal) en función del lado elegido (10).

En los siguientes subapartados se describen con mayor detalle algunos elementos del algoritmo para la evitación de colisiones, como sus entradas, y cada una de las funciones que lo componen.

4.3.1. ENTRADAS

Existen un conjunto de *inputs* o entradas, empleadas por el algoritmo, y que permiten la simulación de diferentes escenarios. Estas entradas se dividen principalmente en:

- Número de agentes (UAV)
- Posición de inicio de cada agente
- Velocidad máxima de cada agente
- Posición de destino de cada agente
- Radio de seguridad de cada agente

4.3.2. VELOCIDAD VÁLIDA

Como se puede observar en la Figura 20, el conjunto de velocidades válidas para un UAV se visualiza en el analizador por medio de una caja verde que dispone de cuatro lados: Norte (N), Sur (S), Este (E) y Oeste (W).

Inicialmente se configura por medio de la velocidad máxima y se actualiza en función de las restricciones introducidas. Estas restricciones se refieren al obstáculo de velocidad generado para cada UAV y que será abordado en el apartado siguiente.

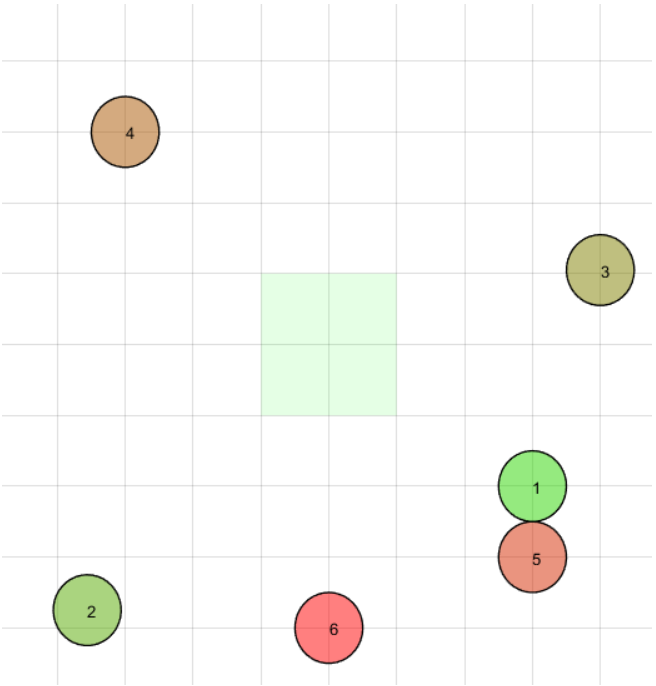

Figura 20. Velocidad válida inicial en el analizador (agente 1)

Una vez construido el obstáculo velocidad (caja roja), las velocidades válidas o permitidas (caja verde) son actualizadas, tal y como se observa en la Figura 21 siguiendo el procedimiento descrito en la Tabla 2. La actualización de estas consiste en su truncamiento a partir del obstáculo velocidad, es decir, dados dos conjuntos R = {valores obstáculo velocidad} y G = {valores de velocidades válidas}:

$$G = G - R = \{x \in G \; y \; x \notin R\}$$

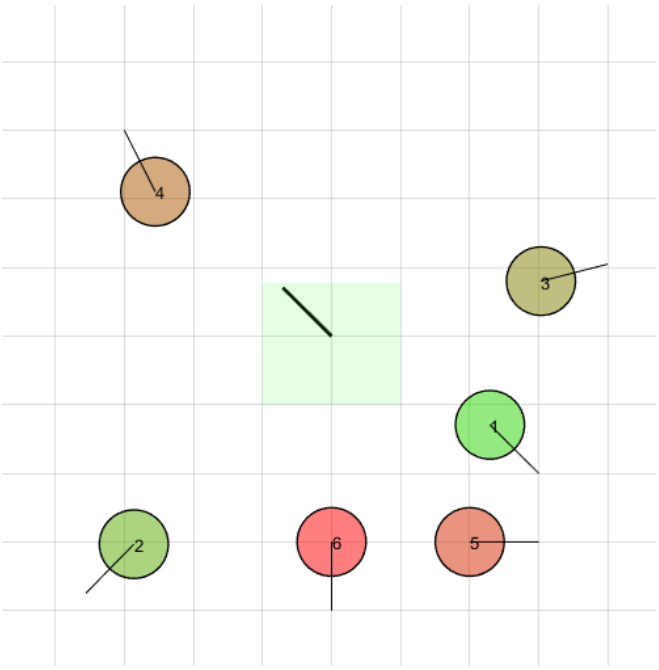

Figura 21. Velocidad válida truncada (agente 1 con 3)

Tabla 2. Acotación de velocidad admisible

	$\mathbf{G}' = \text{Truncate}(b, \mathbf{L}, \mathbf{G}, \mathbf{R})$
01	$[L_\text{N}\ L_\text{S}\ L_\text{E}\ L_\text{W}] = \mathbf{L}$
02	$[G_\text{N}\ G_\text{S}\ G_\text{E}\ G_\text{W}] = \mathbf{G}$
03	$[R_\text{N}\ R_\text{S}\ R_\text{E}\ R_\text{W}] = \mathbf{R}$
04	`switch` b
05	`case` L_N
06	$G'_\text{S} = \max(G_\text{S}, R_\text{N})$
07	`case` L_S
08	$G'_\text{N} = \min(G_\text{N}, R_\text{S})$
09	`case` L_E
10	$G'_\text{W} = \max(G_\text{W}, R_\text{E})$
11	`case` L_W
12	$G'_\text{E} = \min(G_\text{E}, R_\text{W})$
13	`endswitch`

Una vez actualizado/truncado el conjunto de velocidades válidas, se comprueban dos posibles casos:

- La caja de velocidades válidas se encuentra vacía: esto ocurre cuando la caja se ha invertido o sus lados coinciden y el lado S es mayor o igual que el lado N o el lado W es mayor o igual al lado E. En este caso, se establece la velocidad futura (para el siguiente paso de simulación) como el centro de dichas velocidades válidas.
- La caja de velocidades válidas no está vacía: en este caso, la velocidad futura se establece mediante el cálculo de la velocidad óptima, entre las posibles (por medio del procedimiento descrito en el apartado 4.3.4).

4.3.3. OBSTÁCULO DE VELOCIDAD

De la misma forma en la que se ha modelado la velocidad permitida o velocidad válida, se ha implementado el obstáculo de velocidad, pero en este caso mediante una caja roja con lados Norte (N), Sur (S), Este (E) y Oeste (W).

En primera instancia se genera el obstáculo de velocidad circular, en función de la posición y el radio de seguridad del agente vecino. Después, la caja roja se sitúa sobre el obstáculo circular, formando un cuarto del plano en función del obstáculo circular y limitando así la región de velocidades no permitidas con dicho agente. En la Tabla 3 se detalla este proceso y en la Figura 22 se representa gráficamente, donde el obstáculo circular se corresponde con la circunferencia roja y el obstáculo velocidad es situado tangencialmente a este formando “un cuarto del plano”.

Tabla 3. Obstáculo de velocidad cuartiplano

	$\mathbf{R} = \text{QPlane}(\mathbf{O})$
01	$[O_x\ O_y\ O_r] = \mathbf{O}$
02	`if` $O_y < 0$
03	$R_{\text{N}} = O_y + O_r$
04	$R_{\text{S}} = -\infty$
05	`else`
06	$R_{\text{N}} = +\infty$
07	$R_{\text{S}} = O_y - O_r$
08	`endif`
09	`if` $O_x < 0$
10	$R_{\text{E}} = O_x + O_r$
11	$R_{\text{W}} = -\infty$
12	`else`
13	$R_{\text{E}} = +\infty$
14	$R_{\text{W}} = O_x - O_r$
15	`endif`
16	$\mathbf{R} = [R_{\text{N}}\ R_{\text{S}}\ R_{\text{E}}\ R_{\text{W}}]$

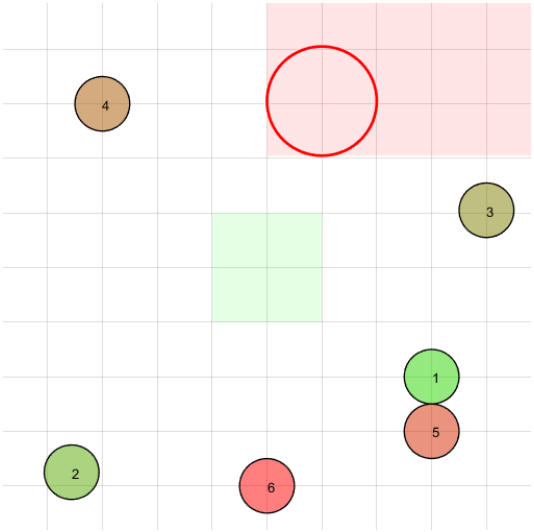

Figura 22. Obstáculo velocidad en simulador (agente 1 con 3)

El siguiente paso consiste en el desplazamiento del obstáculo velocidad (caja roja) en función de la velocidad relativa al agente vecino. Tras ello, se calculan las distancias desde el centro a los lados de dicha caja roja, para obtener el lado óptimo (máxima distancia) y así desplazar el obstáculo velocidad. Este último desplazamiento se realizará de forma horizontal o vertical, generando un semiplano. En la Tabla 4 se especifica mediante pseudocódigo el proceso para construir este obstáculo de velocidad semiplano y en la Figura 23 se representa claramente este desplazamiento, en este caso de forma horizontal.

Tabla 4. Obstáculo de velocidad semiplano

	$\mathbf{R}' = \text{SPlane}(b, \mathbf{L}, \mathbf{R})$
01	$[L_{\text{N}}\ L_{\text{S}}\ L_{\text{E}}\ L_{\text{W}}] = \mathbf{L}$
02	switch b
03	case L_{N}
04	$\mathbf{R}' = \mathbf{R} + [0 \quad -\infty \quad +\infty \quad -\infty]$
05	case L_{S}
06	$\mathbf{R}' = \mathbf{R} + [+\infty \quad 0 \quad +\infty \quad -\infty]$
07	case L_{E}
08	$\mathbf{R}' = \mathbf{R} + [+\infty \quad -\infty \quad 0 \quad -\infty]$
09	case L_{W}
10	$\mathbf{R}' = \mathbf{R} + [+\infty \quad -\infty \quad +\infty \quad 0]$
11	endswitch

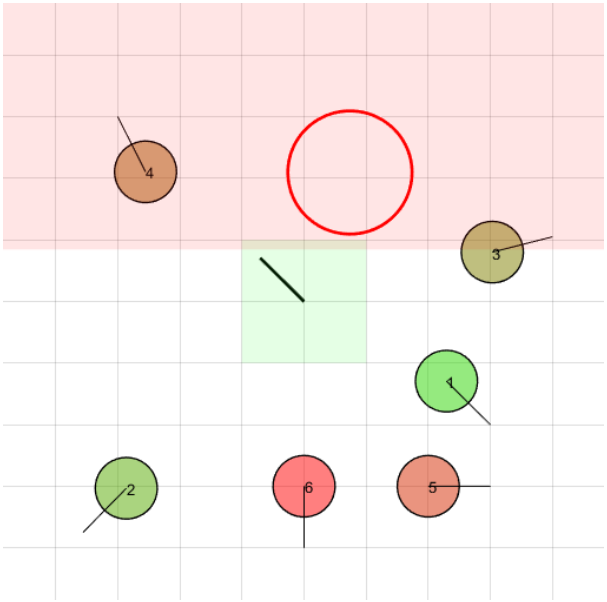

Figura 23. Semiplano de obstáculo velocidad (agente 1 con 3)

El tercer paso se basa en ampliar dicha restricción (obstáculo velocidad) la mitad del vector velocidad del agente con la dirección óptima calculada anteriormente, tal y como se expresa en la Tabla 5. En la Figura 24 se observa dicha ampliación, aunque es casi inapreciable.

Tabla 5. Extensión de obstáculo de velocidad

	$\mathbf{R}' = \text{Extend}(b, \mathbf{L}, \mathbf{v}, \mathbf{R})$
01	$[L_\text{N}\ L_\text{S}\ L_\text{E}\ L_\text{W}] = \mathbf{L}$
02	$[v_\text{x}\ v_\text{y}] = \mathbf{v}$
03	$[R_\text{N}\ R_\text{S}\ R_\text{E}\ R_\text{W}] = \mathbf{R}$
04	`switch` b
05	`case` L_N
06	$R'_\text{N} = \frac{R_\text{N} + v_\text{y}}{2}$
07	`case` L_S
08	$R'_\text{S} = \frac{R_\text{S} + v_\text{y}}{2}$
09	`case` L_E
10	$R'_\text{E} = \frac{R_\text{E} + v_\text{x}}{2}$
11	`case` L_W
12	$R'_\text{W} = \frac{R_\text{W} + v_\text{x}}{2}$
13	`endswitch`

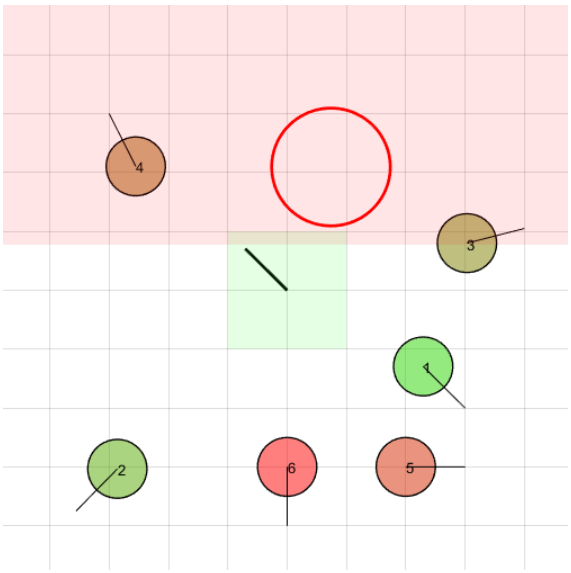

Figura 24. Ampliación de obstáculo según vector velocidad (agente 1 con 3)

Por último, la velocidad válida para un UAV (caja verde) es truncada en base al obstáculo velocidad (caja roja). De esta manera el obstáculo velocidad resultante es el que se empleará para calcular la velocidad óptima, tal y como se había comentado en el apartado anterior.

4.3.4. VELOCIDAD ÓPTIMA

La elección de la velocidad óptima se realiza en función de las velocidades válidas disponibles en cada instante de tiempo, vistas en el apartado “Velocidad válida”.
Para la visualización de este proceso, se incluye en el analizador la velocidad válida circular, la caja de velocidades válidas (caja verde) y la velocidad directa del agente a su objetivo/destino, tal y como se observa en la Figura 25.

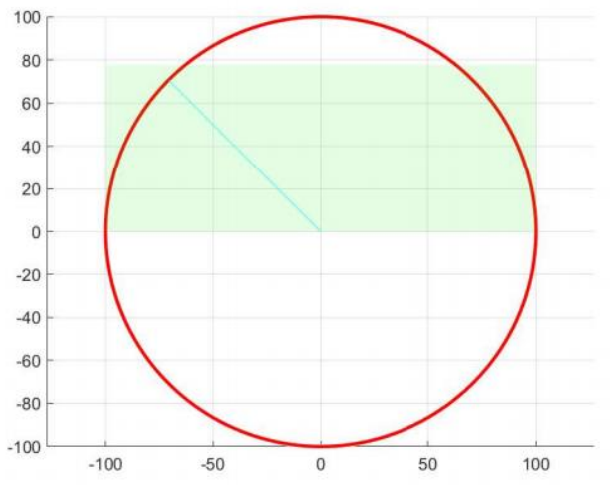

Figura 25. Cálculo de velocidad óptima

La función para el cálculo de la velocidad óptima cuenta con dos posibles casos, que son tratados de forma independiente:

- La velocidad directa al objetivo se encuentra dentro de las velocidades válidas. En este caso, se establece como mejor opción dicha velocidad. Este caso se corresponde con el mostrado en la Figura 25.
- La velocidad directa al objetivo no está dentro de las velocidades válidas. Este caso se trata mediante un método que hemos denominado “método de los doce puntos”.

El método de los “doce puntos” consiste en tratar ocho puntos posibles (velocidades posibles que puede tomar el UAV en el paso de simulación siguiente) de corte entre la caja verde de velocidades válidas y la velocidad válida circular. Es decir, solo serán considerados aquellos puntos (velocidades) que se encuentren dentro de la caja verde de velocidades válidas y esto se realiza por medio de la función representada en la Tabla 6. En la Figura 26 se muestra un ejemplo de los ocho puntos posibles, que son considerados por encontrarse dentro de la caja verde de velocidades válidas, cortando la velocidad válida circular.

Tabla 6. Chequeo de velocidad admisible

	$b = \text{VInside}(\mathbf{v}, \mathbf{G})$
01	$[v_x\ v_y] = \mathbf{v}$
02	$[G_N\ G_S\ G_E\ G_W] = \mathbf{G}$
03	`if` $G_S \leq v_y \leq G_N$ `and` $G_W \leq v_x \leq G_E$
04	$b = \text{true}$
05	`else`
06	$b = \text{false}$
07	`endif`

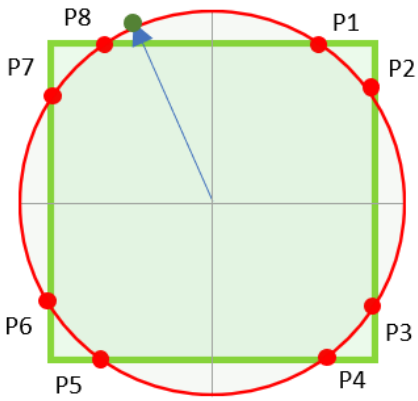

Figura 26. Ocho puntos de corte posibles

Por otro lado, se consideran otros cuatro puntos posibles, pertenecientes a las esquinas de la caja verde de velocidades válidas. Dichas esquinas serán consideradas si se encuentran dentro de la velocidad válida circular, que se encuentra dentro del rango de la velocidad máxima permitida (ver Tabla 6). Los doce puntos son calculados en cada paso de simulación para cada uno de los UAV una vez ha sido truncada su caja de velocidad válida con respecto a los obstáculos de velocidad, de forma que se termine seleccionando el punto óptimo de entre los posibles.

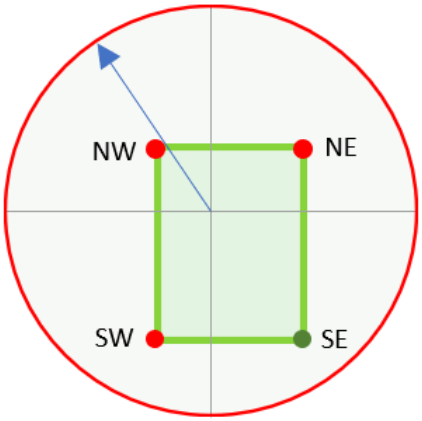

Figura 27. Cuatro puntos en esquinas

Calculados los doce puntos posibles, se determina cuál de ellos es el óptimo. Esto se realiza comparando el ángulo de dichos puntos con respecto a la velocidad directa del agente a su objetivo. De esta manera, se dispone de la velocidad óptima para el paso siguiente en la simulación de los UAV, como aquel de los puntos cuyo ángulo sea menor y permita un mayor desplazamiento. Este proceso se calcula por medio de la función presentada en la Tabla 7.

En la Figura 28 se muestra un ejemplo de la elección del punto óptimo a partir de la velocidad válida o permitida. Como se puede observar la elección se corresponde con aquel punto que forme un ángulo menor con respecto a la velocidad directa del UAV a su objetivo (línea azul). Además, este punto ha sido elegido porque permite el mayor desplazamiento del UAV en cuanto a distancia, cumpliendo la condición de ángulo menor.

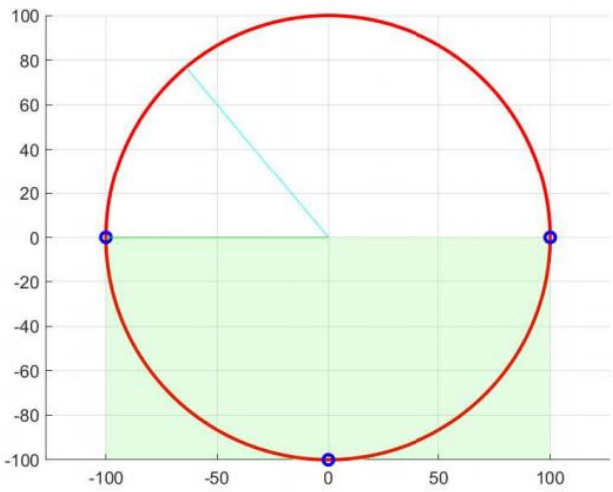

Figura 28. Elección de velocidad óptima

Tabla 7.
Velocidad a partir del cuadro admisible

	$\mathbf{v} = \mathrm{VSelect}(\mathbf{p}, \mathbf{w}, \mathbf{G}, \mathbf{R}, v_{\max}, \tau)$
01	$\mathbf{v} = \mathbf{0}$
02	`if` $\mathbf{p} = \mathbf{w}$
03	`return`
04	`endif`
05	$\mathbf{m} = \mathbf{w} - \mathbf{p}$
06	$\mathbf{v}_\mathrm{d} = \dfrac{\mathbf{m}}{\lvert\mathbf{m}\rvert} \cdot \min\left(\dfrac{\lvert\mathbf{m}\rvert}{\tau}, v_{\max}\right)$
07	`if` $\mathrm{VInside}(\mathbf{v}_\mathrm{d}, \mathbf{G})$
08	$\mathbf{v} = \mathbf{v}_\mathrm{d}$
09	`return`
10	`endif`
11	$[G_\mathrm{N}\ G_\mathrm{S}\ G_\mathrm{E}\ G_\mathrm{W}] = \mathbf{G}$
12	$\mathbf{v}_{\{\mathrm{N\ S}\}\{\pm\}} = \left[\pm\sqrt{v_{\max}^2 + G_{\{\mathrm{N\ S}\}}^2} \quad G_{\{\mathrm{N\ S}\}}\right]$
13	$\mathbf{v}_{\{\mathrm{E\ W}\}\{\pm\}} = \left[G_{\{\mathrm{E\ W}\}} \quad \pm\sqrt{v_{\max}^2 + G_{\{\mathrm{E\ W}\}}^2}\right]$
14	`if not` $\mathrm{VInside}\left(\mathbf{v}_{\{\mathrm{N\ S\ E\ W}\}\{\pm\}}, \mathbf{G}\right)$
15	$\mathbf{v}_{\{\mathrm{N\ S\ E\ W}\}\{\pm\}} = \mathbf{0}$
16	`endif`
17	$\mathbf{v}_{\{\mathrm{NE\ SE\ SW\ NW}\}} = [G_{\{\mathrm{E\ E\ W\ W}\}} \quad G_{\{\mathrm{N\ S\ S\ N}\}}]$
18	`if` $\left\lvert\mathbf{v}_{\{\mathrm{NE\ SE\ SW\ NW}\}}\right\rvert > v_{\max}$
19	$\mathbf{v}_{\{\mathrm{NE\ SE\ SW\ NW}\}} = \mathbf{0}$
20	`endif`
21	$\forall \mathrm{L} \in \{\mathrm{N\ S\ E\ W}\}\{\pm\} \cup \{\mathrm{NE\ SE\ SW\ NW}\}$ `do`
22	$\alpha_\mathrm{L} = \mathrm{acos}\left(\dfrac{\mathbf{v}_\mathrm{d} \cdot \mathbf{v}_\mathrm{L}}{\lvert\mathbf{v}_d\rvert\lvert\mathbf{v}_L\rvert}\right)$
23	`if` $\lvert\mathbf{v}_L\rvert > \lvert\mathbf{v}\rvert$ `or` $(\lvert\mathbf{v}_L\rvert = \lvert\mathbf{v}\rvert$ `and` $\alpha_L < \alpha)$ `)`
24	$\mathbf{v} = \mathbf{v}_L$
25	$\alpha = \alpha_L$
26	`enddo`

Por último, se distingue un caso especial, en el que existen dos puntos con ángulos iguales, cuya elección debe establecerse siempre con un mismo criterio (por ejemplo, “elegir siempre el punto en sentido de las agujas del reloj”).

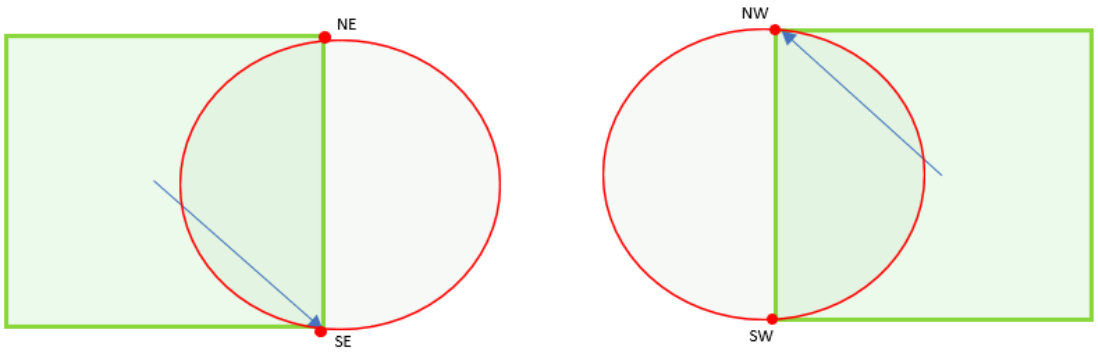

Figura 29. Elección del punto con ángulos iguales

El punto seleccionado será el primero que se encuentre en el sentido de las agujas del reloj en función de la dirección hacia la que se dirige el agente o UAV. Como se muestra en la Figura 29, a la izquierda se encuentra un UAV que avanza de forma positiva en el eje de abscisas y decide seleccionar el punto 'SE' bajo este criterio. A la derecha se muestra la misma situación, pero inversa. De esta manera si dos UAV se encuentran en el camino, se consiguen evitar, tal y como se muestra en la Figura 30.

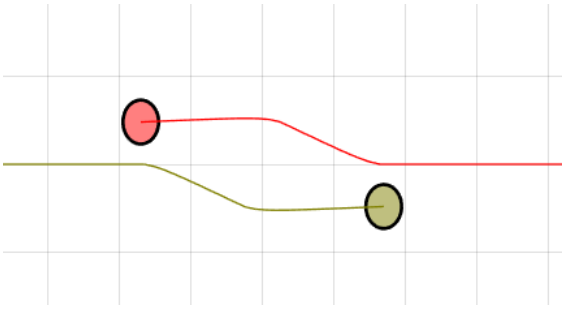

Figura 30. Simulación del caso 'ángulos iguales'

4.4. ESCENARIOS

Para poder analizar el comportamiento de un algoritmo, se ha de generar previamente un conjunto de escenarios. Cada escenario está compuesto principalmente por una serie de parámetros. En concreto, se necesita el número de UAV considerados y, para cada uno de ellos:

- Posición inicial
- Posición objetivo
- Radio de seguridad
- Velocidad máxima

Por ello, se ha incluido en el analizador un menú con diferentes escenarios, de forma que cada simulación sea fácilmente ejecutable.

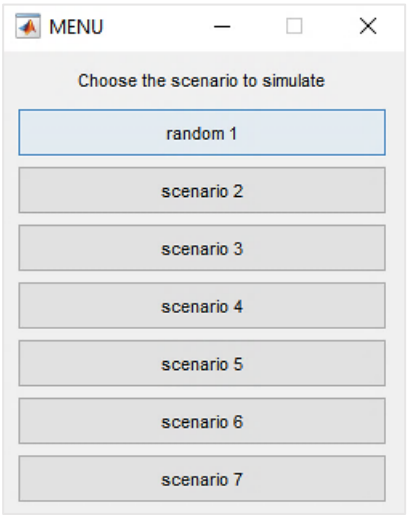

Figura 31. Menú de escenarios para simular

Además de poder ejecutar los escenarios creados, se ha incluido una opción que permita ingresar parámetros, como el número de agentes virtuales involucrados o el área (ver Figura 32). Esta opción genera el resto de los parámetros de forma aleatoria, permitiendo ejecutar escenarios con alta densidad de agentes.

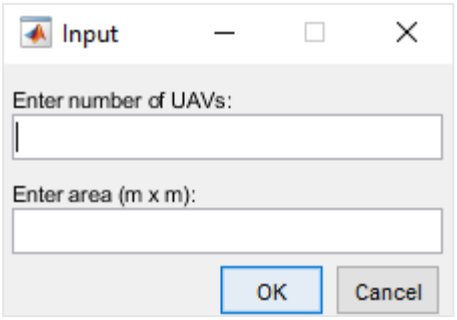

Figura 32. Entradas para simulación aleatoria

La función para generar estos escenarios aleatorios establece un valor a la velocidad máxima y al radio de los UAV, de forma aleatoria bajo unos límites establecidos. Posteriormente genera las posiciones iniciales a una distancia mínima de separación entre ellas, previamente establecida. De la misma forma en la que se generan las posiciones iniciales, se generan las posiciones objetivo de los UAV. Además, se exige que las rutas de los UAV recorran la mitad del área del escenario.

Tabla 8. Pseudocódigo para generar un escenario aleatorio

$[UAVposInit, UAVtarget, vel\_max, UAVrad] =$ randScen(numUAV, $area$)

```
vel_max = número aleatorio ([limMin,limMax]);
UAVrad= número aleatorio ([limMin,limMax]);
distMin = UAVrad*4;

For i=1:numUAV
    UAVok = false;
        While ~UAVok
            UAVposInit(i) = posicion aleatoria según area;
            UAVtarget(i) = posicion aleatoria según area;
          Route = norm(UAVposInit(i)-UAVtarget(i));
        If route < area/2
            continue;
        End
          UAVok = true;
          For j=1:i-1
              If posiciones < distMin
                  UAVok = false;
                  break;
              End
          End
        End
End
```

CAPÍTULO 5. SIMULADOR DE ESPACIO AÉREO

A lo largo de este capítulo se describe el desarrollo de un simulador que permite testear el comportamiento en vuelo de un conjunto de UAV mientras ofrecen una serie de servicios o aplicaciones. Este simulador es la alternativa perfecta al alto coste, tiempo y riesgo humano que supondría el testeo en un entorno real.

En definitiva, se dispondrá de un simulador que permita ejecutar un escenario en tres dimensiones en el que múltiples UAV autónomos sobrevuelan el espacio aéreo en tiempo real, realizando una serie de entregas a los destinos establecidos, y todo ello libre de colisiones gracias a los mecanismos dotados para ello. Estos mecanismos se basan en la propuesta planteada en este trabajo, y que se integra en el simulador de una forma independiente, permitiendo así la flexibilidad y modularidad del sistema.

El sistema se ha desarrollado empleando App Designer de MATLAB para el desarrollo de la interfaz principal que permita la personalización de cada simulación, Simulink para el desarrollo del simulador mediante modelos y una combinación de ambas para la obtención y análisis de los datos. En el Anexo A.2 de esta memoria se muestran los pasos para la utilización del simulador aéreo.

En primer lugar, introduciremos las entidades (o subsistemas) simulados y la forma en la que estas intercambian información. Después detallaremos la forma en la que se ha modelado cada entidad. Finalmente, se presenta brevemente la interfaz del simulador.

5.1. ENTIDADES INVOLUCRADAS EN LA SIMULACIÓN

Entendemos por entidad a aquellos elementos materiales o humanos que se encuentran involucrados en el proceso de la simulación. Como es de esperar, el principal elemento es el UAV. Por otro lado, la gestión del tráfico la lleva a cabo el controlador de tráfico aéreo (ATC, *Air Traffic Control*).

El ATC tiene como labor principal la emisión de cada uno de los destinos o posiciones a los que deberá acudir un UAV cuando este se encuentre disponible para ello. Esto se consigue gracias a la comunicación entre ambos mediante paso de mensajes.

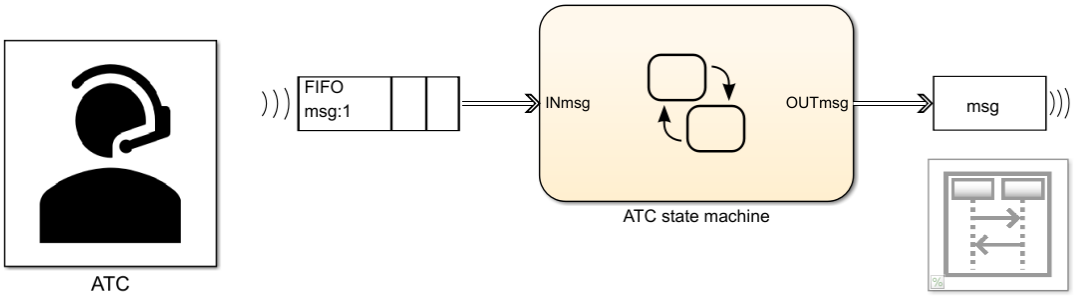

Figura 33. Elemento ATC en el simulador

Ambos elementos emiten y reciben mensajes de forma continua. La recepción de los mensajes se realiza con el bloque '*Queue'* que facilita Simulink y mediante el método FIFO (*First-In First-Out*) se atiende y procesa cada mensaje almacenado.

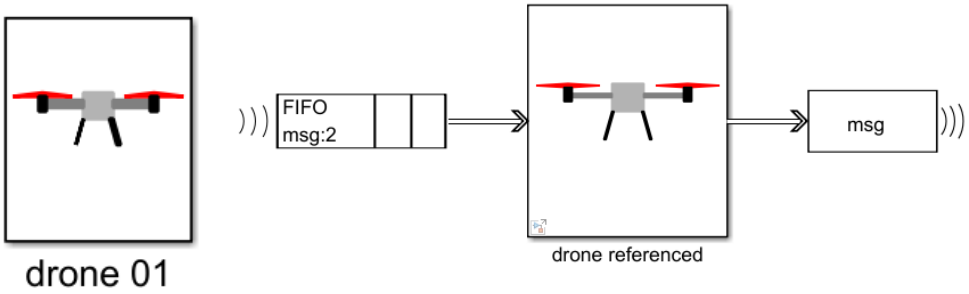

Figura 34. Elemento UAV en el simulador

A continuación, se detalla tanto la composición como el funcionamiento de ambos subsistemas para el desarrollo del simulador. Estos subsistemas han sido implementados mediante el entorno gráfico de Simulink, que permite construir el modelo a simular mediante bloques.

5.2. SUBSISTEMA UAV

Como ya se venía adelantando, cada UAV emitirá y recibirá sistemáticamente mensajes para informar sobre su estado actual, así como obtener información del resto de UAV o recibir órdenes de entrega por parte del ATC.

Los UAV disponen de la unidad de medición inercial (IMU, *Inertial Measurement Unit*) que permite obtener información del dispositivo tal como la velocidad o su posición.

Los elementos principales que componen este subsistema se pueden observar a través de la Figura 35:

- Dynamics: controla el movimiento del UAV, a partir de una velocidad y posición deseadas.
- Pilot: contiene la lógica mediante máquina de estados, así como el procesamiento de mensajes.
- Radar: permite la visualización de la simulación en forma de gráfico 3D.

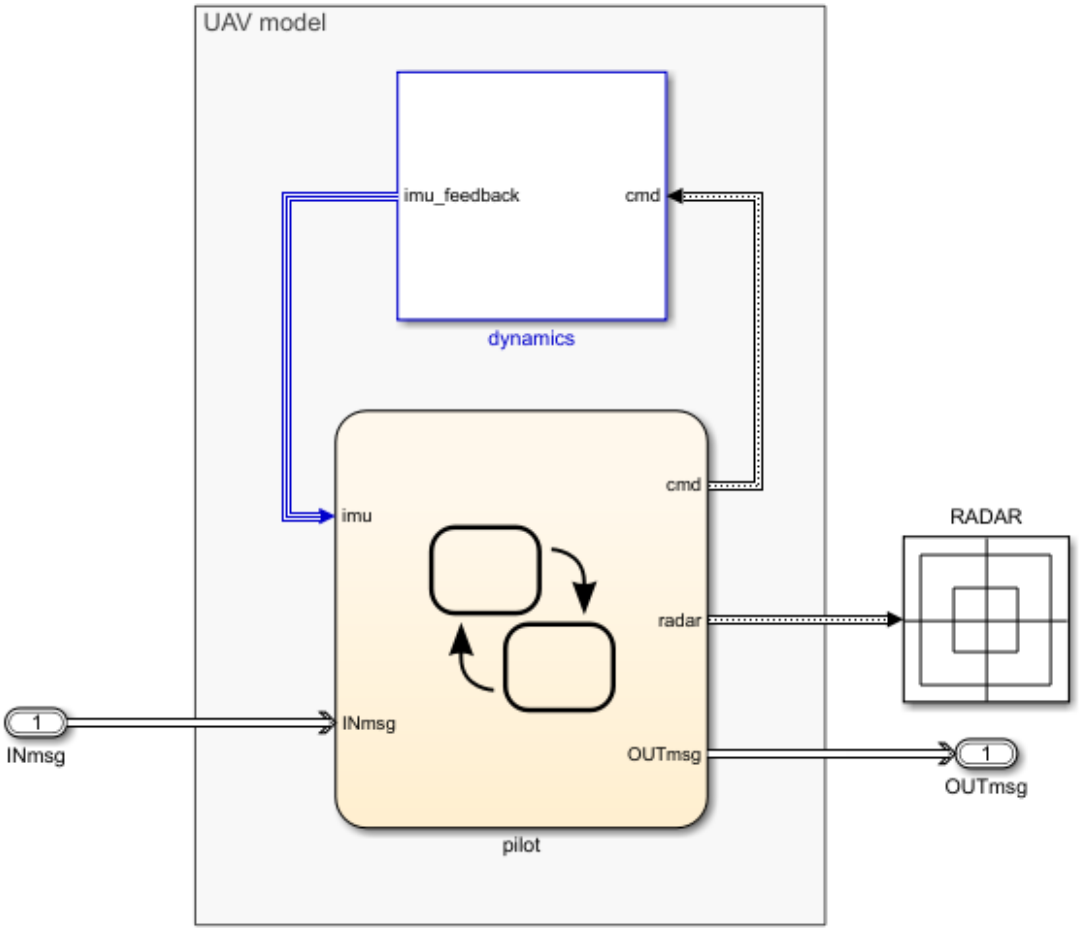

Figura 35. Subsistema UAV en Simulink

5.2.1. DYNAMICS

Como se puede observar en la Figura 36, este componente dispone de dos puertos, uno de entrada y otro de salida. El puerto de entrada '*cmd'* introduce a través de un bus la velocidad que se demanda en cada uno de los ejes (*x, y, z*) y las posiciones iniciales para cada uno de los UAV. Las velocidades demandadas pasan por un actuador que permite obtener la posición actual.

Por último, toda la información introducida a través del puerto '*cmd*' pasa por un integrador para emitir la señal de entrada con respecto al tiempo y así obtener la posición actual (*x, y, z*). La posición se emite a través del puerto de salida '*imu_feedback*'.

Como se puede observar en la Figura 35 ambos puertos (*cmd* e *imu_feedback*) son utilizados en '*pilot'* que es el corazón del subsistema UAV, y contiene la lógica del mismo.

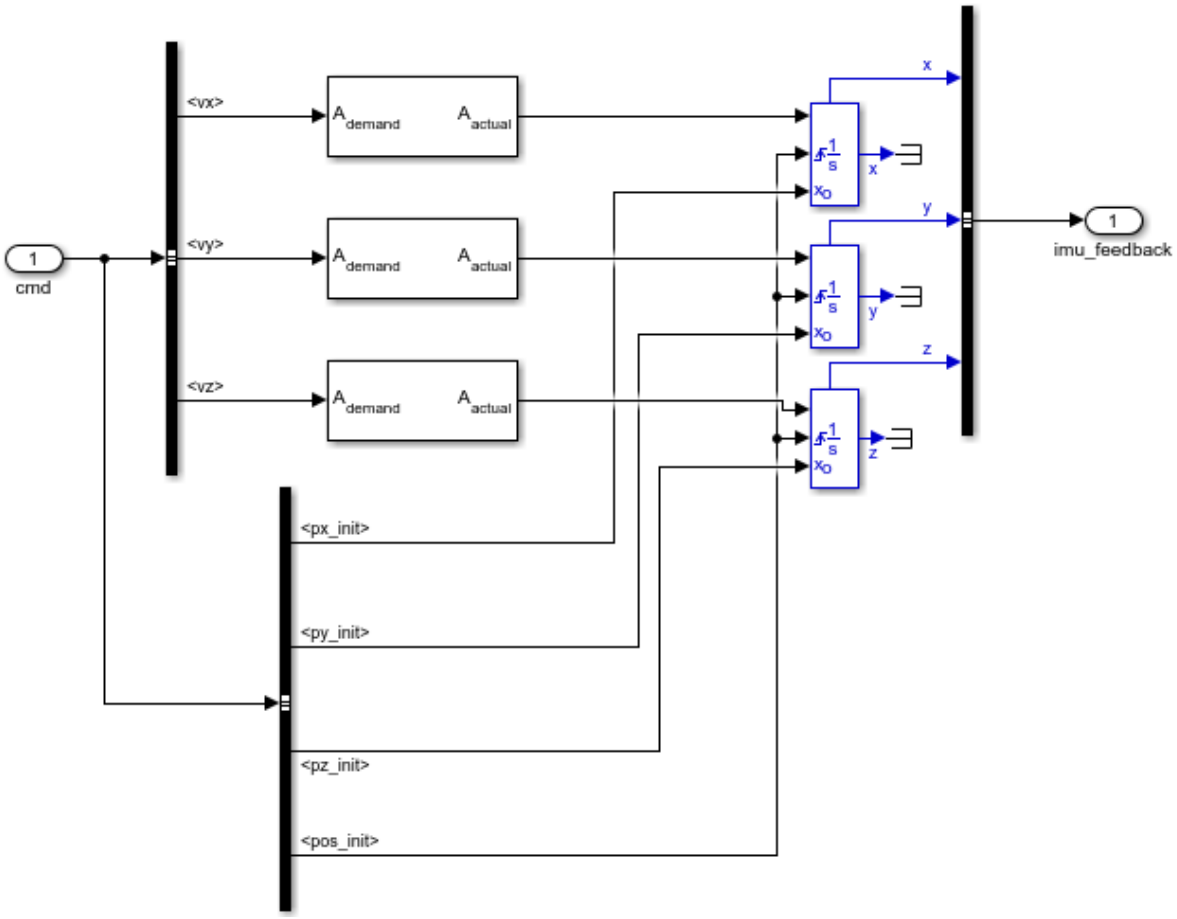

Figura 36. Dynamics en subsistema UAV

5.2.2. PILOT

La implementación de la lógica principal de decisión del UAV se encuentra en '*Pilot'*. Esta se ha desarrollado mediante el Stateflow de Simulink, que permite la simulación de máquinas de estados. Se diferencian dos bloques, uno representa la navegación (ver Figura 37) y el otro la comunicación (ver Figura 38) de cada UAV.

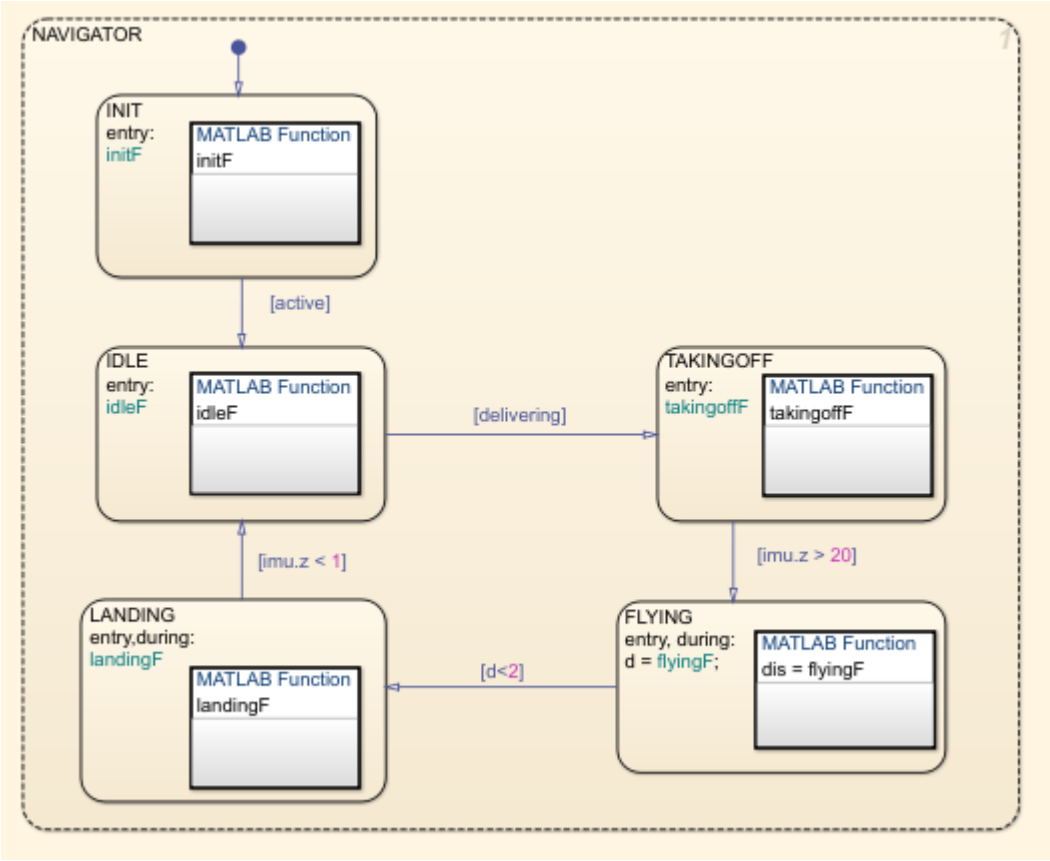

Figura 37. Máquina de estado de UAV

La máquina de estados que controla la navegación de un UAV contiene los estados que tomará en el tiempo dependiendo de una serie de transiciones. Los estados en los que se encontrará un UAV son:

- Init: es el estado inicial y solo se ejecutará la primera vez en cada simulación. A través de este estado se inicializa la estructura de datos local de cada UAV, las posiciones iniciales, el número de UAV involucrados en la simulación y la variable que permite la transición entre estados.

- Idle: representa el estado de un UAV en el suelo, en reposo. En este estado el UAV se encuentra a la espera de una entrega por parte del ATC.
- Taking off: es el estado que indica cuando el UAV está despegando. Se emite a través del puerto '*cmd*' la velocidad de ascenso en el eje '*z*'.
- Flying: es el estado que representa al UAV en vuelo. Principalmente aquí se establece a través del puerto '*cmd*' la velocidad y posición deseadas.
- Landing: es el estado en el cual el UAV se encuentra aterrizando, una vez ha llegado a su destino. Nuevamente se establecen las nuevas velocidades, donde las velocidades en los ejes '*x*' e '*y*' permanecen a cero y es la coordenada '*z*' la que decrece.

Por otro lado, el bloque de comunicación, que es ejecutado en paralelo al bloque de navegación, aunque con menor prioridad, se encarga del tratamiento de los distintos mensajes recibidos.

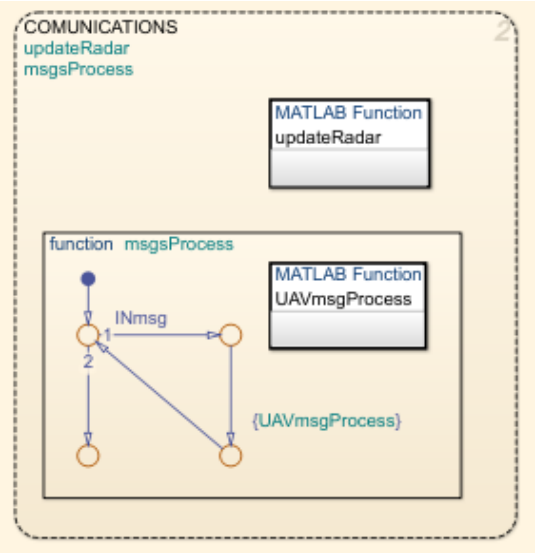

Figura 38. Bloque comunicación de UAV

Gracias a dicho tratamiento es posible la actualización del escenario de simulación, así como la puesta en marcha hacia un nuevo destino por parte del UAV o la detección de una colisión. Tal y como se observa en la Figura 39, cuando se recibe un mensaje en el bloque de comunicación, se comprueba quien es el destinatario y en función de ello se realiza una acción u otra. Si el mensaje recibido es emitido por el ATC, se trata de una nueva entrega que debe realizar un UAV concreto. Si por el contrario el mensaje no proviene del ATC, se comprueba si existe una colisión.

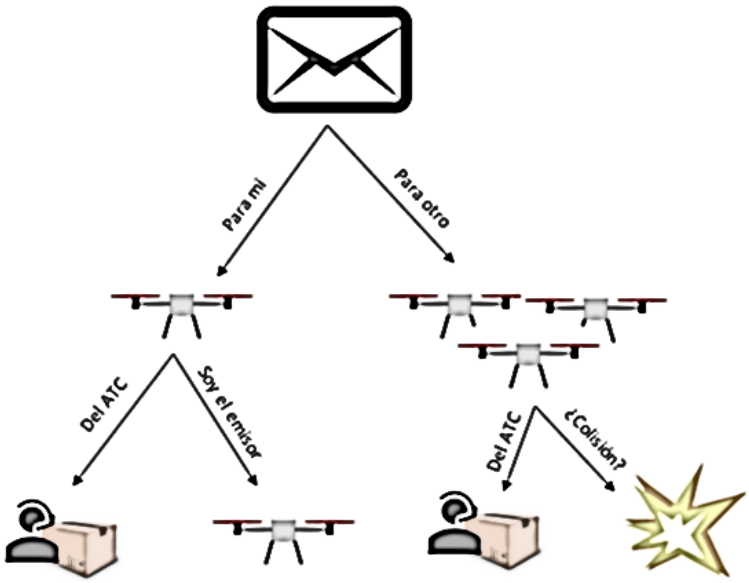

Figura 39. Esquema tratamiento de mensajes en UAV

5.2.3. RADAR

Como puede observarse en la Figura 40, el gráfico 3D que permitirá ver el transcurso de la simulación, es alimentado a través del puerto '*radar*'. A través de este puerto se actualiza la información necesaria, como la posición o el id de cada UAV.

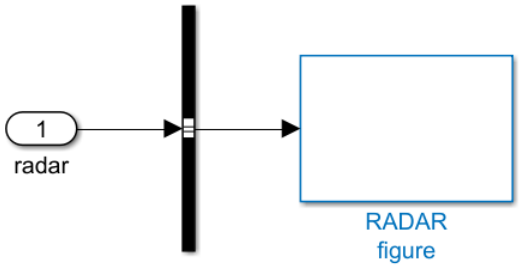

Figura 40. Radar en subsistema UAV

5.3. SUBSISTEMA ATC

Al igual que ocurre en el subsistema UAV, el ATC también recibe y emite mensajes, principalmente para poder enviar órdenes de entrega a cada uno de los UAV. El subsistema ATC se compone de un único bloque de control (ver Figura 41) que se encarga del procesamiento de dichos mensajes. Además inicializa la estructura de datos que maneja el ATC para el control sobre las entregas.

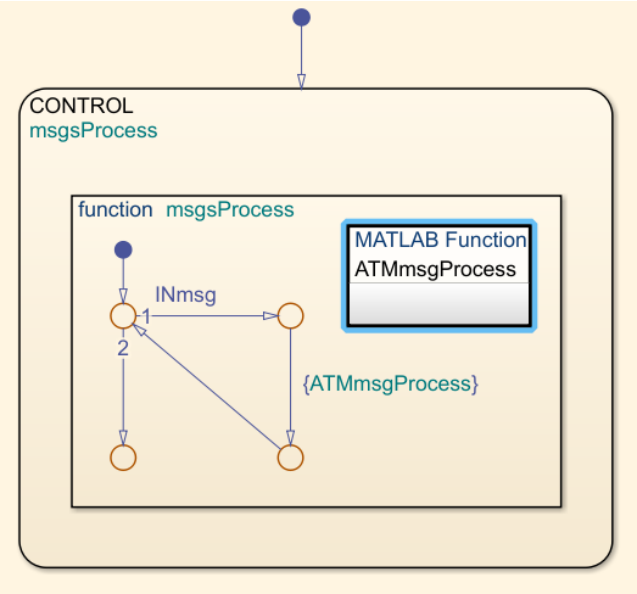

Figura 41. Bloque control de ATC

El procesamiento de los mensajes por parte del ATC puede observarse en la Figura 42. Cuando el ATC recibe un mensaje por parte de un UAV con estado "delivering = false" , este comprueba si existen entregas pendientes y de ser así establece una nueva entrega a dicho UAV que se encuentra disponible para recibir órdenes de más entregas. De no existir entregas, el ATC envía de regreso al UAV a su punto de partida.

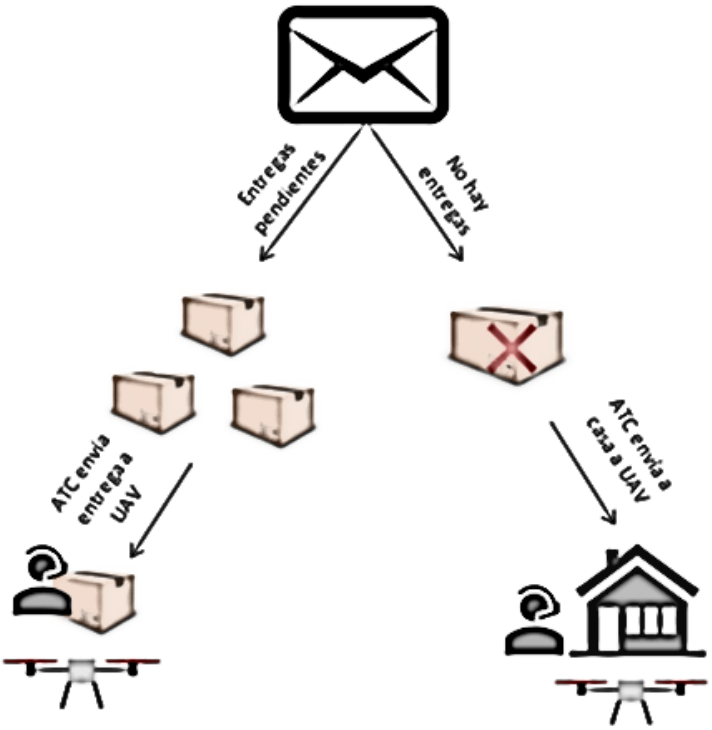

Figura 42. Esquema tratamiento de mensajes ATC

5.4. INTERFAZ DE USUARIO

El simulador tiene una serie de parámetros que deben ser inicializados antes del comienzo de la ejecución. Para el establecimiento de dichos parámetros y así lograr una ejecución personalizada, se ha desarrollado una interfaz de usuario muy intuitiva que permite facilitar dicha tarea (ver Figura 43).

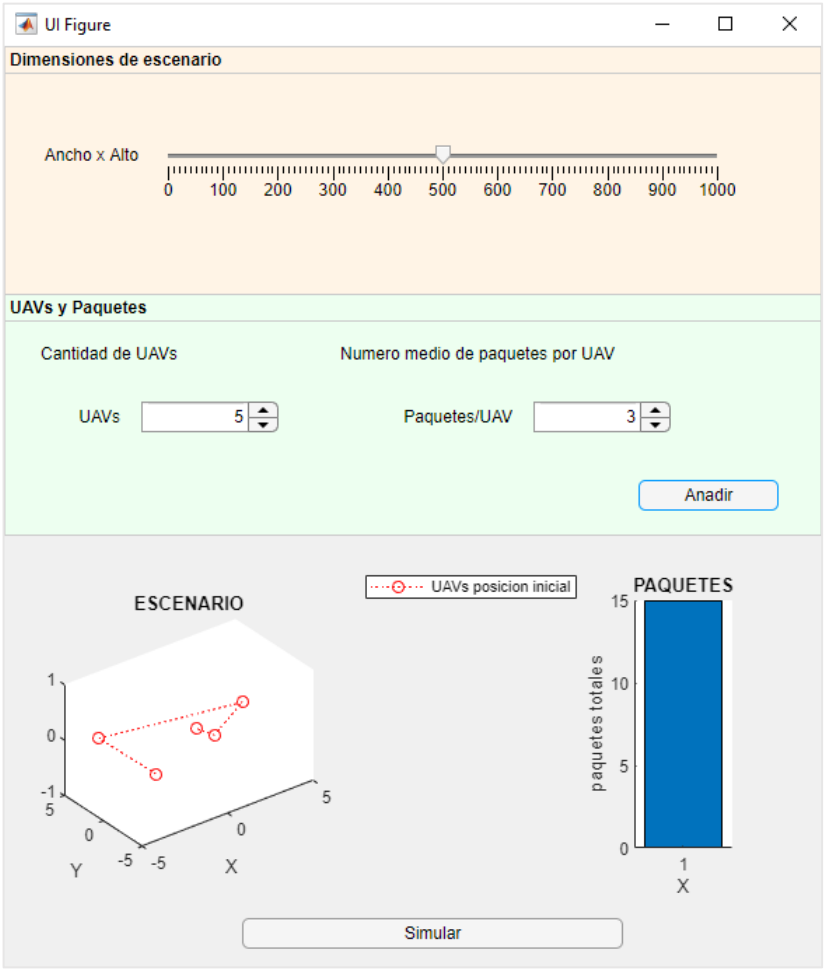

Figura 43. Interfaz de usuario del simulador

Cuando se lanza la simulación de un escenario por medio de la interfaz de usuario proporcionada, se muestra una ventana representando la ejecución de dicha simulación en tiempo real, tal y como se observa en la Figura 44.

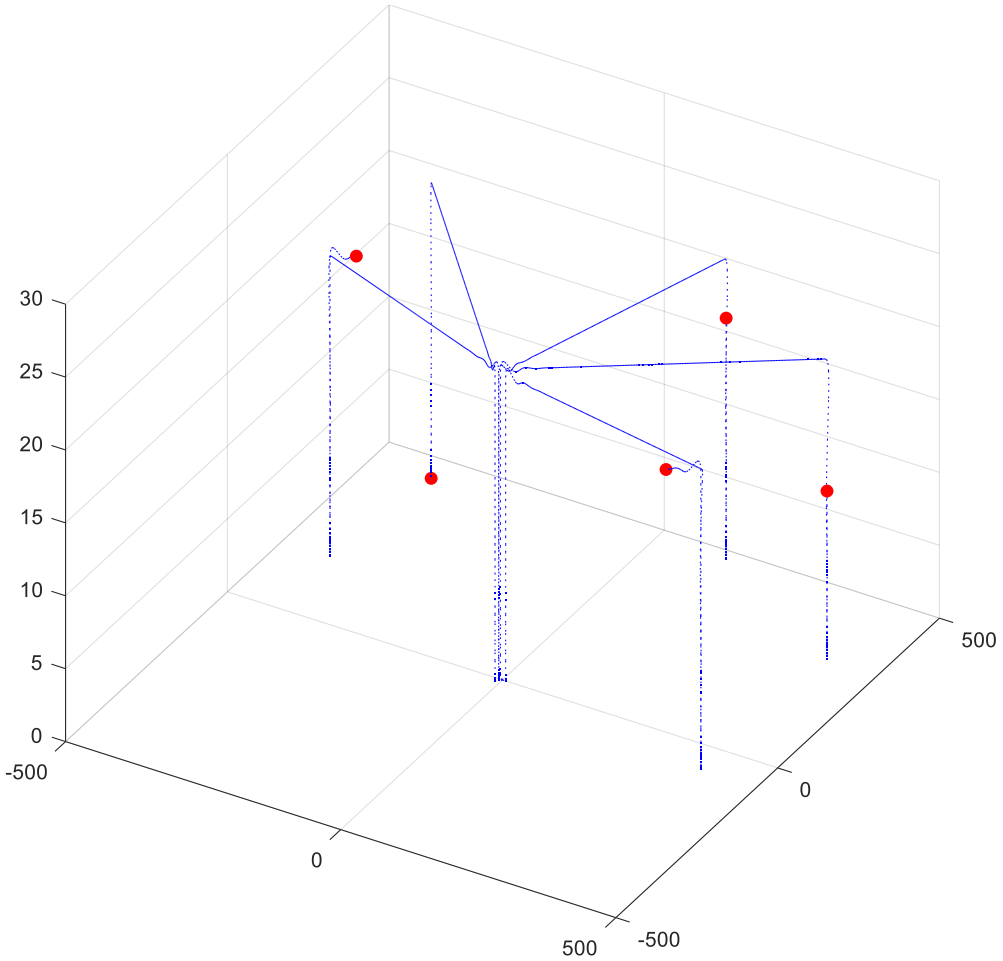

Figura 44. Ejemplo de la ejecución del simulador

CAPÍTULO 6. EXPERIMENTOS Y RESULTADOS

6.1. INTRODUCCIÓN

Uno de los aspectos más importantes de este trabajo consiste en el estudio y análisis de los datos obtenidos sobre el comportamiento del algoritmo BBCA propuesto. Este estudio determinará la viabilidad de la solución, la posible mejora respecto a otros algoritmos, en definitiva, su optimalidad.
Para realizar este estudio se ha elaborado un banco de pruebas por medio de la modificación de aquellas variables que interesan de cara a comparativas o análisis del comportamiento del algoritmo.

Las simulaciones se han llevado a cabo en el analizador de algoritmos que fue abordado en el CAPÍTULO 4 de esta memoria. La metodología empleada para las simulaciones realizadas conlleva cuatro pasos:

1. Elaboración de escenarios almacenados en archivos .m (de MATLAB), etiquetados con los parámetros que caracterizan al escenario (por ejemplo, "*eval8_2A_50V_5R*") o por el contrario generación de escenario de forma aleatorizada.
2. Elaboración de un *script* para generación de gráficas estadísticas ("Analitics.m").
3. Ejecución de la simulación.
4. Almacenamiento de los datos obtenidos en la simulación en archivos .mat, .csv, .fig o .pdf para cada estudio.

De esta manera se genera un banco de pruebas de forma ordenada, permitiendo su posterior análisis y extracción de datos de forma eficiente. El estudio se divide en dos partes:

- Escenarios con dos UAV.
- Escenarios con múltiples UAV.

Se analiza el comportamiento de forma precisa para dos agentes virtuales, siendo extendido más tarde a escenarios con múltiples agentes.

6.2. ESTUDIO CON DOS UAV

La idea principal es realizar un estudio en profundidad del comportamiento del algoritmo BBCA frente al algoritmo *direct*. El algoritmo *direct* dirige a cada UAV en línea recta hacia su objetivo. En este apartado se analizan en detalle los escenarios con dos UAV involucrados.

Uno de los detalles principales que hacen que los resultados obtenidos sean válidos para el estudio del comportamiento del algoritmo propuesto se basa en la totalidad del impacto entre los UAV. Como ya se ha indicado, los UAV disponen de un radio de seguridad que permite modelar una circunferencia como zona de prohibición para el resto de UAV. Si la colisión producida por dos UAV implica el solapamiento de sus circunferencias, se habla de un impacto total.

Para que el impacto de todos los escenarios simulados sea total, se ha situado al primer UAV con una posición inicial y final estática. Es el segundo UAV el que irá modificando sus posiciones. Las trayectorias relativas de este segundo UAV son elegidas en base a las combinaciones posibles de una circunferencia, tal y como se muestra en la Figura 45. Por lo tanto, el segundo UAV tomará las posiciones situadas desde el grado cero hasta el 170, reservando la posición relativa al grado 180 para el primer UAV (en total se simulan 18 escenarios). La posición de destino para cada UAV se corresponde con su posición opuesta respecto a la circunferencia mostrada en la Figura 45. Por lo tanto, el primer UAV se desplazará desde el oeste hasta el este, que se corresponde con la posición relativa a la circunferencia desde el grado 180 al cero. El centro de dicha circunferencia se corresponde con el punto de colisión en todos los casos.

Para la obtención de los datos se ha generado un *script*, de la misma forma en la que se venía haciendo, que automatice todo el proceso de forma sistemática, almacenando los datos de la simulación y las figuras (recorrido de UAV y estadísticas) de cada simulación.

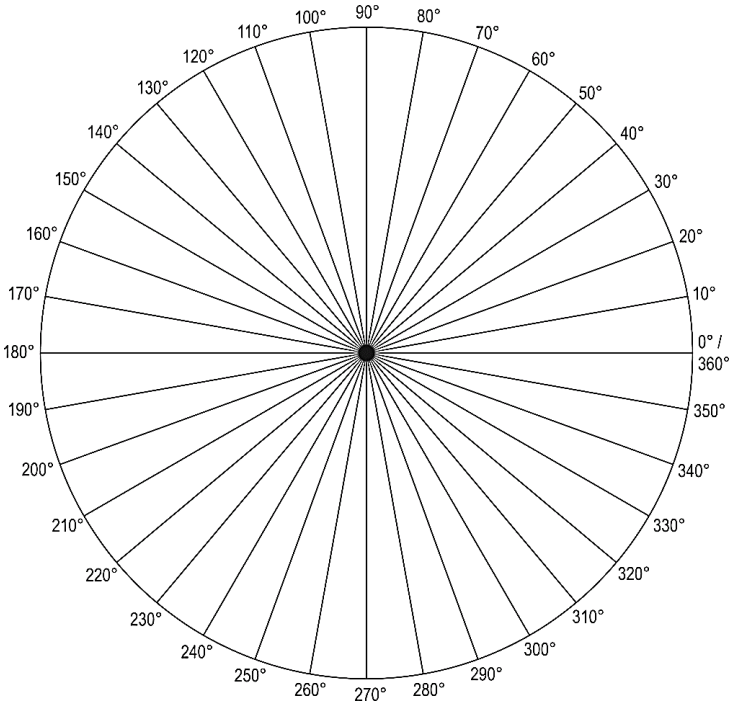

Figura 45. Grados en una circunferencia

Los escenarios se configuran con las siguientes características:

- Velocidad máxima de 13.89 m/s (50 km/h).
- Radio de seguridad de los UAV de 50 m.
- Cada UAV realiza un recorrido de 2000 m desde su posición inicial hasta el punto de colisión (centro de la circunferencia) y otras 2000 m desde el punto de colisión hasta su destino.

En la Figura 46 se representa en el eje de abscisas el número de escenario correspondiente a la simulación de los dos UAV con las rutas relativas descritas anteriormente. El eje de ordenadas representa la media de colisiones producidas. Por lo tanto, los resultados representan el número medio de colisiones producidas en cada escenario, empleando el algoritmo *direct* y el algoritmo BBCA. Como se puede observar, el número de colisiones empleando *direct* es del 100%, es decir, en todos los escenarios los dos UAV colisionan. Por el contrario, al emplear el algoritmo BBCA, las colisiones medias producidas por escenario son de un 0%.

Por lo tanto, el algoritmo BBCA logra evitar el 100% de las colisiones para los 18 escenarios y esto se debe al éxito de este.

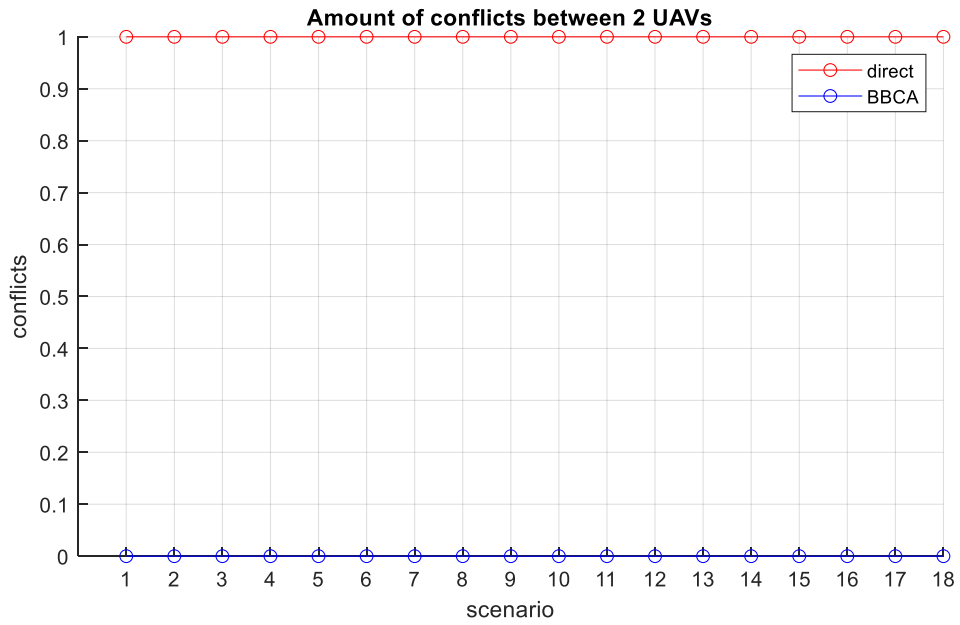

Figura 46. Media de colisiones empleando los algoritmos direct y BBCA

Pero para que el algoritmo BBCA funcione correctamente es necesario pagar el coste del incremento en la distancia recorrida y en el tiempo empleado por los UAV para llegar a su destino. La Figura 47 representa en el eje de abscisas el número de escenario y en el eje de ordenadas la distancia media recorrida por los UAV, expresada en metros. El uso del algoritmo BBCA supone un incremento respecto a emplear el algoritmo *direct*. Esto se debe a que los UAV modifican sus rutas originales para evitar la colisión, lo que se traduce en un aumento de la distancia recorrida.

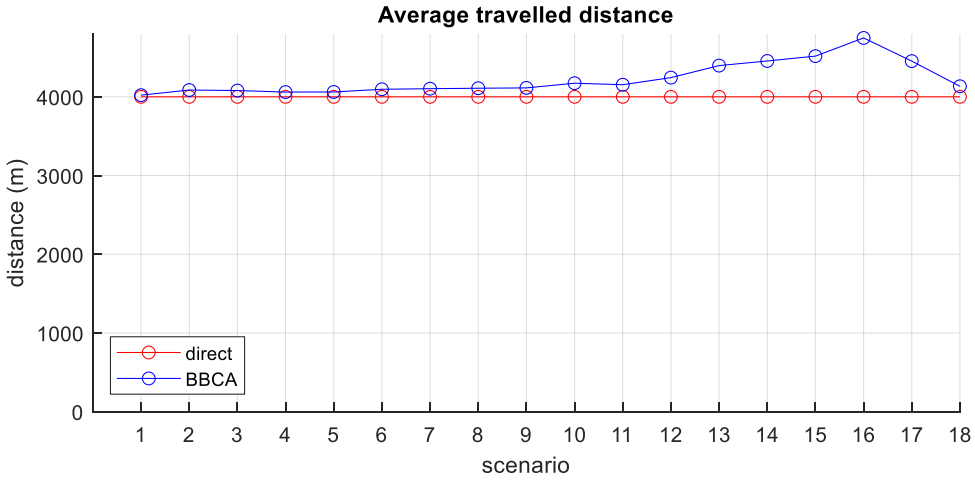

Figura 47. Media de la distancia recorrida para 2 UAV

En la Figura 48 se observa de forma más clara este incremento en la distancia recorrida por los UAV. La figura representa en el eje de abscisas el número de escenario y en el eje de ordenadas el incremento relativo de la distancia recorrida respecto al algoritmo *direct*. Como se puede observar, en el peor de los casos este incremento no supera el 20%.

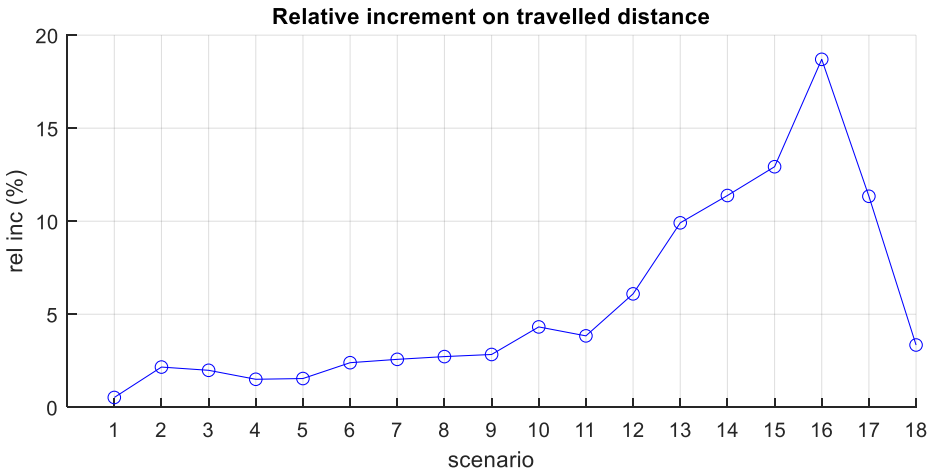

Figura 48. Incremento en la distancia recorrida para 2 UAV

Como es obvio, el tiempo medio consumido por los UAV cuando se utiliza el algoritmo BBCA frente a no utilizarlo (algoritmo *direct*) es proporcional a la distancia media recorrida por estos. La Figura 49 representa en el eje de abscisas el número de escenario y en el eje de ordenadas el tiempo medio consumido por los UAV, expresado en segundos. Como se puede observar, el uso de BBCA supone un incremento en el tiempo que emplean los UAV para llegar a su destino. Esto se debe a que los UAV se desvían de su ruta inicial para evitar la colisión, produciendo un incremento en la distancia recorrida y por consiguiente un aumento en el tiempo empleado para llegar a destino.

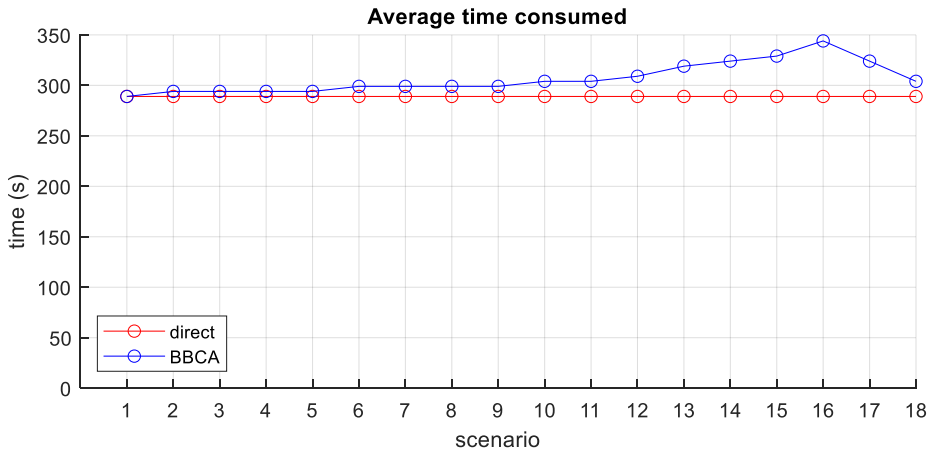

Figura 49. Media de tiempo consumido para 2 UAV

Al igual que se ha observado un incremento superior en algunos escenarios con respecto al resto en el caso de la distancia media recorrida por los UAV, se observa en la Figura 50 el mismo incremento, pero esta vez con respecto al tiempo consumido por los UAV. Este incremento se debe a esa desviación de la ruta inicial de la que se ha hablado anteriormente, para conseguir evitar la colisión. Como se puede observar en la figura, el incremento en el tiempo consumido por los UAV al emplear el algoritmo BBCA con respecto al uso del algoritmo *direct* no llega a un 20%, en el peor de los casos.

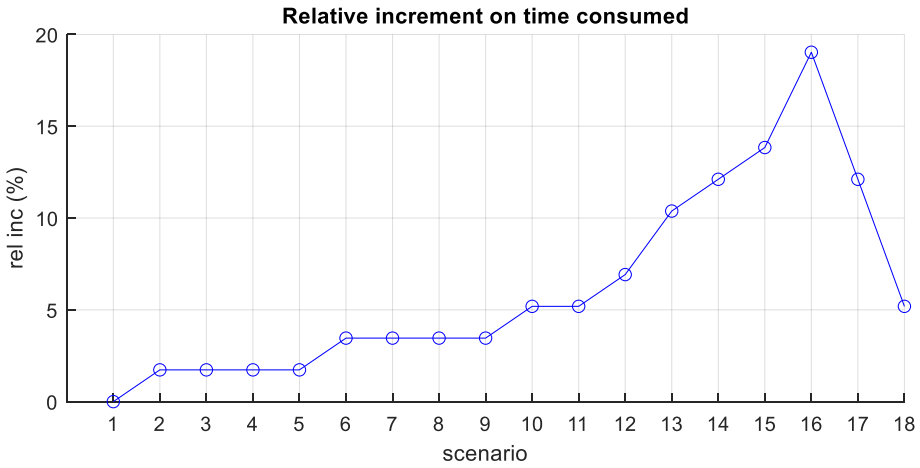

Figura 50. Incremento en el tiempo consumido para 2 UAV

En resumen, el estudio realizado para escenarios compuestos por dos UAV supone un éxito en el comportamiento del algoritmo BBCA. Esto se debe a que evitan el 100% de las colisiones, para todos los escenarios, suponiendo, no obstante, un incremento de hasta el 20% en la distancia media recorrida y en el tiempo medio empleado por los UAV para llegar a su destino.

6.2.1. ANÁLISIS DE UN ESCENARIO REALISTA

Una de las características más deseables de una simulación es que sea lo más próxima posible a la realidad. A continuación, se realiza la simulación sobre un escenario en el que los agentes virtuales (UAV) deben ofrecer un servicio en un municipio/ciudad, desde un punto inicial hasta su objetivo/destino.

Las características empeladas en el escenario que se analiza en este apartado se corresponden con las siguientes:

- Área de 10 Km x 10 Km.
- 2 UAV (numUAV).
- UAV con radio de seguridad 50 m (UAVrad).
- Velocidad máxima 50 km/h, equivalente a 13.89 m/s (vel_max).
- Las rutas suponen el cruce (colisión) de los UAV.

Sin el empleo del algoritmo BBCA, el detector de colisiones del analizador detecta dicha situación.

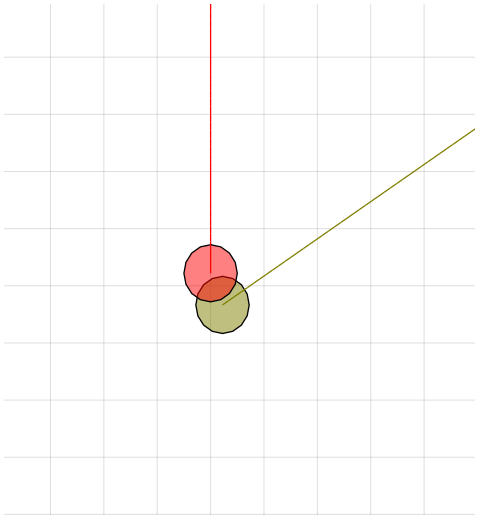

Figura 51. Colisión entre 2 UAV sin empleo de BBCA

El objetivo principal del algoritmo es evitar que se produzca la colisión entre los dos UAV (ver Figura 51). Para este escenario, dicha colisión ha sido resuelta tal y como se observa en la Figura 52, obteniendo unos resultados que se muestran a continuación en forma de gráficas.

Figura 52. Simulación de escenario con 2 UAV empleando BBCA (izquierda) y la desviación realizada en la simulación (derecha)

La Figura 52 (izquierda) se corresponde con el resultado de la simulación de este escenario y la Figura 52 (derecha) con la maniobra (desviación) realizada por los UAV para evitar su colisión.

Es necesario matizar que en la figura se muestra el mapa de Albacete, como el municipio sobre el que vuelan los dos UAV. Sin embargo, como se comenta en el CAPÍTULO 2, según el Real Decreto 1036/2017, de 15 de diciembre, los UAV deben encontrarse a una distancia mínima de 8 km respeto del punto de referencia de cualquier aeropuerto y tendrán prohibido el vuelo en zonas controladas. Por lo tanto, esta imagen es meramente ilustrativa y se ha seleccionado teniendo en cuenta que no podría realizarse según la normativa vigente.

A continuación, se muestran los resultados obtenidos de esta simulación.

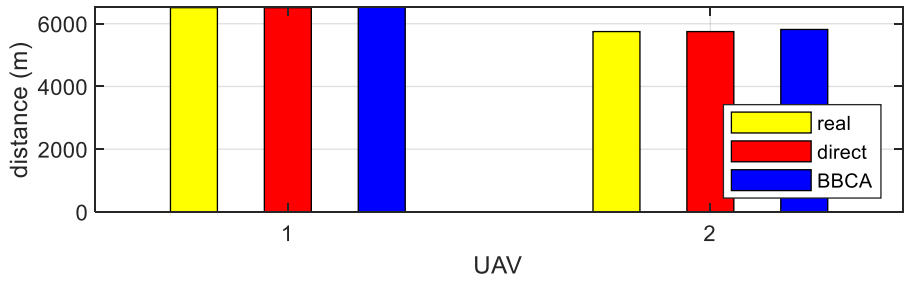

Figura 53. Distancia recorrida por los UAV utilizando BBCA, direct y distancia real

Como se puede observar en la Figura 53, la distancia recorrida al emplear el algoritmo BBCA es similar a la distancia recorrida al utilizar el algoritmo *direct* o a la distancia real (sin empleo de algoritmos). Por lo tanto, la desviación del objetivo es mínima, tal y como se observa en Figura 54.

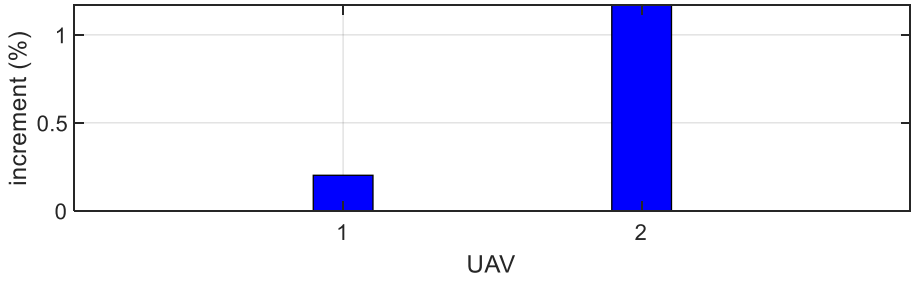

Figura 54. Porcentaje de desviación de la ruta inicial por los UAV

Por otro lado, el tiempo empleado para realizar el recorrido al utilizar el algoritmo BBCA no se ve incrementado, tal y como se muestra en Figura 55.

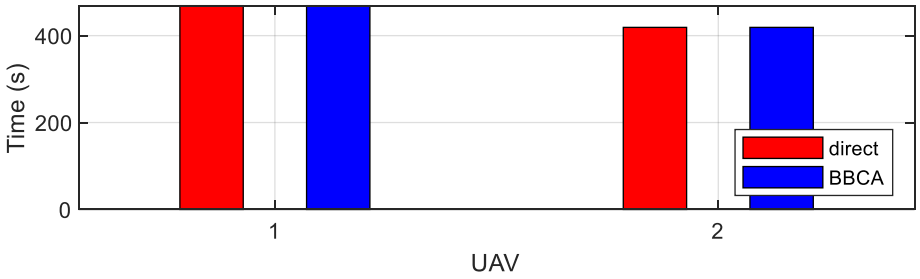

Figura 55. Tiempo total empelado con BBCA vs direct

En resumen, los UAV logran llegar a su destino evitando la colisión, y todo ello sin incrementar el tiempo de vuelo. Solo se observa un ligero incremento en la distancia recorrida por los UAV para llegar a su destino, y este incremento no llega al 2% con respecto a su ruta inicial.

6.3. ESTUDIO CON MÚLTIPLES UAV

En este apartado se analiza el comportamiento del algoritmo BBCA en escenarios con varios UAV. En concreto, se considerarán entre 10 y 100 UAV.

Para poder obtener el conjunto de resultados se han automatizado las simulaciones, generando un *script* que lance todo el proceso. El proceso consiste en la ejecución de 24 escenarios aleatorios diferentes para 10, 20, 30, 40, 50, 60, 70, 80, 90 y 100 UAV. Estas 24 simulaciones se realizan aplicando el algoritmo BBCA y repitiendo el proceso sin aplicarlo, es decir, en línea recta (algoritmo *direct*). En total, se consideran 240 escenarios distintos para el análisis. La velocidad máxima de cada UAV es de 50 km/h, el radio de seguridad de cada UAV es de 50 m y el área del escenario es de 5 km x 5 km, para todas las simulaciones.

Tabla 9. Media de colisiones con múltiples UAV

Number of UAV	*Collisions expected*	*Collisions using BBCA*
10	1.75	0.08
20	7.00	0.21
30	13.42	0.42
40	27.87	1.37
50	42.87	2.75
60	57.54	3.87
70	83.83	8.21
80	105.21	9.58
90	133.87	16.29
100	175.67	21.17

La Tabla 9 representa la media de colisiones producidas en las simulaciones. La Figura 56 representa en el eje de abscisas la cantidad de UAV que conforman los escenarios y en el eje de ordenadas el número medio de colisiones producidas. Se observa como su comportamiento es creciente, es decir, a medida que el espacio aéreo dispone de un mayor número de UAV, el número de conflictos empleando el algoritmo *direct* aumenta. Por otro lado, se observa como el empleo del algoritmo BBCA supone una reducción considerable de conflictos frente a no utilizarlo. El número medio de conflictos utilizando BBCA no supera el 10%, es decir, se consigue una reducción de conflictos en torno a un 90% en el peor de los casos.

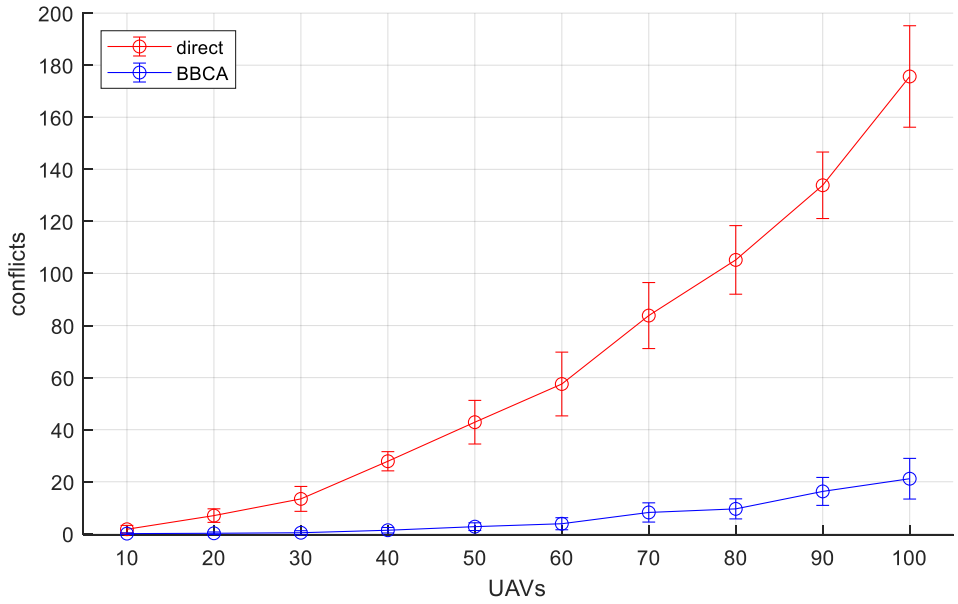

Figura 56. Media y desviación típica de conflictos para 240 escenarios con diferente cantidad de UAV (área de 5 km x 5 km)

Los siguientes resultados muestran la penalización al utilizar el algoritmo BBCA en cuanto a distancia recorrida y a tiempo consumido por los UAV, de la misma forma en la que se ha estudiado anteriormente para dos UAV.

En la Figura 57 se representa en el eje de abscisas la cantidad de UAV y en el eje de ordenadas la distancia recorrida, expresada en metros. Se aprecia que la distancia media recorrida por los UAV es creciente cuando se utiliza BBCA, cosa que no sucede en el caso de *direct*. Por lo tanto, para poder evitar casi el 90% de las colisiones en el peor de los casos, los UAV emplean rutas de mayor longitud para llegar a su destino.

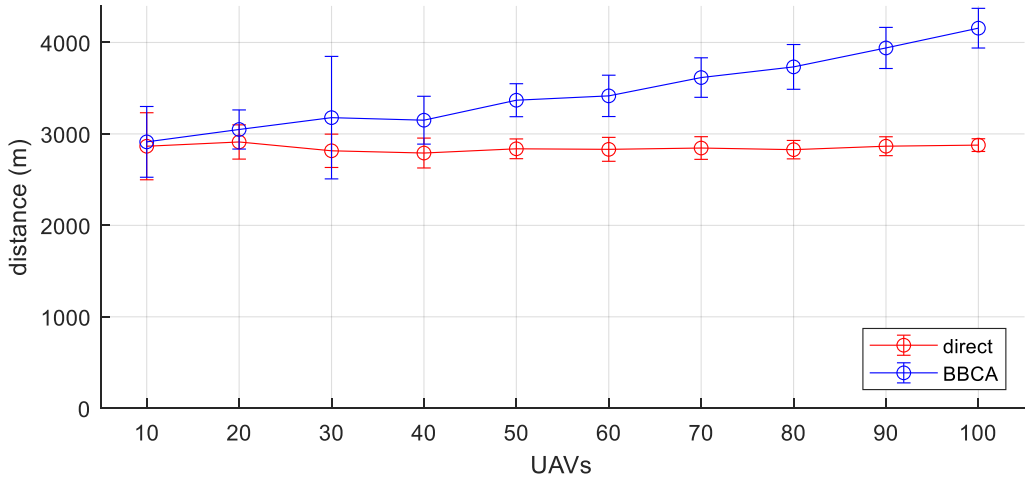

Figura 57. Distancia media recorrida por los UAV con los algoritmos BBCA y direct

En la Figura 58 se representa en el eje de abscisas la cantidad de UAV y en el eje de ordenadas el incremento relativo en la distancia recorrida por los UAV, expresado en porcentaje. Se observa el incremento en la distancia recorrida por los UAV al emplear el algoritmo BBCA. En la mayoría de los casos este incremento no supera el 30%, siendo inferior al 50% en el peor de ellos. Por lo tanto, si consideramos que el algoritmo evita casi el 90% de las colisiones en el peor de los casos, se puede afirmar que la penalización en distancia recorrida por los UAV no es excesiva.

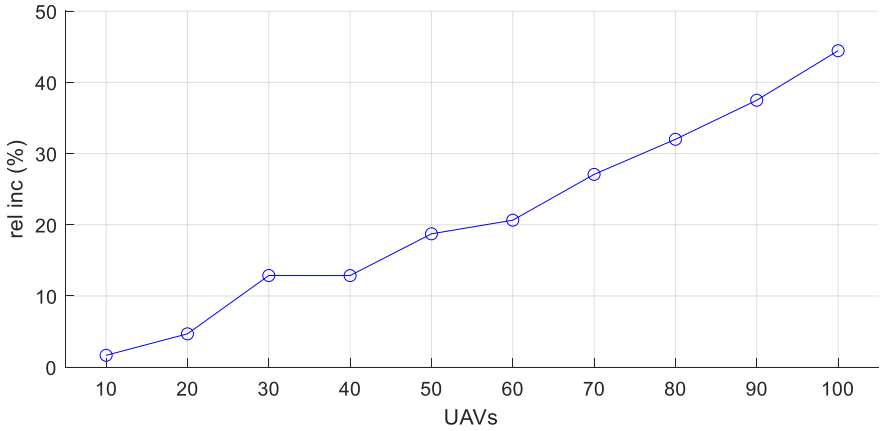

Figura 58. Incremento en la distancia recorrida por los UAV al utilizar BBCA y direct

Las siguientes figuras muestran el tiempo medio empleado por los UAV para llegar a su destino, al usar los algoritmos BBCA y *direct* (en línea recta). Se observa un comportamiento similar a lo visto anteriormente en el caso de la distancia recorrida. En la Figura 59 se representa en el eje de abscisas la cantidad de UAVs y en el eje de ordenadas el tiempo medio consumido por los UAV, expresado en segundos. Se aprecia claramente un comportamiento creciente al emplear BBCA, frente a lo que sucede en el caso del algoritmo *direct*. Este aumento de tiempo se corresponde con la desviación de su ruta inicial que deben realizar los UAV para evitar la colisión. Esta desviación supone un aumento de la distancia recorrida y por lo tanto del tiempo consumido por cada UAV.

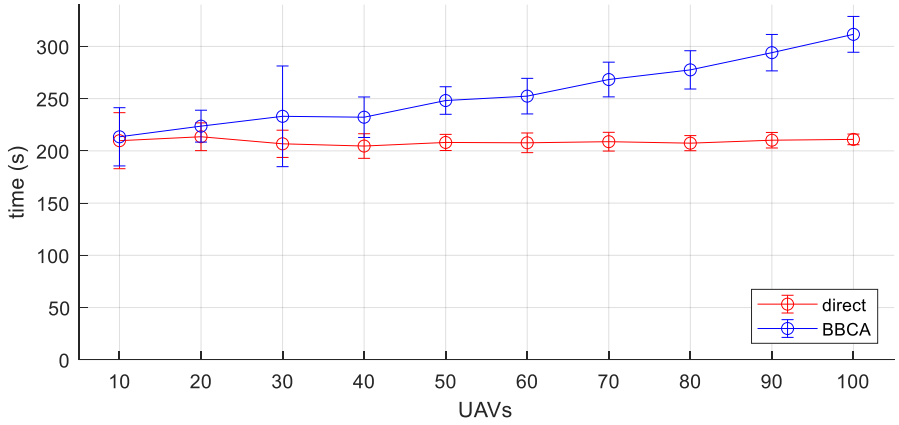

Figura 59. Tiempo medio consumido por los UAV con los algoritmos BBCA y direct

En la Figura 60 se representa en el eje de abscisas la cantidad de UAV y en el eje de ordenadas el incremento relativo de tiempo consumido por los UAV, expresado en %. El incremento del tiempo consumido por los UAV no supera el 30% en la mayoría de los casos, y en el peor de ellos no supera al 50%. Este incremento es equivalente al de la distancia recorrida por los UAV, como era de esperar.

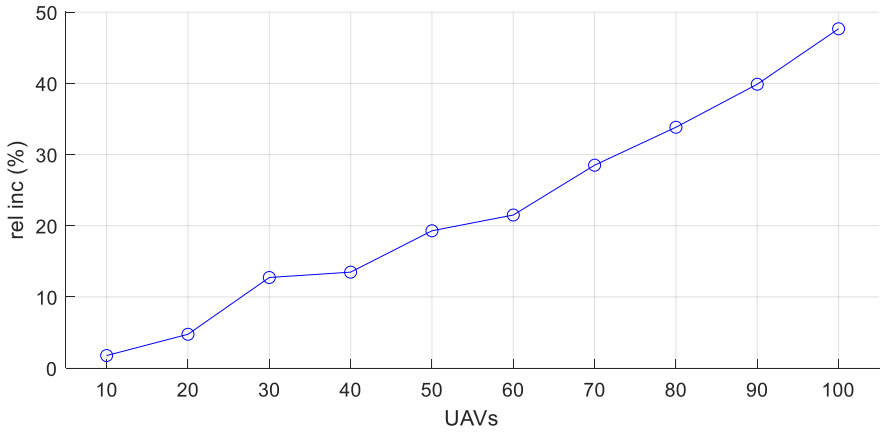

Figura 60. Incremento en el tiempo consumido por los UAV al utilizar BBCA y direct

Utilizar el algoritmo BBCA en escenarios con múltiples UAV en el espacio aéreo supone una reducción del 90% de las colisiones, con un incremento medio del 20% en la distancia recorrida y en el tiempo consumido, para todos los escenarios. Por lo tanto, el algoritmo BBCA propuesto en este trabajo se comporta correctamente, aunque lo ideal sería observar una reducción del 100% de las colisiones a un coste de tiempo y distancia recorrida mínimo.

CAPÍTULO 7. CONCLUSIONES Y PROPUESTAS

En este capítulo se detallan las principales conclusiones de nuestro trabajo y se sugieren algunas líneas de trabajo directamente relacionados con la tarea aquí iniciada.

7.1. CONCLUSIONES

Como se indica en el CAPÍTULO 1 de esta memoria, el objetivo principal de este trabajo era **"realizar un primer acercamiento a los mecanismos dinámicos de detección y resolución de colisiones entre drones"**. Consideramos que dicho objetivo ha sido sobradamente cumplido.

A continuación, recuperamos los subobjetivos planteados inicialmente y detallamos la manera en la que se ha satisfecho cada uno de ellos:

1. **Estudiar los mecanismos existentes en la literatura para la detección y resolución de conflictos en entornos con múltiples UAV**. Para cumplir este primer subobjetivo, hemos tenido que realizar una revisión de las publicaciones del estado del arte. Las más relevantes para nuestro trabajo han sido resumidas en el CAPÍTULO 2 de esta memoria.
2. **Desarrollar una herramienta de análisis que permita testear el mecanismo de evitación de colisiones propuesto en este trabajo**. Esta herramienta ha sido abordada en el CAPÍTULO 4 de esta memoria. Posibilita la simulación de escenarios con múltiples UAV, reproduciendo sus trayectorias y mostrando el comportamiento de cada fase del algoritmo. Además, permite obtener de forma rápida, sencilla y ordenada un conjunto de resultados útiles para el análisis.
3. **Desarrollar una herramienta de simulación que permita probar nuevos mecanismos de evitación de colisiones en escenarios con tres dimensiones**. Esta herramienta reproduce de una manera realista el espacio aéreo, en el que un conjunto de UAV se desplazan hacia una serie de destinos que son proporcionados por una

entidad específica (el ATC). El modelado de las entidades involucradas en la simulación ha sido detallado en el CAPÍTULO 5 de esta memoria. La herramienta permite la inclusión de mecanismos de evitación de colisiones de una forma sencilla y almacena los datos obtenidos en la simulación para un posterior análisis.

4. **Diseñar un algoritmo sencillo de evitación de colisiones**. El algoritmo BBCA propuesto en el CAPÍTULO 4 de esta memoria es un mecanismo sencillo, en lo que a implementación y coste computacional se refiere, con respecto a otras propuestas presentes en la literatura. Esto se debe principalmente a la complejidad de cálculo asociada a los mecanismos tradicionales de evitación de colisiones, suponiendo en la mayoría de los casos un problema de optimización. Nuestra propuesta se enmarca en los métodos geométricos o de velocidad, debido a que se apoya principalmente en la actualización de la velocidad de los UAV con el apoyo de configuraciones geométricas, garantizando esta simplicidad.
5. **Llevar a cabo una evaluación comparativa de las prestaciones del nuevo algoritmo. En su caso, proponer mejoras al mismo**. La evaluación realizada se presenta en el CAPÍTULO 6 de esta memoria, donde se presentan y analizan los resultados obtenidos considerando escenarios con dos UAV y con múltiples UAV. Este análisis muestra un buen comportamiento general del algoritmo BBCA, ya que logra evitar el 100% de las colisiones en escenarios con dos UAV y el 90% en escenarios con múltiples UAV, con un incremento en la distancia recorrida y en el tiempo consumido por los UAV inferior al 30%.

7.2. TRABAJO FUTURO

Para finalizar, se proponen una serie de temas de trabajo que podrían dar continuidad o complementar el presente TFG:

1. Mejora del mecanismo BBCA propuesto para la detección y evitación de colisiones, de forma que el número de colisiones en cualquier escenario sea nulo, sin llegar a pagar un alto coste en cuanto a incrementos en la longitud de la ruta o en el tiempo necesario para recorrerla.
2. Llevar a cabo una nueva evaluación comparativa, pero en este caso comparando las prestaciones del mecanismo BBCA con los mecanismos de evitación para UAV más

populares (basados en velocidad o no) del estado del arte (ORCA, APF, etc.).

3. Adaptar nuestra propuesta a colisiones en 3D, para lograr ofrecer una solución mucho más realista y flexible, en el sentido de que se permita el vuelo de los UAV a distintas alturas.
4. En el proceso de definición del algoritmo BBCA propuesto se establecieron las bases para la elaboración de algoritmos para la gestión de conflictos alternativos. En un futuro se pretende retomar la exploración de dichas propuestas, que fue aplazada para centrarnos en el algoritmo BBCA finalmente desarrollado.
5. Mejora o ampliación de la herramienta de simulación de espacio aéreo desarrollada, de forma que sirva como medio de control de los UAV que se encuentren en vuelo en una región determinada. Consideramos que esto es muy interesante, ya que se prevé que la normativa que regula estos vehículos exija como requisito en un futuro la inclusión de los UAV autónomos en una herramienta que permita su control.
6. Adaptación o particularización de la herramienta de simulación de espacio aéreo desarrollada para aplicaciones civiles concretas.

BIBLIOGRAFÍA

[1] "BOE.es - Documento BOE-A-2017-15721." [Online]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2017-15721.

[2] "'Sense and Avoid', seguridad en el vuelo de los UAV | Embention." [Online]. Available: https://www.embention.com/es/news/sense-and-avoid-seguridad-vuelo-uav/.

[3] J. K. Kuchar and L. C. Yang, "A Review of Conflict Detection and Resolution Modeling Methods," 2000.

[4] J. Tang, "Conflict Detection and Resolution for Civil Aviation: A literature survey," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 34, no. 10, pp. 20–35, 2019, doi: 10.1109/MAES.2019.2914986.

[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, no. 4. pp. 179–189, Dec-2000, doi: 10.1109/6979.898217.

[6] International Civil Aviation Organization (ICAO), "Doc 4444. PANS-ATM, Procedures for Air Navigation Services. Air Traffic Management," 2016.

[7] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson, "Unmanned aircraft system traffic management (UTM) concept of operations," in *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016.

[8] R. Shakeri *et al.*, "Design Challenges of Multi-UAV Systems in Cyber-Physical Applications: A Comprehensive Survey and Future Directions," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 4, pp. 3340–3385, 2019, doi: 10.1109/COMST.2019.2924143.

[9] S. Huang, R. S. H. Teo, and K. K. Tan, "Collision avoidance of multi unmanned aerial vehicles: A review," *Annu. Rev. Control*, vol. 48, pp. 147–164, 2019, doi: 10.1016/j.arcontrol.2019.10.001.

[10] F. Ho, R. Geraldes, A. Goncalves, M. Cavazza, and H. Prendinger, "Improved Conflict Detection and Resolution for Service UAVs in Shared Airspace," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1231–1242, 2019, doi: 10.1109/TVT.2018.2889459.

[11] P. Fiorini and Z. Shillert, "Motion Planning in Dynamic Environments using Velocity

Obstacles," *Int. J. Rob. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[12] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans.*, vol. 28, no. 5, pp. 562–574, 1998, doi: 10.1109/3468.709600.

[13] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "Velocity Obstacle Approaches for Multi-Agent Collision Avoidance," *Unmanned Syst.*, vol. 7, no. 1, pp. 55–64, 2019, doi: 10.1142/S2301385019400065.

[14] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "A Comparative Study of Velocity Obstacle Approaches for Multi-Agent Systems," in *2018 UKACC 12th International Conference on Control, CONTROL 2018*, 2018, pp. 289–294, doi: 10.1109/CONTROL.2018.8516848.

[15] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-Body Collision Avoidance," in *Springer Tracts in Advanced Robotics*, vol. 70, no. STAR, 2011, pp. 3–19.

[16] H. Niu, C. Ma, P. Han, and J. Lv, "An airborne approach for conflict detection and resolution applied to civil aviation aircraft based on ORCA," in *Proceedings of 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference, ITAIC 2019*, 2019, pp. 686–690, doi: 10.1109/ITAIC.2019.8785582.

[17] J. Sun, J. Tang, and S. Lao, "Collision Avoidance for Cooperative UAVs with Optimized Artificial Potential Field Algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, Aug. 2017, doi: 10.1109/ACCESS.2017.2746752.

[18] Y. Li *et al.*, "A satisficing conflict resolution approach for multiple UAVs," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1866–1878, 2019, doi: 10.1109/JIOT.2018.2885147.

[19] D. Alejo, J. A. Cobano, G. Heredia, and A. Ollero, "Particle Swarm Optimization for collision-free 4D trajectory planning in Unmanned Aerial Vehicles," in *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, 2013, pp. 298–307, doi: 10.1109/ICUAS.2013.6564702.

[20] B. Li, X. Qi, B. Yu, and L. Liu, "Trajectory Planning for UAV Based on Improved ACO Algorithm," *IEEE Access*, vol. 8, pp. 2995–3006, 2020, doi: 10.1109/ACCESS.2019.2962340.

[21] O. Khatib, "The Potential Field Approach And Operational Space Formulation In

Robot Control," in *Adaptive and Learning Systems*, Boston, MA: Springer US, 1986, pp. 367–377.

[22] R. Poli, "Analysis of the Publications on the Applications of Particle Swarm Optimisation," *J. Artif. Evol. Appl.*, vol. 685175, 2008, doi: 10.1155/2008/685175.

[23] "Wireless Technology: A PhD Engineer's Random Thoughts: An Introduction to Particle Swarm Optimization (PSO) with Applications to Antenna Optimization Problems." [Online]. Available: http://wirelesstechthoughts.blogspot.com/2013/06/an-introduction-to-particle-swarm.html.

[24] Z. A. Daniels, L. A. Wright, J. M. Holt, and S. Biaz, "Collision Avoidance of Multiple UAS Using a Collision Cone-Based Cost Function."

[25] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *Am. J. Math.*, 1957, doi: 10.2307/2372560.

[26] M. Shanmugavel, A. Tsourdos, B. White, and R. Zbikowski, "Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs," *Control Eng. Pract.*, vol. 18, no. 9, pp. 1084–1092, Sep. 2010, doi: 10.1016/j.conengprac.2009.02.010.

[27] J. Snape, S. J. Guy, M. C. Lin, D. Manocha, and J. Van Den Berg, "Reciprocal collision avoidance and multi-agent navigation for video games," *AAAI Workshop - Technical Report*, 2012. .

[28] "MATLAB - El lenguaje del cálculo técnico - MATLAB & Simulink." [Online]. Available: https://es.mathworks.com/products/matlab.html.

[29] "Simulink - Simulación y diseño basado en modelos - MATLAB & Simulink." [Online]. Available: https://es.mathworks.com/products/simulink.html.

[30] "Crear un repositorio Git en Github y subir el código." [Online]. Available: https://desarrolloweb.com/articulos/crear-repositorio-git-codigo.html.

[31] "Chat, reuniones, llamadas y colaboración | Microsoft Teams." [Online]. Available: https://www.microsoft.com.

[32] "Trello." [Online]. Available: https://trello.com/es.

[33] "What is Scrum?" [Online]. Available: https://www.scrum.org/resources/what-is-scrum.

ANEXOS

A.1. USO DE LA HERRAMIENTA "ANALIZADOR"

A continuación, se describe paso a paso el uso de la herramienta con un ejemplo simple. Se muestra en la Figura 61 cada uno de estos pasos.

- Paso 1: debemos disponer de la herramienta MATLAB, en su versión R2019-b (o posterior).
- Paso 2: ejecutar el *script* "analiz.m" ubicado en la carpeta "analiz_boundingBox".
- Paso 3: aparecerá un menú con escenarios precargados. Debemos seleccionar aquel que deseemos o ejecutar un escenario de forma aleatoria ingresando los parámetros que nos solicite la interfaz.
- Paso 4: la simulación se encuentra en marcha.
- Paso 5: al finalizar la ejecución se obtienen los resultados. Estos son almacenados en dos carpetas (datos y figuras).

Además, es posible ejecutar de forma automática múltiples escenarios, para así obtener un banco de pruebas. Para ello debemos sustituir el "Paso 2", y ejecutar en su caso el *script* "simulator_automatic_2UAVs.m" almacenado en la carpeta "Simulator-automatic-2UAVs" o el mismo procedimiento para múltiples UAVs. Finalmente, cada ejecución nos mostrará dos ventanas (para el "Paso 4"):

- Ventana de simulación: nos permite visualizar la trayectoria de los UAV.
- Ventaja del algoritmo: nos permite visualizar el comportamiento del algoritmo. Se aconseja desactivar esta opción, en caso de simulaciones con escenarios densos.

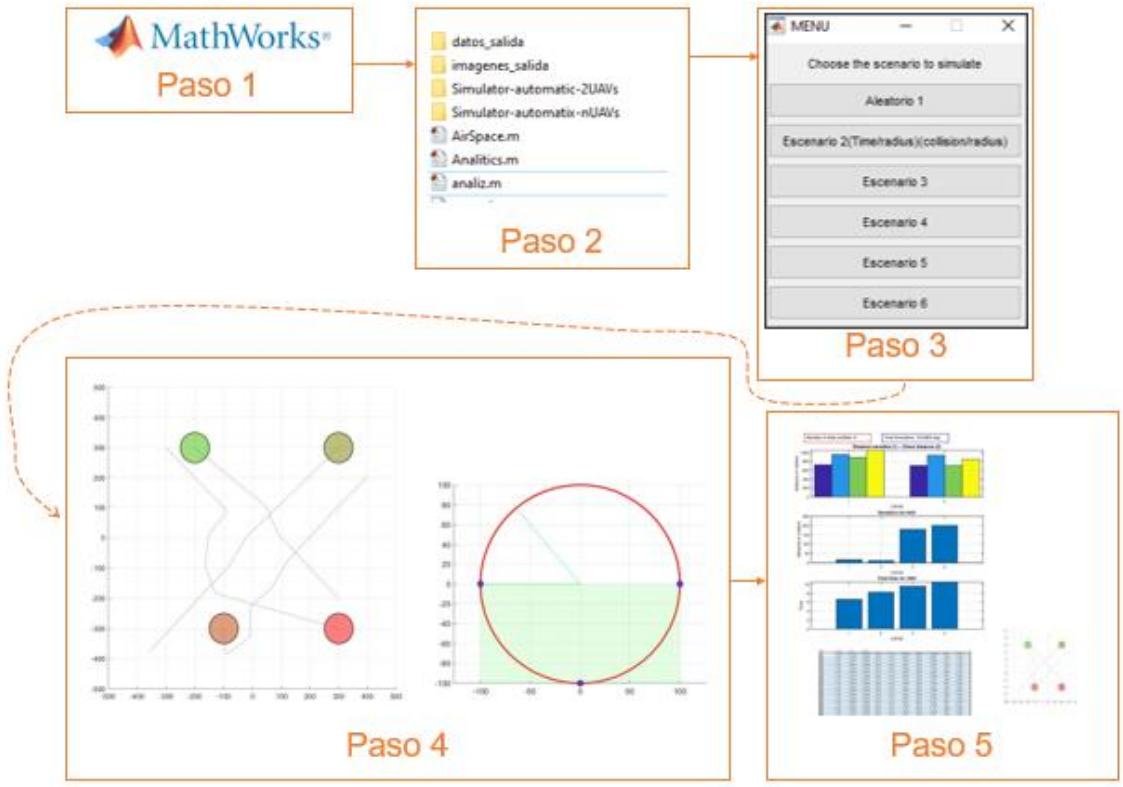

Figura 61. Pasos para el uso de la herramienta "Analizador"

A.2. USO DE LA HERRAMIENTA "SIMULADOR"

A continuación, se describe paso a paso el uso de la herramienta con un ejemplo simple.

- Paso 1: debemos disponer de la herramienta MATLAB, en la que se integra Simulink, en su versión R2019-b (o posterior).
- Paso 2: ejecutar la aplicación "app_UAV.mlapp".
- Paso 3: ingresar los parámetros que nos solicite la aplicación, y seleccionar el botón "Simular".
- Paso 4: los resultados obtenidos son almacenados en archivos .csv.