



**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 colisiones 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 colisiones 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 colisiones 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 14 de junio de 2020

Fdo.: Paloma Sánchez de la Torre

Resumen

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

Este algoritmo ha sido implementado mediante la herramienta MATLAB, para 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, ya que se consideran escenarios congestionados (con cientos de drones).

Para la visualización y obtención de datos, se ha desarrollado un analizador a través de MATLAB, que permita ejecutar el algoritmo y poder graficar cada paso de este a la vez que visualizamos el vuelo de los diferentes UAV involucrados. Los datos obtenidos, para simulaciones con dos o múltiples UAV, permiten el estudio y análisis del comportamiento del algoritmo. Además, se ha desarrollado un simulador mediante Simulink y MATLAB, que permita la ejecución de escenarios en tres dimensiones.

Agradecimientos

A mi abuela, por apoyarme en todas mis locuras y por esa confianza que tiene en mí.

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](#_Toc42084031)

[Resumen v](#_Toc42084032)

[Agradecimientos vii](#_Toc42084033)

[CAPÍTULO 1. Introducción 1](#_Toc42084034)

[1.1. Motivación 1](#_Toc42084035)

[1.2. Objetivos 3](#_Toc42084036)

[1.3. Estructura de la memoria 4](#_Toc42084037)

[CAPÍTULO 2. Estado del arte 5](#_Toc42084038)

[2.1. Planteamiento general del problema 6](#_Toc42084039)

[2.2. Técnicas de evitación de colisiones entre UAV 8](#_Toc42084040)

[2.3. Algunas técnicas propuestas 9](#_Toc42084041)

[2.3.1. Artificial Potencial Field (APF) 9](#_Toc42084042)

[2.3.2. Particle Swarm Optimization (PSO) 10](#_Toc42084043)

[2.3.3. Collision Cone (CC) 11](#_Toc42084044)

[2.3.4. Dubins Paths 12](#_Toc42084045)

[2.3.5. Optimal Reciprocal Collision Avoidance (ORCA) 13](#_Toc42084046)

[2.3.6. Conclusión 15](#_Toc42084047)

[CAPÍTULO 3. Herramientas y metodología 16](#_Toc42084048)

[3.1. Herramientas empleadas 16](#_Toc42084049)

[3.1.1. Matlab 16](#_Toc42084050)

[3.1.2. Simulink 17](#_Toc42084051)

[3.1.3. App Designer 18](#_Toc42084052)

[3.1.4. Github 18](#_Toc42084053)

[3.1.5. Git Bash 18](#_Toc42084054)

[3.1.6. Mendeley 19](#_Toc42084055)

[3.2. Metodología 19](#_Toc42084056)

[CAPÍTULO 4. Algoritmo propuesto (BBCA) 21](#_Toc42084057)

[4.1. Aproximación al problema 21](#_Toc42084058)

[4.2. Mecanismo para detección de colisiones 22](#_Toc42084059)

[4.3. Mecanismo para evitación de colisiones 23](#_Toc42084060)

[4.3.1. Entradas 24](#_Toc42084061)

[4.3.2. Velocidad válida 25](#_Toc42084062)

[4.3.3. Obstáculo de velocidad 26](#_Toc42084063)

[4.3.4. Velocidad óptima 28](#_Toc42084064)

[4.4. Escenarios 31](#_Toc42084065)

[CAPÍTULO 5. Simulador 34](#_Toc42084066)

[5.1. Entidades involucradas en la simulación 34](#_Toc42084067)

[5.2. Subsistema UAV 35](#_Toc42084068)

[5.2.1. Dynamics 36](#_Toc42084069)

[5.2.2. Pilot 37](#_Toc42084070)

[5.2.3. Radar 40](#_Toc42084071)

[5.3. Subsistema ATC 40](#_Toc42084072)

[5.4. Interfaz de usuario 42](#_Toc42084073)

[CAPÍTULO 6. Experimentos y resultados 43](#_Toc42084074)

[6.1. Análisis del algoritmo BBCA 43](#_Toc42084075)

[6.2. Estudio con dos UAV 44](#_Toc42084076)

[6.2.1. Análisis del tiempo con variación del radio 44](#_Toc42084077)

[6.2.2. Análisis de colisiones con variación de radio 45](#_Toc42084078)

[6.2.3. Análisis de colisiones con variación de velocidad máxima 47](#_Toc42084079)

[6.2.4. Análisis de escenario realista 49](#_Toc42084080)

[6.2.5. Escenario realista con impacto total 51](#_Toc42084081)

[6.3. Estudio con múltiples UAV 55](#_Toc42084082)

[CAPÍTULO 7. Conclusiones y propuestas 57](#_Toc42084083)

[7.1. Conclusiones 57](#_Toc42084084)

[7.2. Trabajo futuro 57](#_Toc42084085)

[Bibliografía 59](#_Toc42084086)

[Anexos 63](#_Toc42084087)

[A.1. Ejemplo de uso de la herramienta 63](#_Toc42084088)

[A.2. Manual de usuario 63](#_Toc42084089)

Índice de Figuras

[Figura 1. UAV 1](#_Toc42103557)

[Figura 2. UAV de mensajería en DHL 2](#_Toc42103558)

[Figura 3. Globo aerostático de 1849 5](#_Toc42103559)

[Figura 4. Sistema Sense, Detect and Avoidance 7](#_Toc42103560)

[Figura 5. Esquema de APF 10](#_Toc42103561)

[Figura 6. Ejemplo PSO con dos mínimos diferentes [23] 11](#_Toc42103562)

[Figura 7. Ejemplo de cono de colisión [24] 12](#_Toc42103563)

[Figura 8. Ejemplo de Dubins Path [24] 13](#_Toc42103564)

[Figura 9. Dos agentes virtuales (izquierda), el obstáculo de velocidad truncado VO (centro) y semiplanos libres de colisiones (derecha) [27] 14](#_Toc42103565)

[Figura 10. MATLAB logo 16](#_Toc42103566)

[Figura 11. Simulink & MATLAB logo 17](#_Toc42103567)

[Figura 12. GitHub logo 18](#_Toc42103568)

[Figura 13. Mendeley logo 19](#_Toc42103569)

[Figura 14. Mecanismo para detectar colisiones 23](#_Toc42103570)

[Figura 15. Pseudocódigo del algoritmo 24](#_Toc42103571)

[Figura 16. Velocidad válida inicial en simulador 25](#_Toc42103572)

[Figura 17. Velocidad válida truncada 26](#_Toc42103573)

[Figura 18. Obstáculo velocidad en simulador (agente 1 con 3) 27](#_Toc42103574)

[Figura 19. Semiplano de obstáculo velocidad (agente 1 con 3) 27](#_Toc42103575)

[Figura 20. Ampliación de obstáculo según vector velocidad (agente 1 con 3) 28](#_Toc42103576)

[Figura 21. Cálculo de velocidad óptima 28](#_Toc42103577)

[Figura 22. Ocho puntos de corte posibles 29](#_Toc42103578)

[Figura 23. Cuatro puntos en esquinas 29](#_Toc42103579)

[Figura 24. Elección de velocidad óptima 30](#_Toc42103580)

[Figura 25. Elección del punto con ángulos iguales 30](#_Toc42103581)

[Figura 26. Simulación del caso ‘ángulos iguales’ 31](#_Toc42103582)

[Figura 27. Menú de escenarios para simular 32](#_Toc42103583)

[Figura 28. Entradas para simulación aleatoria 32](#_Toc42103584)

[Figura 29. Elemento UAV en el simulador 35](#_Toc42103585)

[Figura 30. Elemento ATC en el simulador 35](#_Toc42103586)

[Figura 31. Subsistema UAV en Simulink 36](#_Toc42103587)

[Figura 32. Dynamics en subsistema UAV 37](#_Toc42103588)

[Figura 33. Máquina de estado de UAV 38](#_Toc42103589)

[Figura 34. Bloque comunicación de UAV 39](#_Toc42103590)

[Figura 35. Esquema tratamiento de mensajes en UAV 40](#_Toc42103591)

[Figura 36. Radar en subsistema UAV 40](#_Toc42103592)

[Figura 37. Bloque control de ATC 41](#_Toc42103593)

[Figura 38. Esquema tratamiento de mensajes ATC 41](#_Toc42103594)

[Figura 39. Interfaz de usuario del simulador 42](#_Toc42103595)

[Figura 40. Porcentaje de tiempo con variación de radio 45](#_Toc42103596)

[Figura 41. Porcentaje de colisiones evitadas con variación de radio 46](#_Toc42103597)

[Figura 42. Porcentaje de colisiones evitadas según la velocidad máxima 48](#_Toc42103598)

[Figura 43. Simulación de escenario 1 y 2 (izquierda) y escenario 3 y 4 (derecha) 48](#_Toc42103599)

[Figura 44. Colisión entre 2 UAV sin empleo de BBCA 50](#_Toc42103600)

[Figura 45. Simulación de escenario con 2 UAV empleando BBCA(izquierda) y la desviación realizada en la simulación (derecha) 50](#_Toc42103601)

[Figura 46. Distancia recorrida con BBCA vs distancia directa sin BBCA 51](#_Toc42103602)

[Figura 47. Porcentaje de desviación de la ruta inicial 51](#_Toc42103603)

[Figura 48. Incremento del tiempo en porcentaje por el uso de BBCA vs sin BBCA 51](#_Toc42103604)

[Figura 49. Grados en una circunferencia 52](#_Toc42103605)

[Figura 50. Conflictos evitados según los grados de las rutas de los UAV y vel\_max 53](#_Toc42103606)

[Figura 51. Aumento de tiempo variando el radio con vel\_max = 130 km/h 53](#_Toc42103607)

[Figura 52. Media de conflictos evitados variando el radio con vel\_max = 130 km/h 54](#_Toc42103608)

[Figura 53. Media de conflictos evitados según vel\_max y radio 55](#_Toc42103609)

[Figura 54. Pasos para el uso de la herramienta “Analizador” 64](#_Toc42103610)

Índice de Tablas

[Tabla 1. Pseudocódigo para generar un escenario aleatorio 33](#_Toc42031976)

[Tabla 2. Porcentaje de tiempo con variación de radio 45](#_Toc42031977)

[Tabla 3. Colisiones con variación de radio 46](#_Toc42031978)

[Tabla 4. Colisiones con variación de velocidad máxima 48](#_Toc42031979)

[Tabla 5. Un ejemplo de tabla 63](#_Toc42031980)

2. Introducción

A lo largo de esta memoria se refleja el trabajo realizado en el Trabajo Fin de Grado *“Gestión dinámica de colisiones 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. 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, por sus siglas en inglés, *Unmanned Aerial Vehicle*), comúnmente conocidos como drones. Son aeronaves no tripuladas que sobrevuelan el espacio aéreo, manejadas remotamente. Por otro lado, tendremos como protagonista al software que permitirá la gestión de la que veníamos hablando.



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 como Amazon o DHL ya tienen propuestas en este campo.



1. 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. Como ya se comentaba, 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 escenario, 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 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 dispondría de 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. Objetivos

El objetivo principal 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 conseguir 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 simulación que permita desarrollar y probar nuevos mecanismos de evitación de colisiones en este escenario.
3. Diseñar un algoritmo sencillo de evitación de colisiones.
4. Llevar a cabo una evaluación comparativa de las prestaciones del nuevo algoritmo. En su caso, proponer mejoras al mismo.

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.

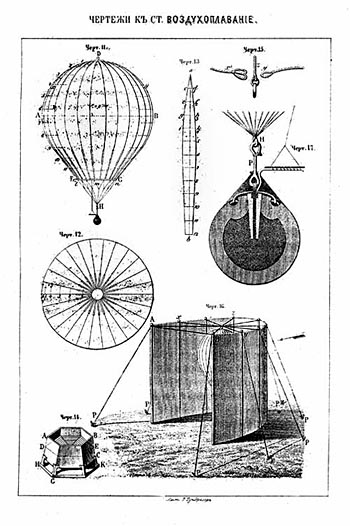
* 1. Estructura de la memoria

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 como 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 existente de los UAV y los algoritmos/métodos existentes en la literatura.
* Capítulo 3: se mencionan las herramientas principales empleadas para el desarrollo del proyecto en su conjunto.
* Capítulo 4: se aborda el analizador empleado para el testeo del algoritmo, así como el funcionamiento del propio mecanismo para evitación de colisiones.
* Capítulo 5: se corresponde con el simulador desarrollado, describiendo cada parte que lo compone, así como su comportamiento.
* Capítulo 6: se trata del análisis y estudios de los resultados obtenidos en las simulaciones del analizador.
* Capítulo 7: se describen las conclusiones obtenidas y el trabajo futuro.

1. 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.



1. 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 del dron, en los materiales o 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”.*

El 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.

* 1. Planteamiento general del problema

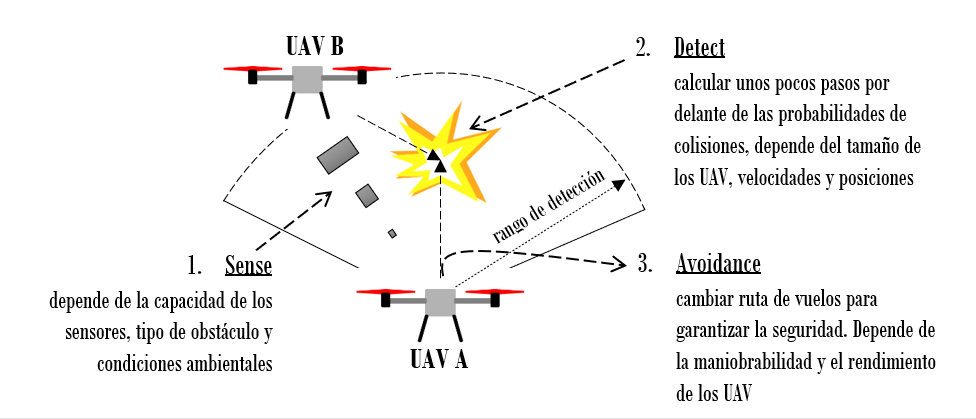
Es evidente que la gestión de los posibles obstáculos que un UAV puede encontrarse durante su vuelo es crucial para el éxito de su misión. La evitación de obstáculos ha sido un problema muy estudiado por la comunidad científica, destacando su aplicación a robots móviles terrestres. En este contexto se 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 (Air Traffic Management, ATM), 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.



1. 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.

* 1. 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 (“Collision Cone”, CC) y obstáculo de velocidad (“Velocity Obstacle”, VO) [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].

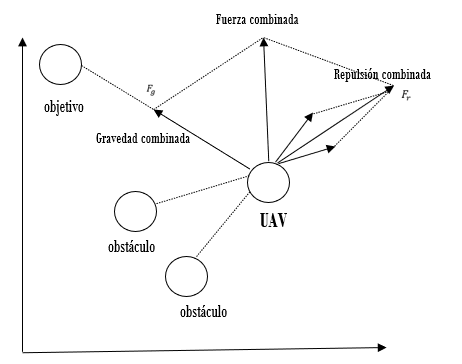
Otras técnicas de evitación de colisiones muy populares en este escenario son las basadas en fuerzas. Es el caso de los campos potenciales artificiales (“Artificial Potential Field”, APF) [17], donde el destino del UAV ejerce una fuerza de atracción, pero otros UAV en su trayectoria hacia dicho destino 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 (“Particle Swarm Optimization”, PSO) [19] o las basadas en colonias de hormigas (“Ant Colony Optimization”, ACO) [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.

* 1. Algunas técnicas propuestas
     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, donde se emplean 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 robot o “partícula” se mueve frente a unos obstáculos, en función de distintas fuerzas hacia su objetivo.



1. 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 (de sus siglas en inglés, “*Artificial Potential Field*”).

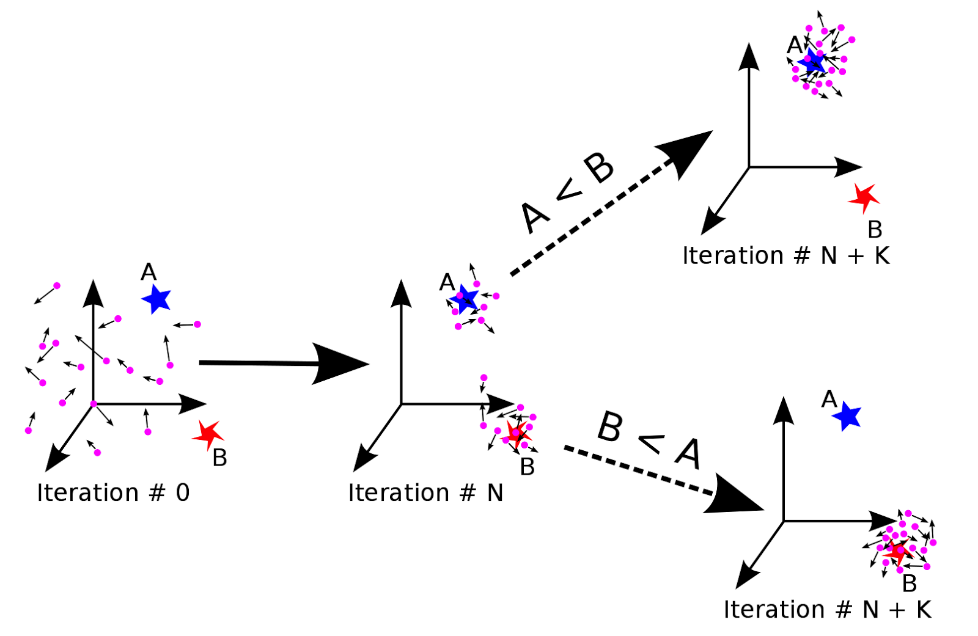
Estos métodos se basan en la idea que Khatib introdujo en 1986, donde el agente se mueve atraído en un campo de fuerzas hacia su objetivo final y el resto de los agentes se comportan como obstáculos con fuerzas repulsivas [21].

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.

* + 1. Particle Swarm Optimization (PSO)

Este método, conocido como optimización por enjambre de partículas (PSO, de sus siglas en inglés, “*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].



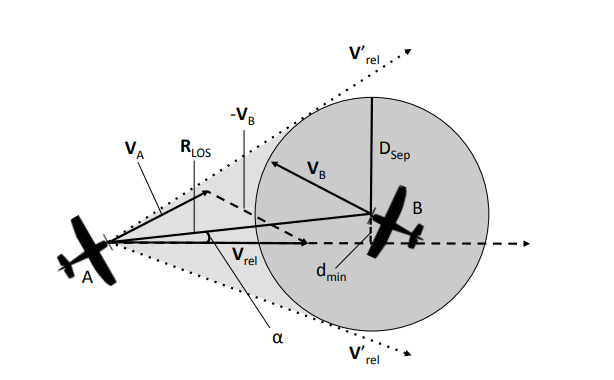
1. Ejemplo PSO con dos mínimos diferentes [23]

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.

* + 1. Collision Cone (CC)

El método conocido como cono de colisión (del inglés, “*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, donde se establece la probabilidad de colisión entre dos agentes en movimiento en función de sus velocidades [12].

El avance 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, donde el sistema de dos agentes es tratado como un sistema de un único agente y un obstáculo estático. En la Figura 7 se muestra como B es el obstáculo estático y A es el agente dinámico con velocidad relativa, ,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 (.



1. Ejemplo de cono de colisión [24]

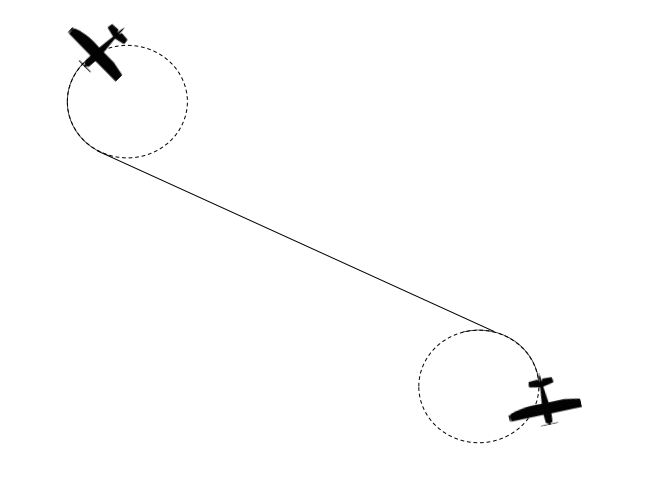
Cuando la  de A sea tangencial al área circular de B, marcada como , los agentes se evitarán con una deviación mínima a su objetivo. A los rangos de velocidades comprendidas entre los dos vectores se le conoce como cono de colisión.

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.

* + 1. 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.



1. 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].

* + 1. Optimal Reciprocal Collision Avoidance (ORCA)

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

Este tipo de métodos 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, bajo esa predicción se establece la nueva velocidad en función de alguna optimización.

ORCA emplea el 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), 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 las velocidades libres de colisiones, emplea la técnica de optimización de programación lineal.

Imagen que contiene paraguas

Descripción generada automáticamente

1. 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.

* + 1. 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 coste computacional y eficiencia. Además, el algoritmo debe enfocarse en observaciones locales y evitar el empleo de intercambio de información entre los UAV involucrados.

1. Herramientas y metodología

Para el desarrollo de este proyecto se ha seguido una metodología ágil…

* 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 de las herramientas existentes sobre las que poder desarrollar e implementar la 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, la eficiencia sobre la solución y la adecuación a las necesidades existentes para el desarrollo del proyecto.

* + 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, por sus siglas en inglés, Integrated Development Environment), con un lenguaje de programación propio basado en matrices.



1. MATLAB logo

Actualmente se encuentra en su versión R2019-b, y, por lo tanto, será la versión empleada en este proyecto.

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, cumplir con las características necesarias para el desarrollo e implementación de la solución. Entre las características que cubre se encuentran: la generación de scripts que permiten graficar y analizar los datos obtenidos, implementación de una interfaz que permita la simulación de cada escenario, la representación de modelos para la implementación del simulador mediante diagramas, etc.

A continuación, se exponen las dos principales herramientas que nos ofrece MATLAB y que han sido empleadas para el desarrollo del simulador.

* + 1. Simulink

Simulink es una herramienta de simulación de modelos o 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.



1. Simulink & MATLAB logo

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 nuestro sistema dinámico del que se compone este trabajo. Y es que Simulink posibilita la ejecución de miles de simulaciones en paralelo, lo que nos permite el análisis y visualización de datos en tiempo récord.

* + 1. 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 una aplicación, como, por ejemplo, botones, listas desplegables, gráficas, etc. [3].

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

* + 1. 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.



1. GitHub logo

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.

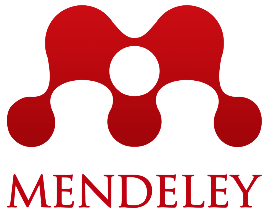
Por lo tanto, este proyecto será mantenido a través de GitHub en el repositorio almacenado en [https://github.com/PalomaSanx/UAVimulation\_TFG](https://github.com/PalomaSanx/UAVsimulation_TFG).

* + 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].

* + 1. Mendeley

Mendeley es un gestor de referencias bibliográficas, el cual extrae de forma automática los metadatos de las distintas webs de interés, así como aquella documentación objeto de ser referenciadas en un documento.



1. Mendeley logo

Por lo tanto, esta herramienta ha sido de gran ayuda en esta labor, para la elaboración de esta memoria.

* 1. Metodología

klasnldasnf

1. Algoritmo propuesto (BBCA)

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 mecanismo 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 mediante la herramienta MATLAB, en la que se ha desarrollado un analizador que permita visualizar la simulación de una forma sencilla mostrando las trayectorias llevadas a cabo por cada uno de los UAV, los distintos aspectos contemplados en el mismo (velocidades válidas, obstáculos de velocidad, velocidad objetivo, velocidad óptima, etc.) así como la comunicación de mensajes en caso de colisión u otro aspecto relevante. Con la finalidad de poder analizar los resultados obtenidos.

* 1. Aproximación al problema

Como punto de partida inicial se han definido una serie de criterios que son indispensables para abordar el problema:

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 es comunicar un conflicto que va a ocurrir en un futuro. De la misma forma puede considerarse una alerta como la proximidad a un obstáculo terrestre o advertir 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 las posiciones y velocidades.
* Un modelo de trayectorias dinámicas: para poder proyectar los estados en el futuro y predecir si existirá un conflicto. Como, por ejemplo, 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. Se 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.

Este mecanismo será implementado, partiendo de la hipótesis de que los UAV colisionan cuando se encuentren en vuelo, es decir, será diseñado para escenarios en dos dimensiones. Por lo tanto, se considera que los UAV sobrevuelan el espacio aéreo a una misma altura, donde ‘*z’* es constante.

* 1. Mecanismo para detección de colisiones

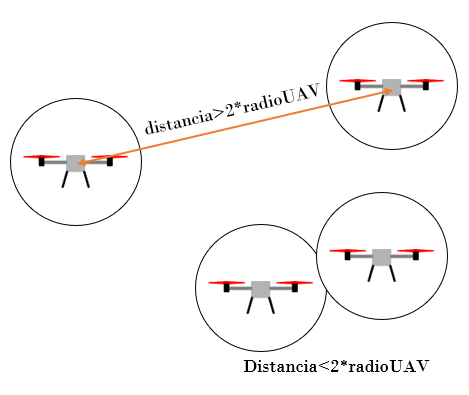
Como ya se había comentado con anterioridad, se entiende por colisión la situación en la que dos o más agentes superan un umbral de seguridad preestablecido. Por simplicidad, cada UAV es modelado mediante una circunferencia cuyo radio coincide con dicho margen de seguridad.

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 *=(x1, y1)* y *=(x2, y2)*.

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

En la Figura 14 puede observarse de forma clara cuando se produce dicha colisión.



1. Mecanismo para detectar colisiones

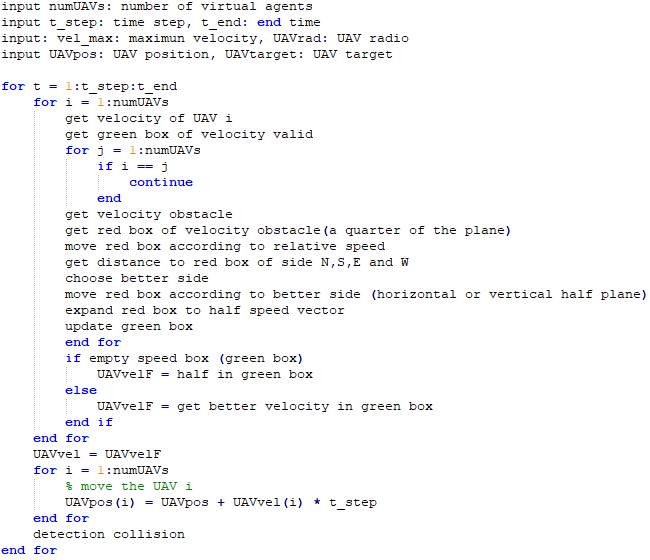
Como se puede observar este mecanismo detecta cuando se produce una colisión de forma fiable. Pero existe una situación en la que este mecanismo puede no llegar a detectar la misma, que se produce cuando ambos se sobrepasan (es decir, colisionan) y el mecanismo realiza la comprobación un instante después. Por ello, se ha incluido una función que devuelve el instante en el que se producirá la colisión, pasando como parámetros la información actual de los UAV (posiciones, velocidades y radio). De esta manera, además de la comprobación anteriormente descrita, se comprueba si dicha colisión ha ocurrido, aunque estos se sobrepasen.

* 1. Mecanismo para evitación de colisiones

El mecanismo que permite evitar las colisiones entre los distintos UAV que sobrevuelan el espacio aéreo con un determinado fin, como sería el de entrega de paquetería a diferentes destinos, es una de las partes más importantes e interesantes que engloban a 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 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). Se presenta en la Figura 15 el pseudocódigo de este mecanismo.



1. Pseudocódigo del algoritmo

El mecanismo requiere inicialmente de la generación de un escenario sobre el que actuar, este escenario se genera con una serie de entradas o inputs mínimos. Como se muestra en la Figura 15, el mecanismo ejecuta una serie de funciones que trabajan principalmente con dos elementos:

* Velocidades válidas: estas son modeladas como una caja de color verde.
* Obstáculo de velocidad: modelado como una caja de color rojo.

A continuación, se describen los elementos y cada una de las funciones de las que se compone.

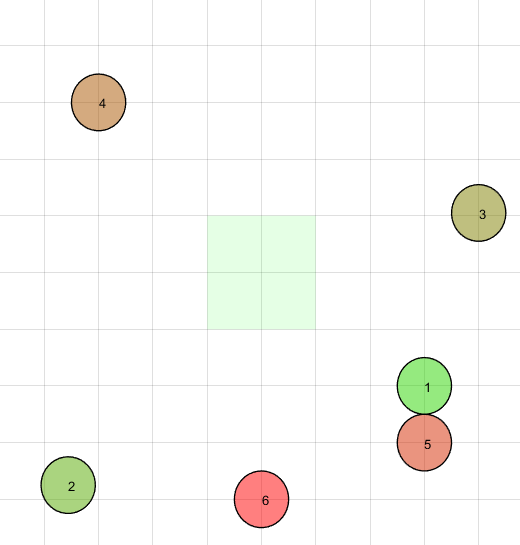
* + 1. Entradas

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 o UAV
* Posición de inicio de cada agente o UAV
* Velocidad máxima
* Posición de destino de cada agente o UAV
* Radio de los UAV
  + 1. Velocidad válida

Como se puede observar en la Figura 16 el modelado de las velocidades válidas se expresa como una caja verde que dispone de cuatro lados: Norte (N), Sur (S), Este (E) y Oeste (W), que permite visualizar las velocidades permitidas para un UAV.

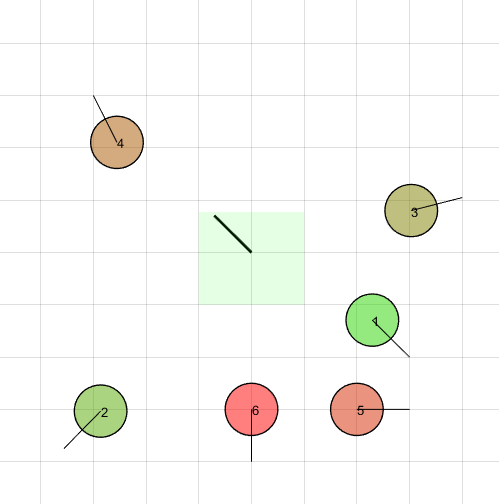
Inicialmente se configura por medio de la velocidad máxima y se actualiza en función de las restricciones introducidas.



1. Velocidad válida inicial en simulador

Una vez construido el obstáculo velocidad (caja roja), las velocidades permitidas (caja verde) son actualizadas.

Dados dos conjuntos R = {valores obstáculo velocidad} y G = {valores de velocidades válidas}:



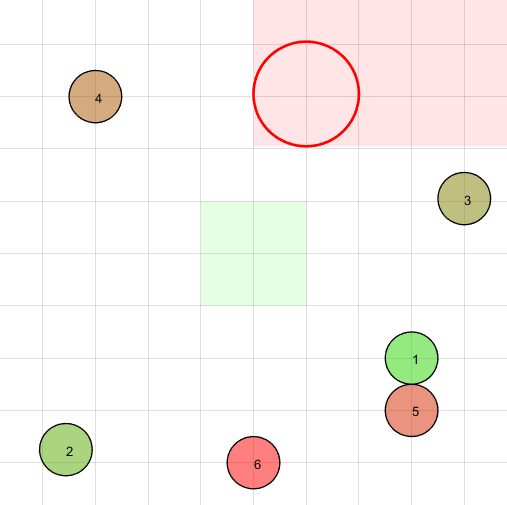
1. Velocidad válida truncada

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

* 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) como el centro de dichas velocidades válidas.
* Caja de velocidades válidas no se encuentra vacía: en este caso la velocidad futura se establece mediante el cálculo de la velocidad óptima, entre las posibles.
  + 1. Obstáculo de velocidad

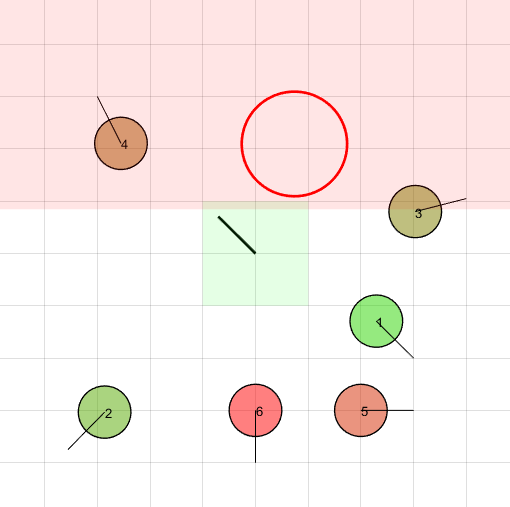
De la misma forma en la que se ha modelado la velocidad permitida o velocidad válida, se ha implementado el obstáculo de velocidad, pero en este caso mediante una caja roja con lados Norte (N), Sur (S), Este (E) y Oeste (W).

En primera instancia se genera el obstáculo de velocidad circular, este se crear en función de la posición y el radio 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.



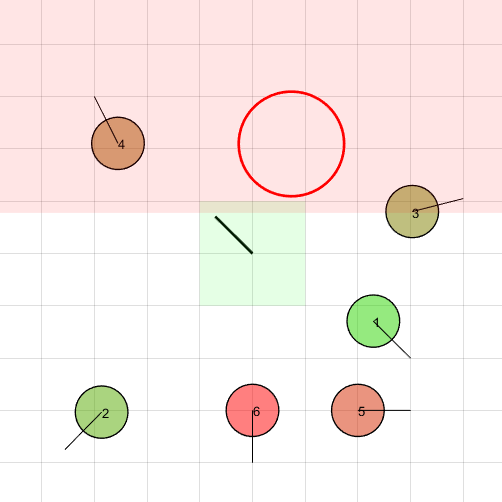
1. Obstáculo velocidad en simulador (agente 1 con 3)

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.



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

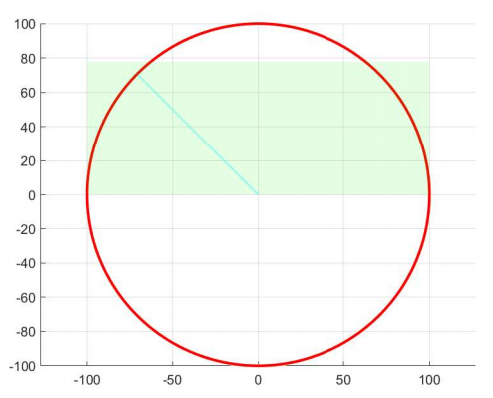
El tercer paso se basa en ampliar dicha restricción (obstáculo velocidad) la mitad del vector velocidad del agente con la dirección óptima calculada anteriormente.



1. Ampliación de obstáculo según vector velocidad (agente 1 con 3)
   * 1. Velocidad óptima

La elección de la velocidad óptima se realiza en función de las velocidades válidas disponibles en dicho instante de tiempo, vistas en el apartado “Velocidad válida”.

Para la visualización de este proceso, se incluyen en el simulador 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 21.

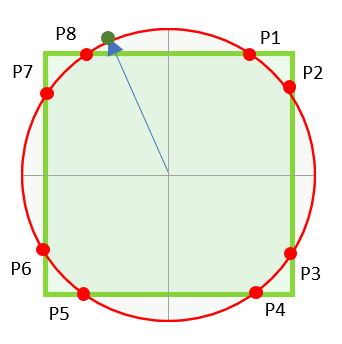


1. Cálculo de velocidad óptima

Esta función cuenta con dos posibles casos que son tratados de forma independiente:

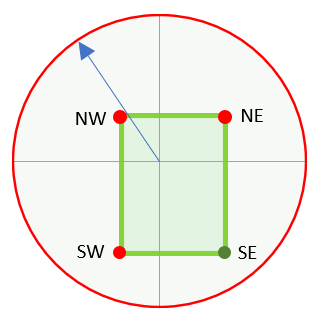
* La velocidad directa al objetivo se encuentra dentro de las velocidades válidas, por lo tanto, se establece como mejor opción dicha velocidad.
* La velocidad directa al objetivo no está dentro de las velocidades válidas: se trata mediante un método que hemos denominado “método de los doce puntos”.

El método de los “doce puntos” consiste en tratar ocho puntos posibles de corte entre la caja verde de velocidades válidas y la velocidad válida circular.



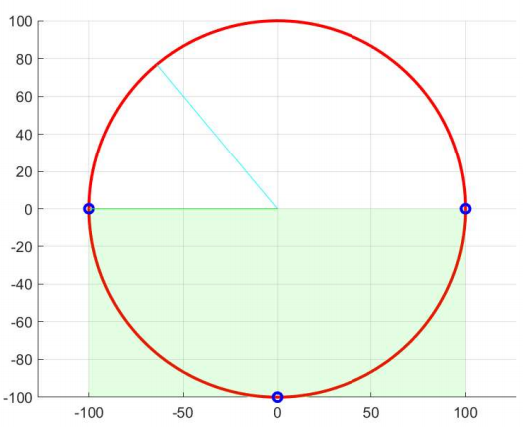
1. Ocho puntos de corte posibles

Por otro lado, se consideran otros cuatro puntos posibles, pertenecientes a las esquinas. Dichas esquinas serán consideradas si se encuentran dentro de la velocidad válida circular, que se encuentra dentro del rango de la velocidad máxima permitida.



1. Cuatro puntos en esquinas

Calculados los doce puntos posibles, se determina cuál de ellos es el óptimo. Esto se realiza mediante el ángulo de dichos puntos respecto a la velocidad directa del agente a su objetivo. De esta manera, se dispone de la velocidad óptima para dicho instante de tiempo, como aquel de los puntos cuyo ángulo sea menor.



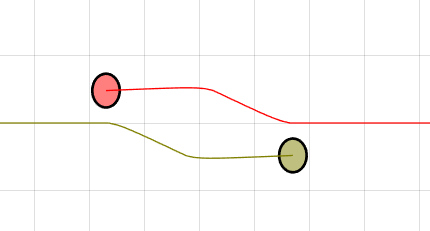
1. Elección de velocidad óptima

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”).



1. 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 25, 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 26.

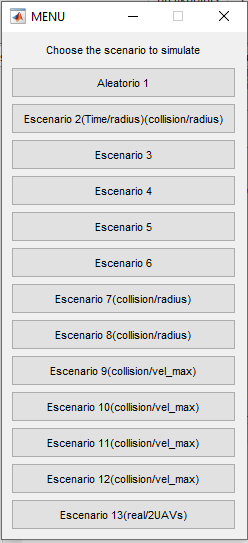


1. Simulación del caso ‘ángulos iguales’
   1. Escenarios

Para poder analizar el comportamiento de un algoritmo, se ha de generar previamente un escenario. Un escenario está compuesto principalmente por los siguientes parámetros:

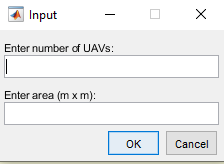
* Posiciones iniciales de UAV
* Posiciones objetivo de UAV
* Radio de UAV
* Velocidad máxima
* Número de UAV

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



1. 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. Esta opción genera el resto de los parámetros de forma aleatoria, permitiendo ejecutar escenarios con alta densidad de agentes.



1. 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, lo hacen las posiciones objetivo de los UAV.

|  |  |
| --- | --- |
|  | |
|  |  |
|  | **For** UAV:numUAV |
|  |  |
|  |  |
|  | **End** |
|  |  |
|  | vel\_max = número aleatorio ([limMin,limMax]) |
|  | UAVrad= número aleatorio ([limMin,limMax]) |
|  | distMin = UAVrad\*2 |
|  |  |
|  |  |
|  | **While** posiciones < distMin |
|  | **For** i:numUAV |
|  | **For** j:numUAV |
|  | **If** posiciones < distMin |
|  |  |
|  |  |
|  | **End** |
|  | **End** |
|  | **End** |
|  | **End** |
|  |  |

1. Pseudocódigo para generar un escenario aleatorio
2. Simulador

A lo largo de este capítulo se describe el desarrollo de un simulador que permita testear los servicios o aplicaciones que ofrecerían los UAV. 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 diferentes herramientas. 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.

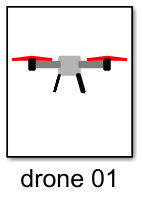
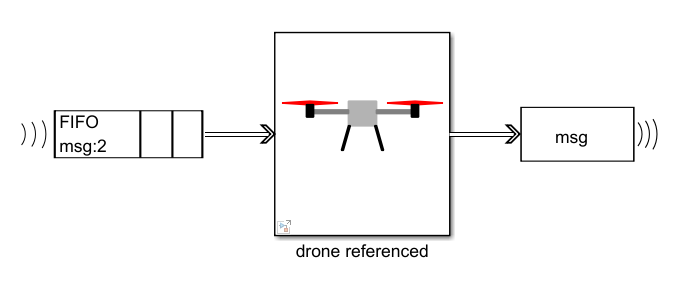
A continuación, se describe el simulador desarrollado, donde se encuentran los objetos/entidades de los que se compone, los distintos subsistemas que lo conforman y su funcionamiento.

* 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, además se encuentra involucrado el controlador de tráfico aéreo (ATC, por sus siglas en inglés, *Air Traffic control*).

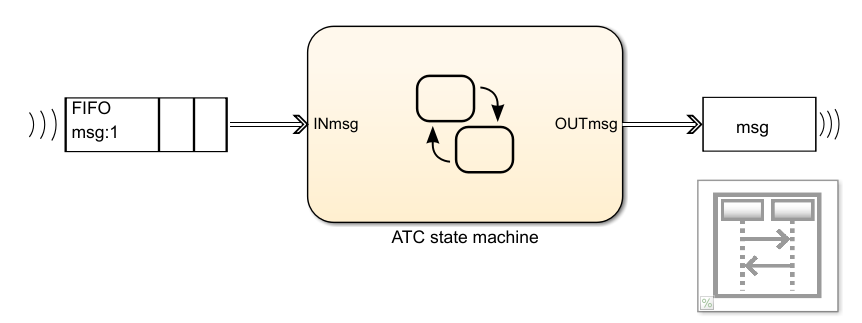
Ambos elementos emiten y reciben mensajes de forma continua. La recepción de los mensajes se realiza con el bloque ‘*Queue’* que facilita Simulink y mediante el método FIFO (por sus siglas en inglés, *First-In First-Out*) se atienden y procesan cada mensaje almacenado.

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

1. Elemento UAV en el simulador

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.

1. Elemento ATC en el simulador

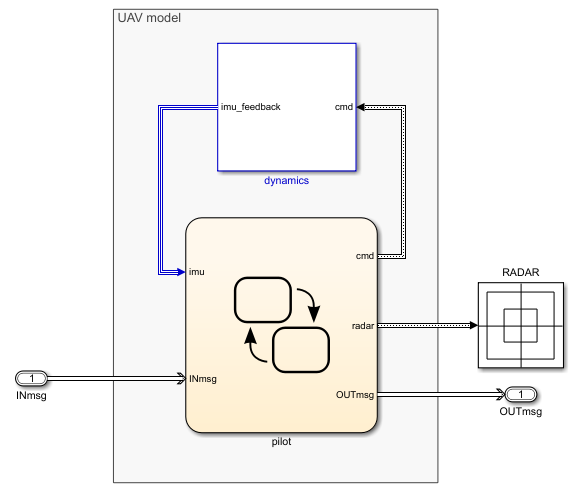
A continuación, se detalla tanto la composición como el funcionamiento de ambos subsistemas para el desarrollo del simulador. Estos subsistemas han sido implementados mediante el entorno gráfico de Simulink que permite construir el modelo a simular mediante bloques.

* 1. Subsistema UAV

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

Los elementos principales que componen este subsistema se pueden observar a través de la Figura 31 y se compone de:

* Dynamics: permite el control de la velocidad, así como la posición.
* 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.

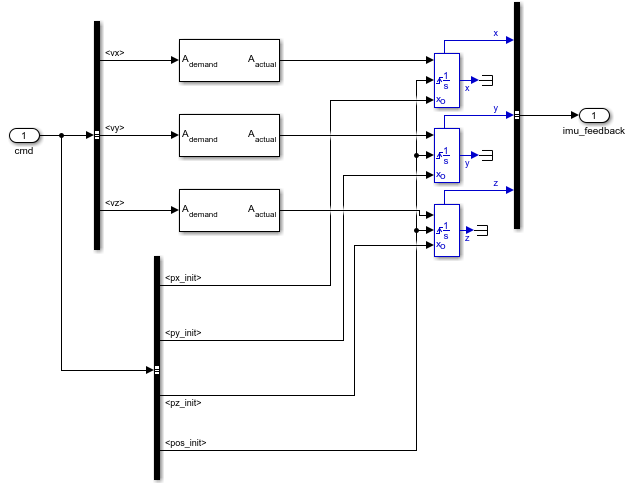


1. Subsistema UAV en Simulink
   * 1. Dynamics

Como se puede observar en la Figura 31 existen 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 una de las direcciones (*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 (*x, y, z*). La posición se emite a través del puerto de salida ‘*imu\_feedback*’.

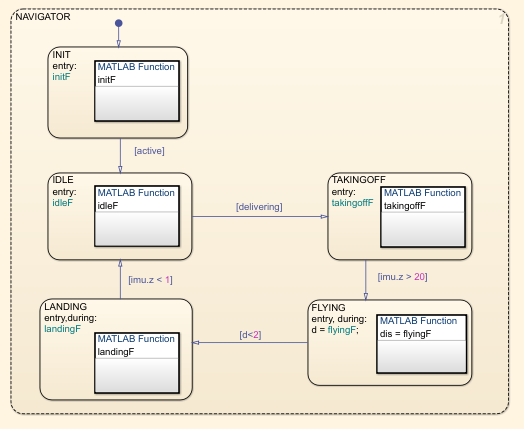
Como se puede observar en la Figura 32 ambos puertos (*cmd* e *imu\_feedback*) son utilizados en ‘*pilot’* que es el corazón del subsistema UAV, contiene la lógica del mismo.



1. Dynamics en subsistema UAV
   * 1. 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 33) y el otro la comunicación (ver Figura 34) de cada UAV.

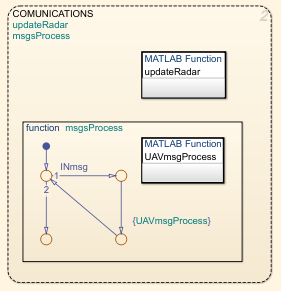


1. Máquina de estado de UAV

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

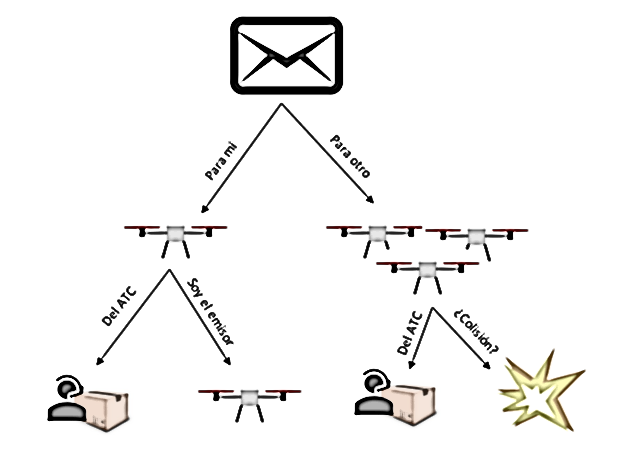
* Init: es el estado inicial y solo se ejecutará la primera vez en cada simulación. A través de este estado se incializará la estructura de estructura de datos local a cada UAV, las posiciones iniciales, el número de UAV involucrados en la simulación y la variable que permite la transición entre estados.
* 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 la coordenada ‘*z’*.
* Flying: es el estado que representa al UAV en vuelo. Principalmente aquí se establece a través del puerto ‘*cmd*’ la velocidad deseada.
* Landing: es el estado en el cual el UAV se encuentra despegando por la llegada a su destino. Nuevamente se establecen las nuevas velocidades, donde las coordenadas ‘*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.



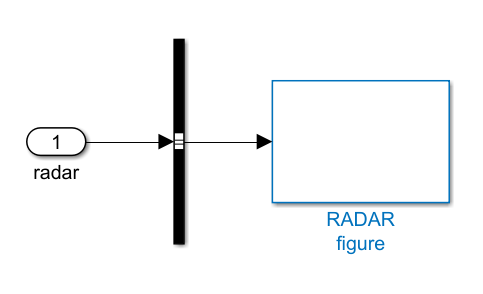
1. Bloque comunicación de UAV

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.



1. Esquema tratamiento de mensajes en UAV
   * 1. Radar

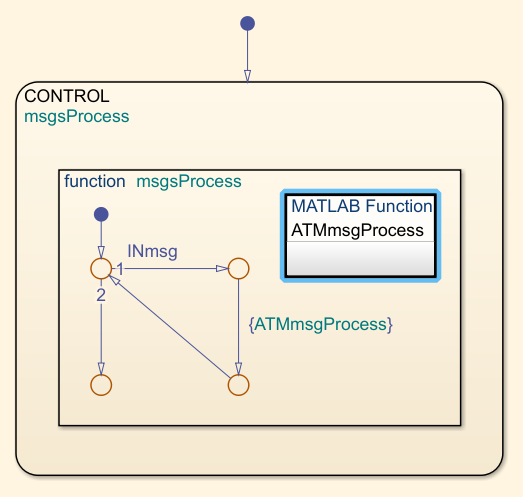
Como puede observarse en la Figura 36, el gráfico 3D que permitirá ver el transcurso de la simulación, es alimentado a través del puerto ‘*radar*’. A través de este puerto se actualiza la información necesaria tal como la posición o el id de cada UAV.



1. Radar en subsistema UAV
   1. Subsistema ATC

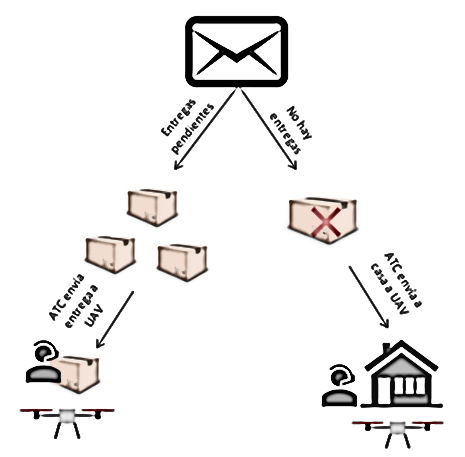
Al igual que ocurre en el subsistema UAV, el ATC también recibe y emite mensajes, principalmente para poder ejecutar órdenes de entrega a cada uno de los UAV.

El subsistema ATC se compone de un único bloque de control (ver Figura 37) que se encarga del procesamiento de dichos mensajes, además inicializa la estructura de datos que maneja el ATC para el control sobre las entregas.



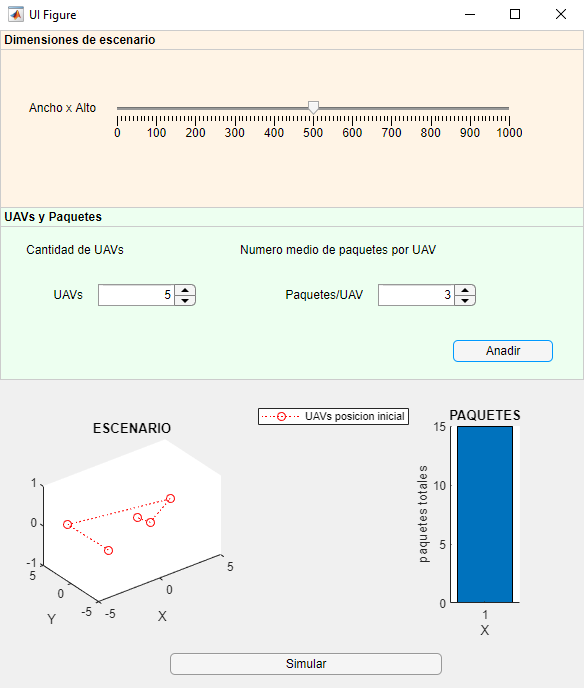
1. Bloque control de ATC

El procesamiento de los mensajes por parte del ATC puede observarse en la Figura 38. Cuando el ATC recibe un mensaje por parte de un UAV comprueba si existen entregas pendientes y de ser así establece una nueva entrega a dicho UAV disponible para ello. De no existir entregas, este envía de regreso al UAV a su punto de partida.



1. Esquema tratamiento de mensajes ATC
   1. Interfaz de usuario

El simulador consta de una serie de parámetros que deben ser inicializados al comienzo de la ejecución. Para el paso de dichos parámetros y así lograr una ejecución personalizada, se ha desarrollado una interfaz de usuario que permita facilitar dicha tarea (ver Figura 39).



1. Interfaz de usuario del simulador
2. Experimentos y resultados

Uno de los aspectos más importantes de este proyecto consiste en el estudio y análisis de los datos obtenidos. Este estudio determinará la viabilidad de la solución, la posible mejora respecto a otros algoritmos, en definitiva, su optimalidad.

Para ello 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 desarrollado para ello, este fue abordado en el Capítulo 4. La metodología empleada para las simulaciones se ha dividido en tres partes:

* Elaboración de escenarios en archivos .m, etiquetados con los parámetros que caracterizan al escenario (ejemplo, “eval8\_2A\_100V\_5R”).
* Elaboración de script para generación de gráficas estadísticas (“Analitics.m”).
* Almacenaje de los datos obtenidos en la simulación en archivos .csv 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.

* 1. Análisis del algoritmo BBCA

El algoritmo “*Bounding Box Collision Avoidance” (BBCA)* desarrollado en este proyecto pretende responder a escenarios realistas, pero para su estudio se ha optado por dos tipos de escenarios:

* Escenario conflictivo: aquí los parámetros no pretenden asemejarse al mundo real, sino que son elegidos en base a su alto potencial de colisión.
* Escenario realista: los parámetros que caracterizan el escenario son elegidos en base a características del mundo real.

De esta manera se dispone del feedback necesario para determinar la optimalidad del algoritmo, que será analizado a lo largo del capítulo. El estudio se divide en dos partes:

* Estudio de escenarios con dos UAV.
* Estudio de escenarios con múltiples UAV.

De esta forma se analiza el comportamiento de forma precisa para dos agentes virtuales, que será extendido en escenarios con múltiples agentes.

* 1. Estudio con dos UAV

En este apartado se analiza en detalle los escenarios compuestos por dos UAV. Cada análisis se caracteriza por el empleo de escenario conflictivo o realista, los datos obtenidos y la posterior conclusión.

* + 1. Análisis del tiempo con variación del radio

A continuación, se muestra el resultado de la simulación de un escenario conflictivo donde se varía el radio de los UAV, para observar el comportamiento del algoritmo en cuanto al incremento de tiempo necesario para llegar a su objetivo en %.

Para ello se ha empleado una velocidad máxima de 100 m/s, el área del escenario es de 1000 m x 1000 m y las posiciones tanto iniciales como finales permanecen estáticas para todas las simulaciones, aunque han sido seleccionadas de forma aleatoria. El escenario elegido es uno de los más habituales en entornos reales, donde los UAV se cruzan en el camino y deben evitarse, de lo contrario se produciría una colisión.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Radio | UAV | Algorithm on | Algorithm off | t (%) |
| 5 | 1 | 09.45 | 09.41 | 00.40 |
| 2 | 08.50 | 08.49 | 00.17 |
| 10 | 1 | 09.45 | 09.41 | 00.40 |
| 2 | 08.50 | 08.49 | 00.17 |
| 15 | 1 | 09.45 | 09.41 | 00.40 |
| 2 | 08.50 | 08.49 | 00.17 |
| 20 | 1 | 09.45 | 09.41 | 00.40 |
| 2 | 08.50 | 08.49 | 00.17 |
| 25 | 1 | 09.45 | 09.41 | 00.40 |
| 2 | 08.50 | 08.49 | 00.17 |
| 30 | 1 | 09.45 | 09.41 | 00.40 |
|  | 2 | 08.50 | 08.49 | 00.17 |
| 35 | 1 | 09.45 | 09.41 | 00.40 |
|  | 2 | 08.55 | 08.49 | 00.76 |
| 40 | 1 | 09.45 | 09.41 | 00.40 |
|  | 2 | 08.60 | 08.49 | 01.33 |
| 45 | 1 | 09.45 | 09.41 | 00.40 |
|  | 2 | 08.90 | 08.49 | 04.66 |
| 50 | 1 | 09.50 | 09.41 | 00.92 |
|  | 2 | 08.85 | 08.49 | 04.12 |

1. Porcentaje de tiempo con variación de radio
2. Porcentaje de tiempo con variación de radio

Como era de esperar, al aumentar el radio de los UAV, el tiempo necesario para llegar a su destino es creciente en su mayoría. Esto ocurre porque en el escenario simulado los UAV deben evitarse y cuanto mayor sea el obstáculo que evitar, mayor será la curva de desviación y, por ende, mayor es el tiempo empleado.

Por otro lado, se observa un valor decreciente, que se debe a la elección de un escenario donde el impacto no es total al ser elegido de forma aleatoria. En posteriores análisis se estudiará el comportamiento con un impacto total en la colisión.

* + 1. Análisis de colisiones con variación de radio

En el apartado anterior se observó el resultado de simular un escenario en el que se incrementaba el radio de los UAV para determinar el aumento de tiempo producido por la desviación del objetivo final de cada uno, para evitar la colisión.

En este apartado se han simulado varios escenarios conflictivos con 2 UAV, en que se varían sus radios, para poder determinar la influencia de este en las colisiones. Como el escenario se compone de 2 UAV, es obvio que el número máximo de colisiones será una, por lo tanto, se determinará en que porcentaje se evitan estas colisiones en función del radio.

Para ello se ha empleado una velocidad máxima de 100 m/s, el área del escenario es de 1000 m x 1000 m y las posiciones tanto iniciales como finales permanecen estáticas en cada escenario para todas las simulaciones. Los escenarios elegidos son de los más habituales en entornos reales, donde los UAV se cruzan en el camino y deben evitarse, de lo contrario se produciría una colisión.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | Radio | Conflict (Algorithm on) | Conflict (Algorithm off) |  |
| 1 | 35 | 0 | 1 | 1 |
| 40 | 0 | 1 | 1 |
| 45 | 0 | 1 | 1 |
| 50 | 0 | 1 | 1 |
| 55 | 0 | 1 | 1 |
| 60 | 0 | 1 | 1 |
| 2 | 35 | 1 | 1 | 0 |
| 40 | 1 | 1 | 0 |
| 45 | 0 | 1 | 1 |
| 50 | 0 | 1 | 1 |
| 55 | 1 | 1 | 0 |
| 60 | 1 | 1 | 0 |
| 3 | 35 | 1 | 1 | 0 |
| 40 | 1 | 1 | 0 |
| 45 | 0 | 1 | 1 |
| 50 | 0 | 1 | 1 |
| 55 | 0 | 1 | 1 |
| 60 | 0 | 1 | 1 |

1. Colisiones con variación de radio
2. Porcentaje de colisiones evitadas con variación de radio

Como se puede observar, dependiendo del valor del radio, se consiguen evitar las colisiones o no. Sin embargo, no existe un orden creciente o decreciente de la gráfica, se observa que dependiendo del radio concreto se evitan o no. Por ello, se ha modificado el valor del intervalo para recalcular la navegación ‘t\_nav\_step = 0.5’, ajustando este en función del radio establecido y ver cómo se comporta.

Para el escenario 2, si se establece t\_nav\_step = 0.2 y en el escenario 3 con t\_nav\_step = 0.1, se logran evitar el 100% de los conflictos. Por lo tanto, el valor establecido a t\_nav\_step es importante en nuestro algoritmo, ya que determinaría su optimalidad. Este parámetro dependerá de varios factores, el cálculo de este debe ser preciso, ya que, según el valor establecido ocasionará que la ruta de cada UAV sea modificada en mayor o menor medida, y, por consiguiente, el tiempo empleado por cada uno.

Según los resultados obtenidos, el cálculo del valor que se establecerá a t\_nav\_step, dependerá de las distancias a las que se encuentren los UAV, el número de UAV y sus radios (entre otros parámetros).

De la misma forma que en el apartado anterior, al ser seleccionados los escenarios de forma aleatorizada, el impacto en la colisión no es del 100%. Este estudio se realizará posteriormente.

* + 1. Análisis de colisiones con variación de velocidad máxima

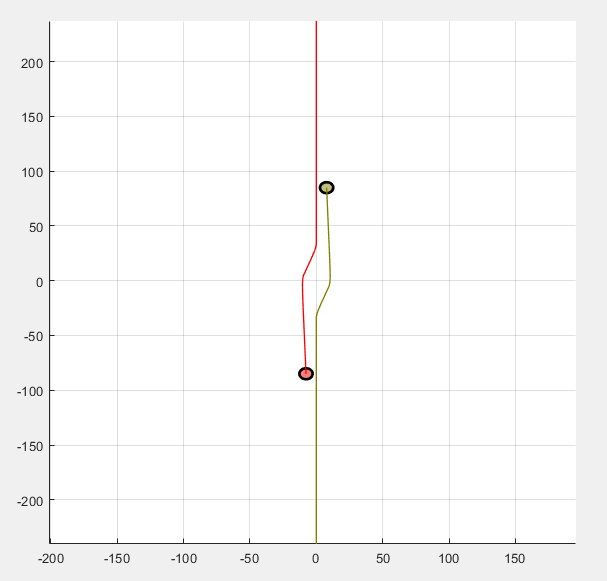
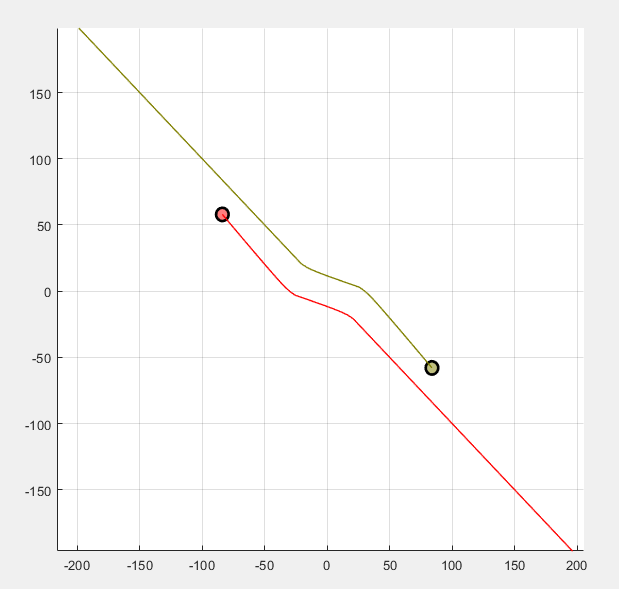
A continuación, se analizan los resultados obtenidos en simulaciones donde la velocidad máxima en cada escenario conflictivo se ha variado, para determinar el porcentaje de colisiones evitadas cuando este valor cambia.

Los escenarios empleados representan el cruce entre 2 UAV que deben evitarse para no colisionar, se ha empleado un radio de 5 m y área 1000 m x 1000 m.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario | Maximum velocity (m/s) | Conflict (Algorithm on) | Conflict (Algorithm off) |  |
| 1 | 10 | 1 | 1 | 0 |
| 20 | 1 | 1 | 0 |
| 40 | 1 | 1 | 0 |
| 60 | 1 | 1 | 0 |
| 80 | 0 | 1 | 1 |
| 100 | 0 | 1 | 1 |
| 2 | 10 | 1 | 1 | 0 |
| 20 | 1 | 1 | 0 |
| 40 | 1 | 1 | 0 |
| 60 | 1 | 1 | 0 |
| 80 | 0 | 1 | 1 |
| 100 | 0 | 1 | 1 |
| 3 | 10 | 0 | 1 | 1 |
| 20 | 0 | 1 | 1 |
| 40 | 0 | 1 | 1 |
| 60 | 0 | 1 | 1 |
| 80 | 1 | 1 | 0 |
| 100 | 1 | 1 | 0 |
| 4 | 10 | 0 | 1 | 1 |
| 20 | 0 | 1 | 1 |
| 40 | 0 | 1 | 1 |
| 60 | 0 | 1 | 1 |
| 80 | 1 | 1 | 0 |
| 100 | 1 | 1 | 0 |

1. Colisiones con variación de velocidad máxima
2. Porcentaje de colisiones evitadas según la velocidad máxima

Como se puede observar el porcentaje en el que se evitan las colisiones para los escenarios simulados es de un 50%, para todas las velocidades establecidas. Esto se debe a que los escenarios 1 y 2, son escenarios donde los UAV se cruzan de forma perpendicular y es a velocidades altas cuando se logra evitar la colisión. Sin embargo, en los escenarios 3 y 4, los cruces son en direcciones opuestas y es en altas velocidades donde no se logra evitar la colisión.



1. Simulación de escenario 1 y 2 (izquierda) y escenario 3 y 4 (derecha)

Para los escenarios 1 y 2, es posible reducir estas colisiones ajustando el valor t\_stab (tiempo de estabilización de velocidad), de tal forma que la curva se abra lo suficiente como para evitar dicha colisión. Por otro lado, el escenario 3 y 4, debe solucionarse disminuyendo el tiempo para recalcular la navegación (t\_nav\_step) y abriendo la curva mediante t\_slab de la misma forma que el escenario 1 y 2. Tras el ajuste de estos parámetros el porcentaje de colisiones evitadas es de un 100%.

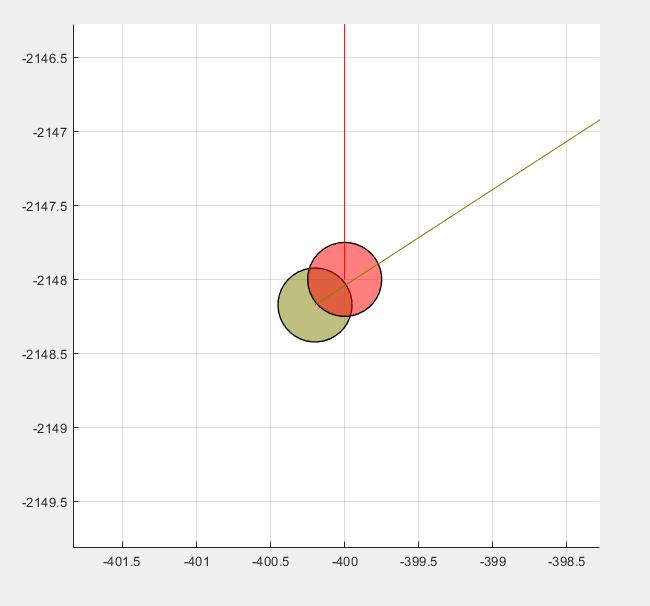
* + 1. Análisis de escenario realista

Uno de los elementos principales de una simulación es representar cada elemento lo más realista posible. 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 del escenario son:

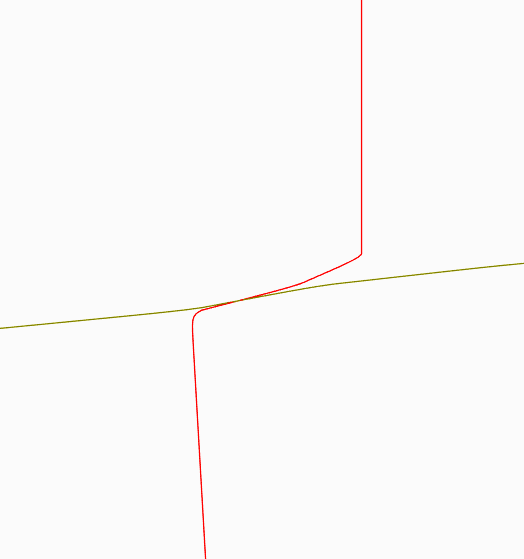
* Área de 10 Km x 10 Km.
* 2 UAV (numUAV).
* UAV con radio 0.25 m (UAVrad).
* Velocidad máxima 100 km/h, equivalente a 27.8 m/s (vel\_max).
* Las rutas suponen el cruce de los UAV.
* Cada paso de simulación en 0.05 s (t\_sim\_step).
* Intervalo para recalcular la navegación (t\_nav\_step) en 0.5.
* Tiempo de estabilización de velocidad (t\_slab) en 0.1 (63%).

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

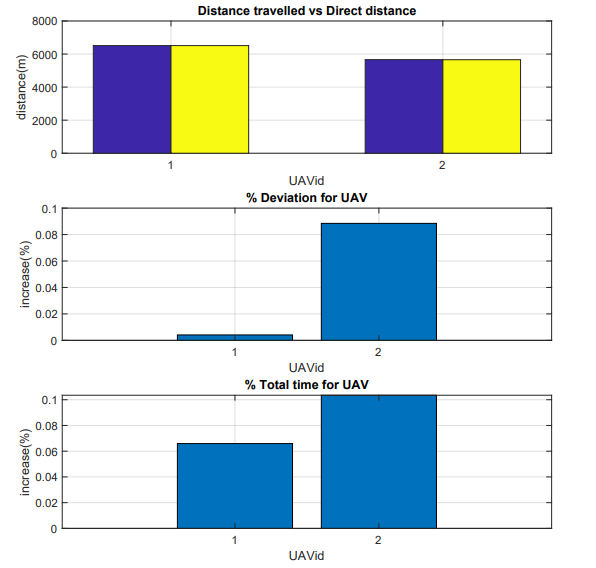


1. Colisión entre 2 UAV sin empleo de BBCA

El objetivo principal del algoritmo es evitar que se produzca la colisión entre los UAV. Para este escenario, dicha colisión ha sido resuelta, obteniendo unos resultados que se muestran a continuación en forma de gráficas.

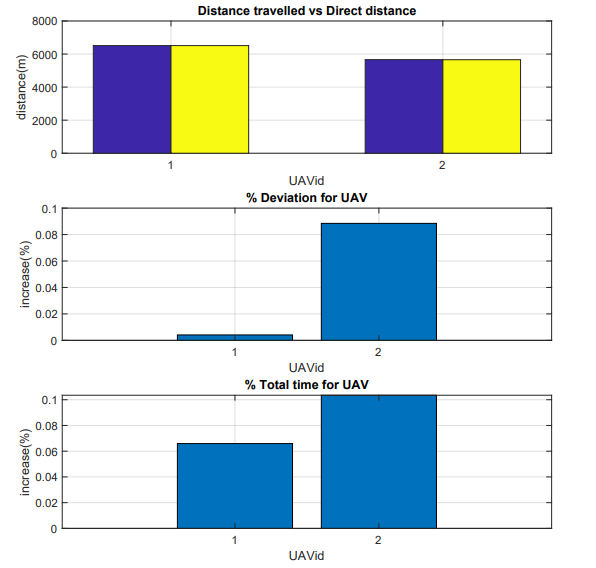


1. Simulación de escenario con 2 UAV empleando BBCA(izquierda) y la desviación realizada en la simulación (derecha)



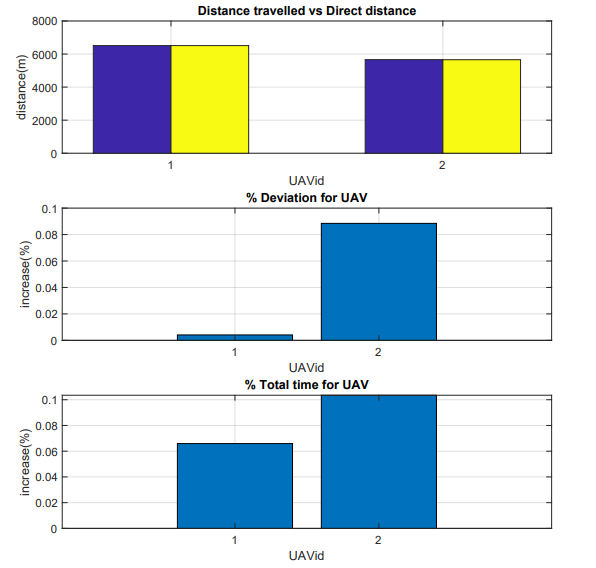
1. Distancia recorrida con BBCA vs distancia directa sin BBCA

Como se puede observar en la Figura 46, la distancia recorrida con el empleo del algoritmo BBCA es similar a la distancia recorrida sin el empleo del algoritmo. Por lo tanto, la desviación del objetivo es mínima, tal y como se observa en Figura 47.



1. Porcentaje de desviación de la ruta inicial

Por otro lado, puede observarse como el tiempo empleado para realizar el recorrido al utilizar el algoritmo, se ve ligeramente incrementado, tal y como se muestra en Figura 48.

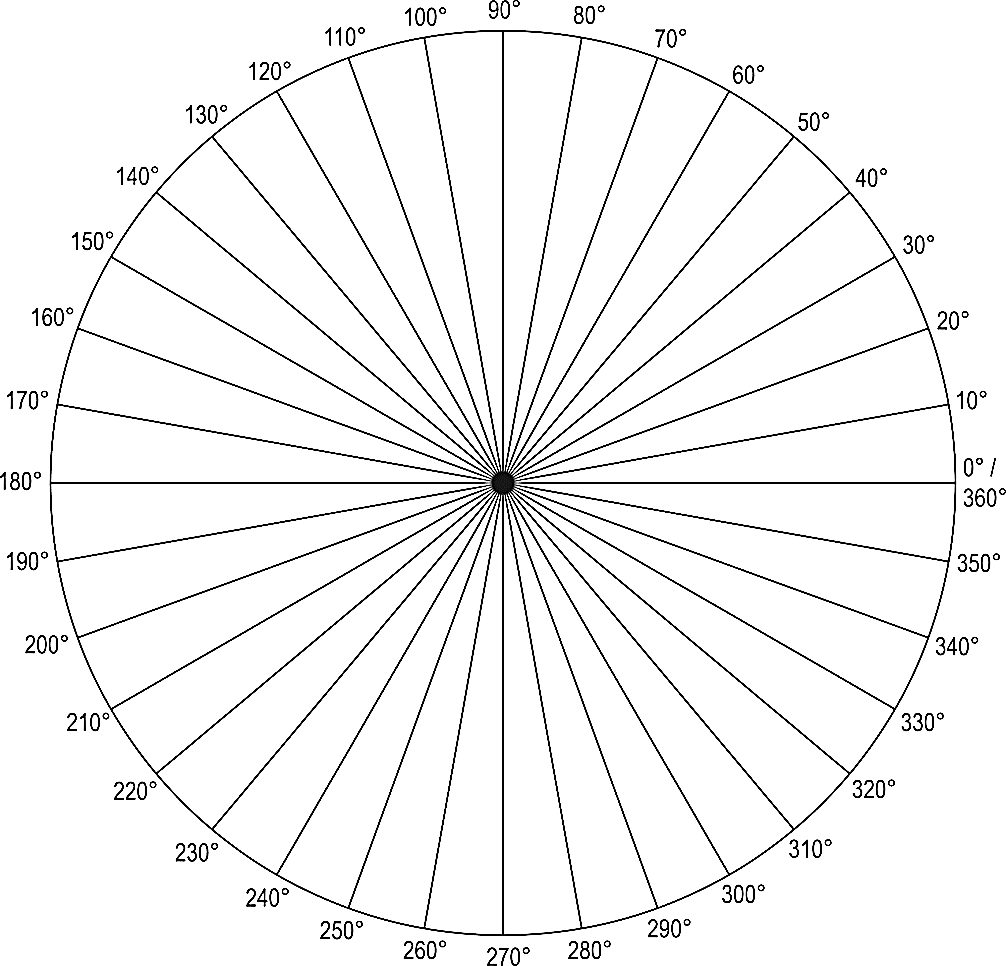


1. Incremento del tiempo en porcentaje por el uso de BBCA vs sin BBCA
   * 1. Escenario realista con impacto total

Cuando hablamos de impacto total, nos referimos a que el punto de colisión es alcanzado por los UAV en el mismo instante de tiempo con la misma masa. Hasta ahora los escenarios conflictivos o realistas empleados en los distintos análisis, incluían un punto de colisión cuyo impacto no era del 100%, debido a su elección aleatoria.

A continuación, se estudia el comportamiento cuando el impacto es total, en escenarios realistas y para todas las combinaciones posibles en un radio de estudio. Para la obtención de los datos se ha generado un script 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.

El script genera todas las combinaciones posibles cada 10º (35 escenarios) en una circunferencia (zona de estudio), tal y como se muestra en Figura 49. De esta forma se garantiza que cada simulación incluye un impacto total.



1. Grados en una circunferencia

El escenario realista cuenta con las siguientes características:

* Velocidad máxima de 36.11 m/s (130 km/h)
* Radio de estudio de 100 m (zona de estudio 200 m)
* Radio de UAV de 0.25 m

El porcentaje de colisiones evitadas tras ejecutar el script ha sido de un 88.6 %. Se puede observar como la mayoría de los conflictos se producen cuando las rutas de los UAV se encuentran entorno a 280º y 350º, el algoritmo funciona peor.

1. Conflictos evitados según los grados de las rutas de los UAV y vel\_max

Se observa como al aumentar la velocidad, el porcentaje de colisiones evitadas aumenta, aunque no significativamente. En general se observa un mismo patrón, en el que la mayoría de las colisiones se producen entre los grados 280º y 350º.

Ahora analizamos el porcentaje de tiempo empleado por cada UAV y el porcentaje de colisiones evitadas, conforme se varía el radio de estos (cuando aumentamos el tamaño del UAV).

1. Aumento de tiempo variando el radio con vel\_max = 130 km/h

Como puede observarse en la Figura 51, cuando se aumenta el tamaño del radio, aumenta el tiempo empleado para realizar el recorrido. Es un resultado esperado, puesto que la curva tomada para evitar la colisión es mayor.

1. Media de conflictos evitados variando el radio con vel\_max = 130 km/h

El algoritmo responde mejor cuando el radio de los UAV es menor para velocidad máxima 130 km/h. A medida que este radio es aumentado, los conflictos evitados son menores. Para entornos realistas en los que los UAV no suelen superar los 150 km/h y sus masas no superan 1 m, se comportaría correctamente, evitando prácticamente el 100% de las colisiones.

Como se ha podido comprobar, dependiendo de la velocidad máxima y el radio establecido, el algoritmo responde mejor o peor. El porcentaje de colisiones evitadas según la velocidad máxima y el radio, se muestran en Figura 53, siendo la media de todos los escenarios, se observa como la configuración con radio a 0.125 y la velocidad máxima de 130 km/h es la que mejor responde. Hay que añadir que las colisiones que no han sido evitadas se dieron cuando las posiciones de los UAV se encontraban en ángulos pequeños, lo que indica que el algoritmo puede mejorarse para estas situaciones, pero en términos generales se comporta bien.

Además, se vio en apartados anteriores como ajustando los parámetros t\_slab y t\_nav\_step el conflicto quedaba solventado, si estos se ajustaban de forma correcta, según el escenario.

1. Media de conflictos evitados según vel\_max y radio
   1. Estudio con múltiples UAV

En este apartado se analiza el comportamiento del algoritmo BBCA, cuando el escenario dispone de varios UAV.

Para poder obtener el conjunto de datos, se han automatizado las simulaciones, para escenarios con 25, 50, 75 y 100 UAV. La velocidad máxima es de 130 m/s, el radio de los UAV de 5 m y el área de 5 km x 5 km.

|  |  |  |
| --- | --- | --- |
| Number of UAV | Collisions expected | Collisions using BBCA |
| 25 |  |  |
| 50 |  |  |
| 75 |  |  |
| 100 |  |  |

1. Conclusiones y propuestas

qw ejroñq erhj ñqeroi qeujqenoñ juherqibwekqriuweiquh woñqe oñ qjoe ruqwerhj oqweju opqwerho qeu iqwehr qweuqweio qweorqiwer oqwe rupqwehr owpqeu rwiqehr qw ejroñq

* 1. Conclusiones

juherqibwekqriuweiquh woñqe oñ qjoe ruqwerhj oqweju opqwerho qeu iqwehr qweu

* 1. Trabajo futuro

juherqibwekqriuweiquh woñqe oñ qjoe ruqwerhj oqweju opqwerho qeu iqwehr qweu

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 UAV 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 UAV 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 UAV,” *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 UAV 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.

Anexos

* + 1. Ejemplo de uso de la herramienta “analizador”

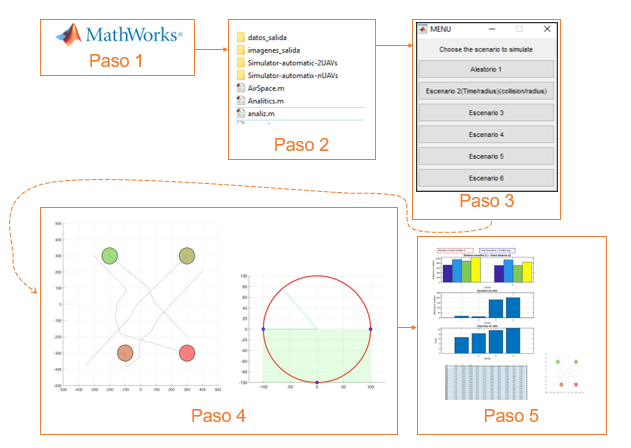
A continuación, se describe paso a paso el uso de la herramienta con un ejemplo simple. Se muestra en la Figura X cada uno de estos pasos.

* Paso 1: lo primero y fundamental es disponer de la herramienta MATLAB en su versión R2019-b.
* 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: esta nos permite visualizar los UAV, su trayectoria.
* Ventaja del algoritmo: esta nos permite visualizar su comportamiento. Se aconseja desactivar esta opción, en caso de simulaciones con escenarios densos.



1. Pasos para el uso de la herramienta “Analizador”
   * 1. Ejemplo de uso de la herramienta “simulador”

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

* Paso 1: lo primero y fundamental es disponer de la herramienta MATLAB en su versión R2019-b, en la cual se integra Simulink.
* 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 dos carpetas (datos y figuras).