



**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

Esto sería el resumen del documento

alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl

Agradecimientos

Aquí los agradecimientos.

alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl dhga alfkaj ldfjaldñf jaldfn adslfkj aldfj adfg jadf nalsg jakgn lagh ag lkadsjf nagla jf kajgn lagj kagnl

Índice de Contenido

[Declaración de Autoría iii](#_Toc41643630)

[Resumen v](#_Toc41643631)

[Agradecimientos vii](#_Toc41643632)

[CAPÍTULO 1. Introducción 1](#_Toc41643633)

[1.1. Motivación 1](#_Toc41643634)

[1.2. Objetivos 3](#_Toc41643635)

[1.3. Estructura de la memoria 4](#_Toc41643636)

[CAPÍTULO 2. Estado del arte 5](#_Toc41643637)

[2.1. Regulación UAVs 6](#_Toc41643638)

[2.2. Evitación de obstáculos 6](#_Toc41643639)

[2.3. Algoritmos para la deteción y evitación de colisiones entre múltiples UAVs 7](#_Toc41643640)

[2.3.1. Sense and avoid 7](#_Toc41643641)

[2.3.2. Artificial Potencial Field (APF) 8](#_Toc41643642)

[2.3.3. Particle Swarm Optimization (PSO) 9](#_Toc41643643)

[2.3.4. Collision Cone 10](#_Toc41643644)

[2.3.5. Dubins Paths 11](#_Toc41643645)

[2.3.6. Optimal Reciprocal Collision Avoidance (ORCA) 11](#_Toc41643646)

[2.3.7. Conclusión 13](#_Toc41643647)

[CAPÍTULO 3. Herramientas empleadas 14](#_Toc41643648)

[3.1. Matlab 14](#_Toc41643649)

[3.1.1. Simulink 15](#_Toc41643650)

[3.1.2. App Designer 16](#_Toc41643651)

[3.2. Github 16](#_Toc41643652)

[3.2.1. Git Bash 17](#_Toc41643653)

[3.3. Mendeley 17](#_Toc41643654)

[CAPÍTULO 4. Metodología y desarrollo 19](#_Toc41643655)

[CAPÍTULO 5. Simulador 20](#_Toc41643656)

[5.1. Aproximación al problema 20](#_Toc41643657)

[5.2. Entidades involucradas 21](#_Toc41643658)

[5.3. Subsistema UAV 22](#_Toc41643659)

[5.3.1. Dynamics 23](#_Toc41643660)

[5.3.2. Pilot 24](#_Toc41643661)

[5.3.3. Radar 27](#_Toc41643662)

[5.4. Subsistema ATC 27](#_Toc41643663)

[5.5. Interfaz de usuario 29](#_Toc41643664)

[CAPÍTULO 6. Analizador y algoritmo 30](#_Toc41643665)

[6.1. Mecanismo para detección de colisiones 30](#_Toc41643666)

[6.2. Mecanismo para evitación de colisiones (BBCA) 31](#_Toc41643667)

[6.2.1. Entradas 32](#_Toc41643668)

[6.2.2. Velocidad válida 32](#_Toc41643669)

[6.2.3. Obstáculo de velocidad 34](#_Toc41643670)

[6.2.4. Velocidad óptima 36](#_Toc41643671)

[6.3. Escenarios 39](#_Toc41643672)

[CAPÍTULO 7. Experimentos y resultados 43](#_Toc41643673)

[7.1. Análisis del algoritmo BBCA 43](#_Toc41643674)

[7.2. Estudio con dos UAVs 44](#_Toc41643675)

[7.2.1. Análisis del tiempo con variación del radio 44](#_Toc41643676)

[7.2.2. Análisis de colisiones con variación de radio 45](#_Toc41643677)

[7.2.3. Análisis de colisiones con variación de velocidad máxima 47](#_Toc41643678)

[7.2.4. Análisis de escenario realista 49](#_Toc41643679)

[7.2.5. Escenario realista con impacto total 51](#_Toc41643680)

[7.3. Estudio con múltiples UAVs 55](#_Toc41643681)

[CAPÍTULO 8. Conclusiones y propuestas 57](#_Toc41643682)

[8.1.1. Conclusiones 57](#_Toc41643683)

[8.1.2. Trabajo futuro 57](#_Toc41643684)

[Bibliografía 59](#_Toc41643685)

[Anexos 61](#_Toc41643686)

[A.1.1. Ejemplo de uso de la herramienta 61](#_Toc41643687)

[A.1.2. Manual de usuario 61](#_Toc41643688)

Índice de Figuras

[Figura 1. UAV 1](#_Toc41643729)

[Figura 2. UAV de mensajería en DHL 2](#_Toc41643730)

[Figura 3. Globo aerostático de 1849 5](#_Toc41643731)

[Figura 4. Sense and Avoid 8](#_Toc41643732)

[Figura 5. Esquema de APF 8](#_Toc41643733)

[Figura 6. Ejemplo PSO con dos mínimos diferentes [6] 9](#_Toc41643734)

[Figura 7. Ejemplo de cono de colisión 10](#_Toc41643735)

[Figura 8. Ejemplo de Dubins Path 11](#_Toc41643736)

[Figura 9. Dos agentes virtuales (izquierda), el obstáculo de velocidad truncado VO (centro) y semiplanos libres de colisiones (derecha). 13](#_Toc41643737)

[Figura 10. MATLAB logo 14](#_Toc41643738)

[Figura 11. Simulink & MATLAB logo 15](#_Toc41643739)

[Figura 12. GitHub logo 16](#_Toc41643740)

[Figura 13. Mendeley logo 17](#_Toc41643741)

[Figura 14. Elemento UAV en el simulador 22](#_Toc41643742)

[Figura 15. Elemento ATC en el simulador 22](#_Toc41643743)

[Figura 16. Subsistema UAV en Simulink 23](#_Toc41643744)

[Figura 17. Dynamics en subsistema UAV 24](#_Toc41643745)

[Figura 18. Máquina de estado de UAV 25](#_Toc41643746)

[Figura 19. Bloque comunicación de UAV 26](#_Toc41643747)

[Figura 20. Esquema tratamiento de mensajes en UAV 27](#_Toc41643748)

[Figura 21. Radar en subsistema UAV 27](#_Toc41643749)

[Figura 22. Bloque control de ATC 28](#_Toc41643750)

[Figura 23. Esquema tratamiento de mensajes ATC 28](#_Toc41643751)

[Figura 24. Interfaz de usuario del simulador 29](#_Toc41643752)

[Figura 25. Mecanismo para detectar colisiones 31](#_Toc41643753)

[Figura 26. Pseudocódigo del algoritmo 32](#_Toc41643754)

[Figura 27. Velocidad válida inicial en simulador 33](#_Toc41643755)

[Figura 28. Velocidad válida truncada 34](#_Toc41643756)

[Figura 29. Obstáculo velocidad en simulador (agente 1 con 3) 35](#_Toc41643757)

[Figura 30. Semiplano de obstáculo velocidad (agente 1 con 3) 35](#_Toc41643758)

[Figura 31. Ampliación de obstáculo según vector velocidad (agente 1 con 3) 36](#_Toc41643759)

[Figura 32. Cálculo de velocidad óptima 36](#_Toc41643760)

[Figura 33. Ocho puntos de corte posibles 37](#_Toc41643761)

[Figura 34. Cuatro puntos en esquinas 37](#_Toc41643762)

[Figura 35. Elección de velocidad óptima 38](#_Toc41643763)

[Figura 36. Elección del punto con ángulos iguales 38](#_Toc41643764)

[Figura 37. Simulación del caso ‘ángulos iguales’ 39](#_Toc41643765)

[Figura 38. Menú de escenarios para simular 40](#_Toc41643766)

[Figura 39. Entradas para simulación aleatoria 40](#_Toc41643767)

[Figura 40. Porcentaje de tiempo con variación de radio 45](#_Toc41643768)

[Figura 41. Porcentaje de colisiones evitadas con variación de radio 46](#_Toc41643769)

[Figura 42. Porcentaje de colisiones evitadas según la velocidad máxima 48](#_Toc41643770)

[Figura 43. Simulación de escenario 1 y 2 (izquierda) y escenario 3 y 4 (derecha) 49](#_Toc41643771)

[Figura 44. Colisión entre 2 UAVs sin empleo de BBCA 50](#_Toc41643772)

[Figura 45. Simulación de escenario con 2 UAVs empleando BBCA(izquierda) y la desviación realizada en la simulación (derecha) 50](#_Toc41643773)

[Figura 46. Distancia recorrida con BBCA vs distancia directa sin BBCA 51](#_Toc41643774)

[Figura 47. Porcentaje de desviación de la ruta inicial 51](#_Toc41643775)

[Figura 48. Incremento del tiempo en porcentaje por el uso de BBCA vs sin BBCA 51](#_Toc41643776)

[Figura 49. Grados en una circunferencia 52](#_Toc41643777)

[Figura 50. Conflictos evitados según los grados de las rutas de los UAVs y vel\_max 53](#_Toc41643778)

[Figura 51. Aumento de tiempo variando el radio con vel\_max = 130 km/h 53](#_Toc41643779)

[Figura 52. Media de conflictos evitados variando el radio con vel\_max = 130 km/h 54](#_Toc41643780)

[Figura 53. Media de conflictos evitados según vel\_max y radio 55](#_Toc41643781)

Índice de Tablas

[Tabla 1. Pseudocódigo para generar un escenario aleatorio 41](#_Toc41643782)

[Tabla 2. Porcentaje de tiempo con variación de radio 45](#_Toc41643783)

[Tabla 3. Colisiones con variación de radio 46](#_Toc41643784)

[Tabla 4. Colisiones con variación de velocidad máxima 48](#_Toc41643785)

[Tabla 5. Un ejemplo de tabla 61](#_Toc41643786)

2. Introducción

A lo largo de la siguiente memoria se refleja el trabajo realizado en el proyecto *“Gestión dinámica de colisiones en entornos con múltiples drones”*. En este primer capítulo se exponen los principales ítems que permitan introducir el proyecto de una forma clara y concisa.

* 1. Motivación

La forma con la que nos afrontamos 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.

La cosa es que 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 conocido como drones, son aeronaves no tripuladas que sobrevuelan el espacio aéreo. Por otro lado, tendremos como protagonista al software que permitirá la gestión de la que veníamos hablando.



1. UAV

La motivación del proyecto 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 los drones 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 como la navegación, dinámicas, sensores, comunicaciones, etc. hacen posible su empleo para distintas aplicaciones y por lo tanto se abre un gran abanico en los cuales se prevé se encontrarán los drones, cubriendo un sinfín de necesidades actuales y futuras.

En este Trabajo de Fin de Grado (de ahora 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 en tiempo real, de forma autónoma, para unos fines concretos, y con ausencia de colisiones.

* 1. Objetivos

En este proyecto nos centramos en la necesidad existente en la evitación de colisiones entre múltiples drones que se encuentran en el espacio aéreo, que permitan el desplazamiento de estos de forma exitosa, sin llegar a colisionar. Para llegar a obtener una solución al problema o necesidad planteado, será necesario abordar lo siguiente:

El desarrollo e implementación de la solución no sería posible sin antes realizar un análisis y estudio de los diferentes algoritmos para la evitación de colisiones en entornos con múltiples UAVs. Por ello, se ha realizado previamente y reflejado en el CAPÍTULO 2, un estudio sobre los diferentes algoritmos o soluciones existentes hasta el momento (**estado del arte**).

Por otro lado, es necesario un estudio que permita ver las distintas herramientas sobre la que se desarrollará la solución, y así justificar su elección. Las **herramientas empleadas** en este proyecto podrán encontrarse descritas en el CAPÍTULO 3 de esta memoria.

Una vez en contexto y establecidas las herramientas de desarrollo, y otras herramientas necesarias sobre las que apoyarse, pasamos al objetivo principal de este proyecto.

Debido al alto coste, el tiempo y riesgo que supondría testear los servicios o aplicaciones que ofrecerían los UAVs, 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.

Por lo tanto, el objetivo principal del proyecto se basa en el **desarrollo de un simulador** que permita implementar nuestro algoritmo para evitación de colisiones, y de esta forma poder simular múltiples UAVs en vuelo de forma autónoma y en tiempo real, y así obtener un conjunto de datos que permitan sacar conclusiones.

Por último, se realizará un **análisis de los datos obtenidos**, permitiendo contrastar nuestra solución con las ya existentes e incluyo ofrecer una propuesta de calidad.

Con todo ello, nos hemos marcado como objetivo final la **publicación de un paper**, que refleje todo el trabajo realizado.

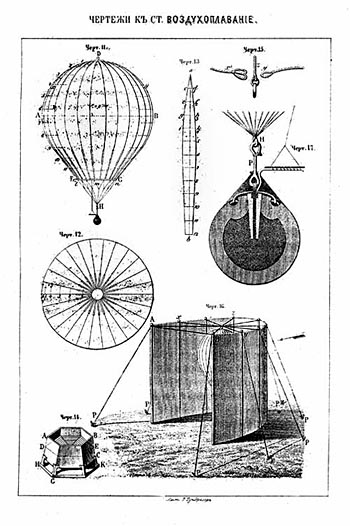
* 1. Estructura de la memoria

Este documento se estructura en diferentes capítulos con sus correspondientes desgloses. 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 UAVs 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 describe la metodología empleada en el ciclo de vida del proyecto.
* Capítulo 5: se corresponde con el simulador desarrollado, describiendo cada parte que lo compone, así como su comportamiento.
* Capítulo 6: se aborda el analizador empleado para el testeo del algoritmo, así como el funcionamiento del propio mecanismo para evitación de colisiones.
* Capítulo 7: se trata del análisis y estudios de los resultados obtenidos en las simulaciones del analizador.
* Capítulo 8: 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, de 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 usos 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 uso de estos vehículos para diversas aplicaciones 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 compone, sino que surge un nuevo problema que quizá es de mayor interés, la detección y evitación de colisión entre estos vehículos.

Por lo tanto, nace un requisito indispensable para que estos vehículos pueden cubrir las nuevas aplicaciones a las que se enfrentan, la capacidad de evitar colisionarse.

A continuación, se detallarán aquellos aspectos que se han tenido en cuenta como partida para el desarrollo de este proyecto.

* 1. Regulación UAVs

Aunque el auge del uso de UAVs en aplicaciones civiles es relativamente nuevo, ya se han elaborado pautas o requisitos que regulan el uso de estos en el espacio aéreo.

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 [1].

* 1. Evitación de obstáculos

La evitación de obstáculos ha sido un problema muy estudiado por la comunidad científica, destacando su aplicación a robots móviles terrestres. Por lo tanto, se han desarrollado diferentes algoritmos que ofrecen una solución, más o menos acertada, al problema planteado.

Tras un estudio de los diferentes algoritmos propuestos, se concluye que todos se apoyan en los diferentes sensores de los que dispone el hardware con el que se trabaja, además de aplicarse 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.

El algoritmo empleado para la detección y evitación de colisiones entre múltiples UAVs será aquel que se destine a un escenario donde se empleen las tres dimensiones. A continuación, se describen algunos de los algoritmos encontrados para este fin, así como la propuesta para este proyecto, siendo el punto de partida de este.

* 1. Algoritmos para la deteción y evitación de colisiones entre múltiples UAVs

El proyecto se centra en el algoritmo que permita detectar y evitar la colisión entre múltiples drones, donde existen una serie de características que hacen que este problema sea complejo:

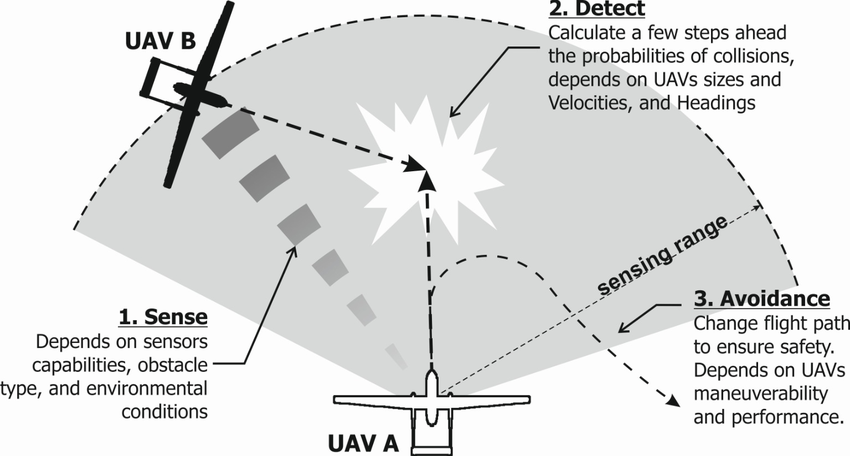
* Los obstáculos se encuentran en movimiento.
* Alta velocidad de movimiento.
* Entidades con toma de decisiones.
* Información en tiempo real, donde pueden producirse retrasos.

Existen multitud de algoritmos ya diseñados para este fin, por lo tanto, se ha creado una taxonomía de aquellos que resulten de mayor interés ya que existen infinidad de propuestas tales como: métodos de campos potenciales, algoritmos evolutivos, basados en grafos, búsqueda en el espacio de estados, relaciones geométricas, basados en fuerzas o velocidades, entre otros.

Sense and avoid

El término “Sense and Avoid” ha sido acuñado por la comunidad aeronáutica para referirse a los sistemas SAAS (del inglés, Software As A Service) donde existen don funciones fundamentales: una función sensorial que permita adquirir información del entorno y una función de evitación que evalúa el riesgo de una posible colisión, tomando medidas ya sea mediante una llamada al piloto o tomando acciones de forma autónoma.

La automatización de evitar colisiones ha sido muy estudiada a lo largo de los años y podemos encontrar en la literatura más de 60 métodos diferentes [2]. Hay que añadir que no todos son interesantes cuando hablamos de aplicaciones en tiempo real, por el aumento computacional del mismo.

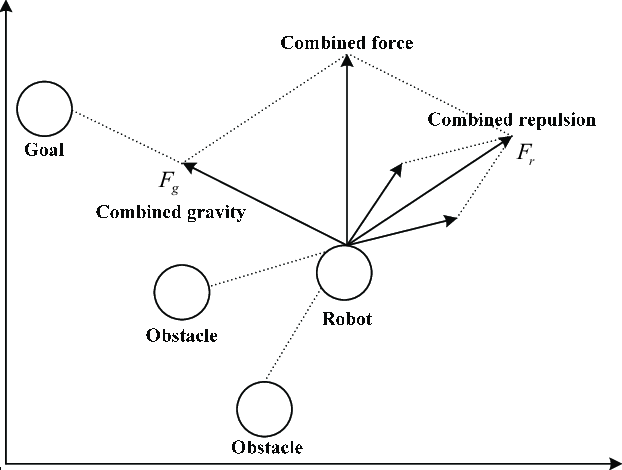


1. Sense and Avoid

Artificial Potencial Field (APF)

Existen métodos basados en la fuerza, donde se modelan grupos de agentes virtuales como 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.

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 [3], un algoritmo APF optimizado (de sus siglas en inglés, Artificial Potential Field).



1. Esquema de APF

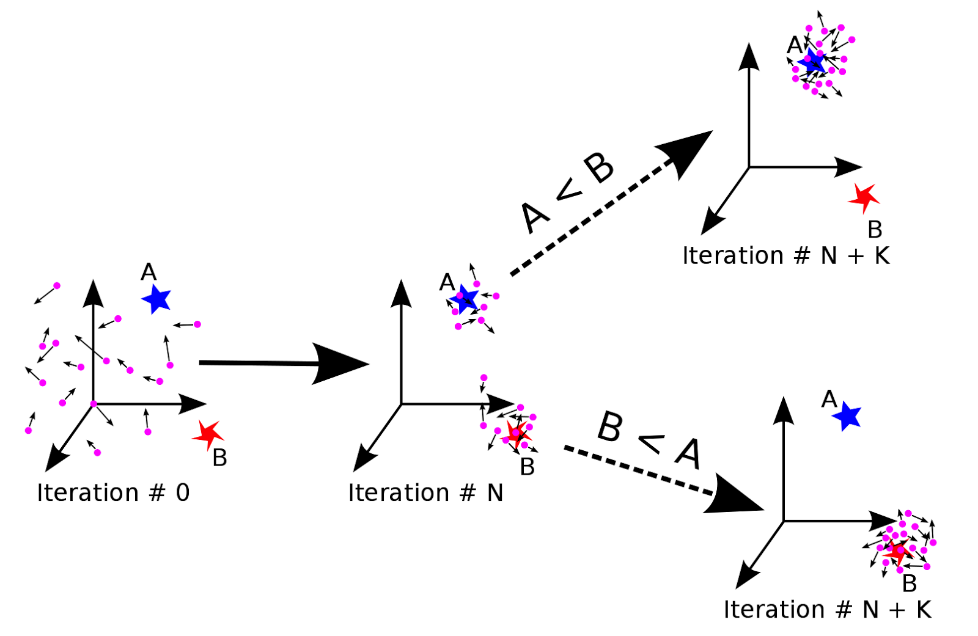
Estos métodos se basan en la idea que Khatib introdujo en 1986, donde el agente se mueve atraído en un campo de fuerzas hacia su objetivo final y el resto de los agentes se comportan como obstáculos con fuerzas repulsivas [4].

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.

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 [5].



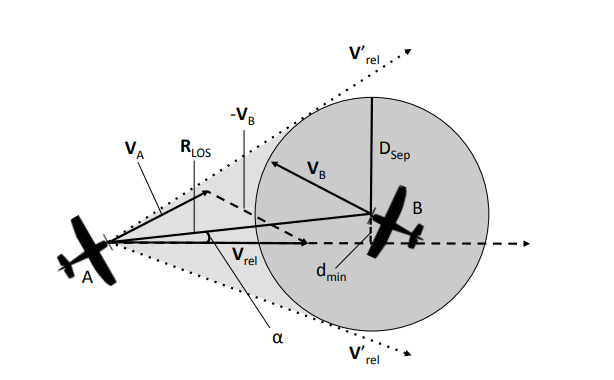
1. Ejemplo PSO con dos mínimos diferentes [6]

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.

Collision Cone

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 [7].

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

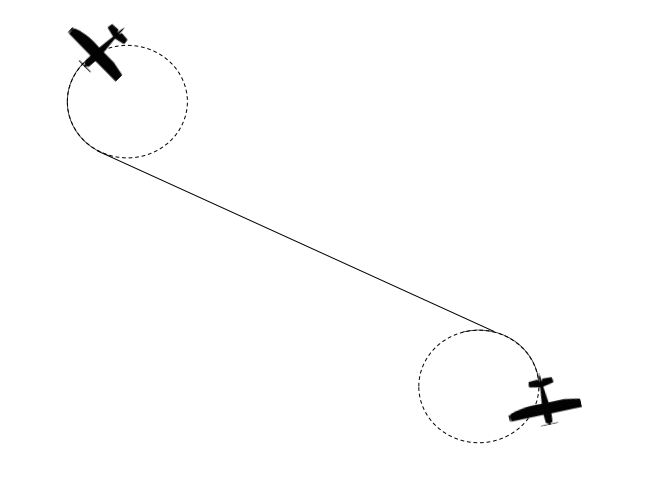
Cuando la  de A sea tangencial al área circular de B, marcada como , los agentes se evitarán con una deviación mínima a su objetivo. A los rangos de velocidades comprendidas entre los dos vectores se le conoce como cono de colisión.

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 [8] se ofrece un estudio en profundidad sobre este método.

Dubins Paths

Este método fue propuesto por Lester Dubins en 1957, en el que demostró que el camino más corto se puede formar uniendo líneas tangenciales con arcos circulares.

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

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 [9].

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 velocidad 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”, que incluye aquellas velocidades prohibidas para un agente respecto de otro. Una vez establecido el obstáculo de velocidad, se generan semiplanos 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).

ORCA garantiza la navegación libre de colisiones cuando los escenarios no son densos [10], 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 [11].

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.

Conclusión

Tras el estudio y análisis de los diferentes métodos o algoritmos propuestos en la literatura, se ha determinado que uno de 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 UAVs involucrados.

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 para este proyecto, 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 sobre la que se ha trabajado 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. [12].

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.

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 [13]. 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 proyecto. 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.

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. Además permite trabajar de forma conjunta con el Git que ofrece MATLAB.

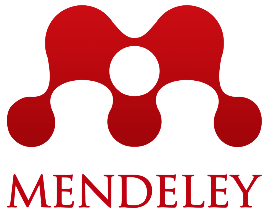
Por lo tanto, este proyecto será mantenido a través de GitHub en el repositorio almacenado en <https://github.com/PalomaSanx/UAVsimulation_TFG>.

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 [14].

* 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 y desarrollo

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

1. Simulador

A lo largo de este capítulo se describe el objetivo principal de este proyecto, el desarrollo de un simulador que permita testear los servicios o aplicaciones que ofrecerían los UAVs. 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 el que múltiples UAVs 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 como extensión a un algoritmo ya existente, 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 como se comentó en el Herramientas empleadasCAPÍTULO 3. Se ha empleado 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 proceso para la construcción del simulador, donde se encuentra un primer acercamiento al problema que permite tener las pautas principales, los objetos/entidades de los que se compone y su funcionamiento. Además, se describirá el desarrollo de un segundo simulador, que ha permitido testear el funcionamiento del algoritmo propuesto para la labor de detección y evitación de colisiones.

* 1. Aproximación al problema

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

Un conflicto o colisión, es la pérdida de separación mínima entre dos UAVs, 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:

* Información de estado actual: 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. Se puede emplear información del estado actual: línea recta del vector de velocidad actual.

Mecanismo para evitación de colisión, es 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 necesaria para la detección del conflicto para determinar la modificación de la velocidad de cada entidad, así como la actualización de los parámetros involucrados, como por ejemplo las posiciones.

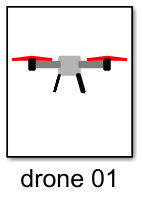
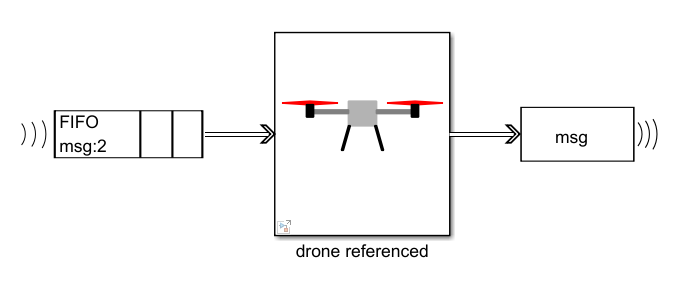
Por último, se han establecido cada uno de los estados por los que puede pasar un UAV:

* Inicio
* Libre
* Despegue
* Vuelo
* Aterrizaje
  1. Entidades involucradas

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).

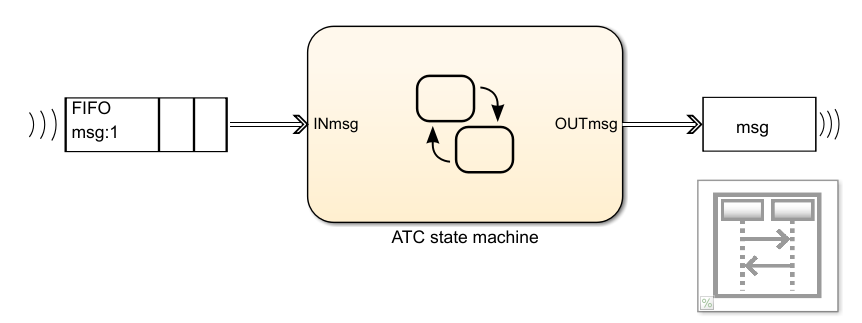
Ambos elementos emiten y reciben mensajes de forma continua. La recepción de los mensajes se realiza con el bloque ‘*Queue’* que facilita Simulink y mediante el método FIFO (por sus siglas en inglés, First-In First-Out) se antienden y procesan cada mensaje almacenado.

Los UAVs 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

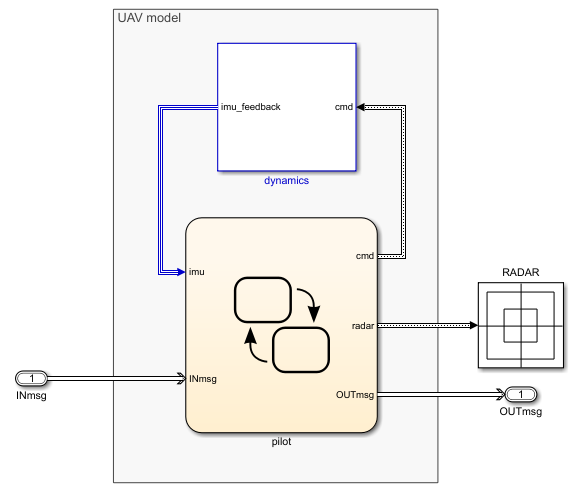
A continuación, se detalla tanto la composición como el funcionamiento de ambos subsistemas para el desarrollo del simulador. Estos subsistemas han sido implementados mediante el entorno gráfico de Simulink que permite construir el modelo a simular mediante bloques.

* 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 16 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.



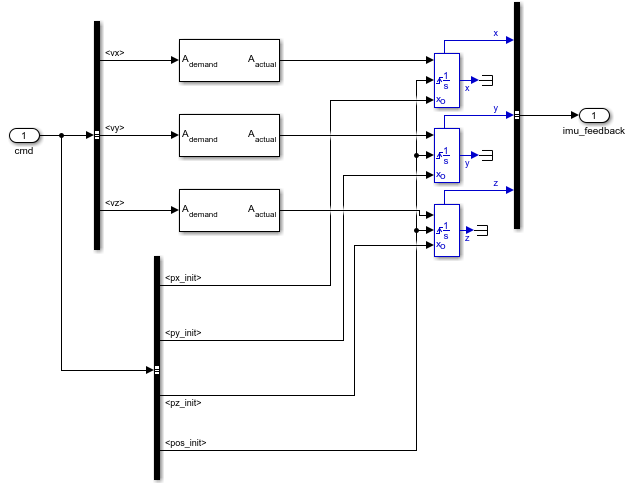
1. Subsistema UAV en Simulink

Dynamics

Como se puede observar en la Figura 17 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 UAVs. 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 16 ambos puertos (*cmd* e *imu\_feedback*) son utilizados en ‘*pilot’* que es el corazón del subsistema UAV, contiene la lógica del mismo.

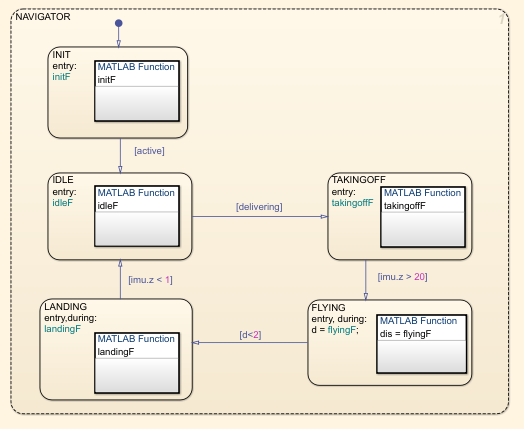


1. Dynamics en subsistema UAV

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 18) y el otro la comunicación (ver Figura 19) de cada UAV.

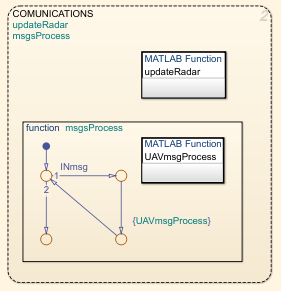


1. Máquina de estado de UAV

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:

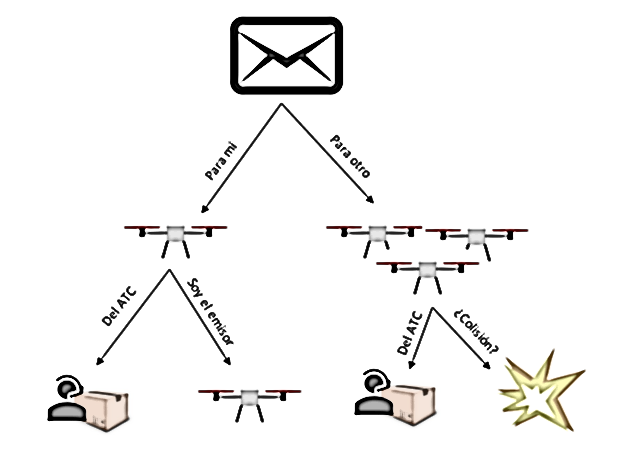
* Init: es el estado inicial y solo se ejecutará la primera vez en cada simulación. A través de este estado se incializará la estructura de estructura de datos local a cada UAV, las posiciones iniciales, el número de UAVs involucrados en la simulación y la variable que permite la transición entre estados.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

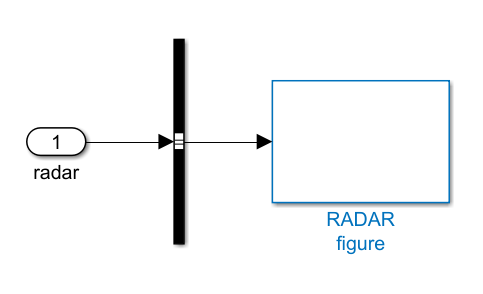
Gracias a dicho tratamiento es posible la actualización del escenario de simulación, así como la puesta en marcha hacia un nuevo destino por parte del UAV o la detección de una colisión.



1. Esquema tratamiento de mensajes en UAV

Radar

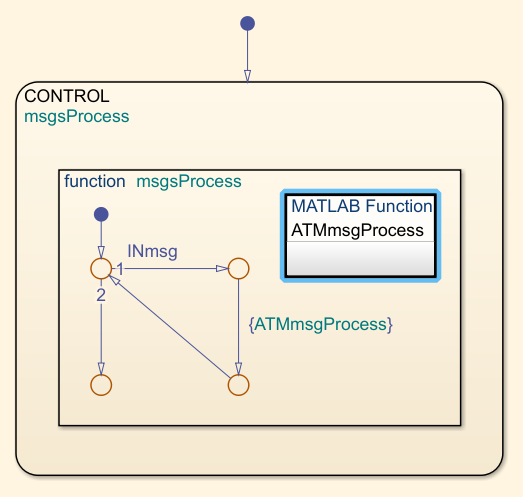
Como puede observarse en la Figura 21, el gráfico 3D que permitirá ver el transcurso de la simulación, es alimentado a través del puerto ‘*radar*’. A través de este puerto se actualiza la información necesaria tal como la posición o el id de cada UAV.



1. Radar en subsistema UAV
   1. Subsistema ATC

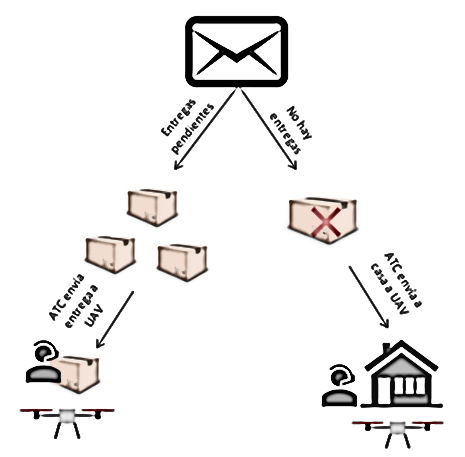
Al igual que ocurre en el subsistema UAV, el ATC también recibe y emite mensajes, principalmente para poder ejecutar órdenes de entrega a cada uno de los UAV.

El subsistema ATC se compone de un único bloque de control (ver Figura 22) que se encarga del procesamiento de dichos mensajes, además inicializa la estructura de datos que maneja el ATC para el control sobre las entregas.



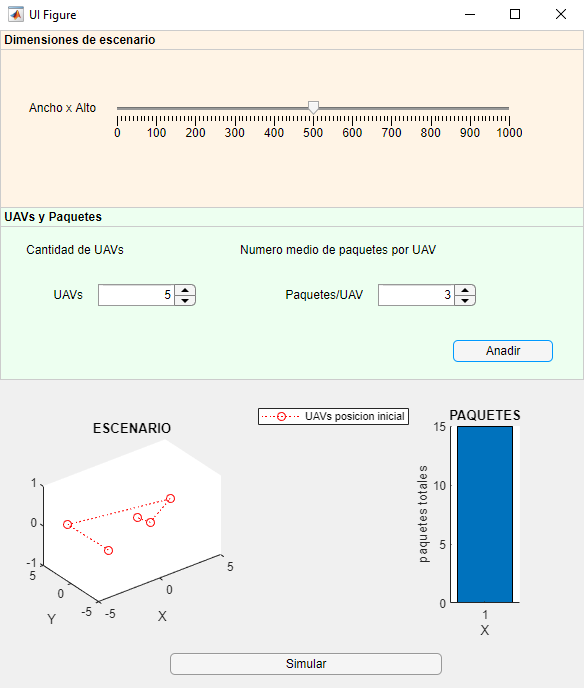
1. Bloque control de ATC

El procesamiento de los mensajes por parte del ATC puede observarse en la Figura 23. 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 24).



1. Interfaz de usuario del simulador
2. Analizador y algoritmo

Debido a la complejidad del sistema desarrollado, se ha optado por la implementación de un segundo simulador/analizador que permita testear la evolución del algoritmo para la evitación de colisiones entre múltiples UAVs. De esta forma, se han eliminado aquellas tareas que son irrelevantes para el testeo de este, logrando un ahorro en tiempo tanto en el desarrollo del algoritmo como en la ejecución de las distintas simulaciones. Además, permite su integración en el simulador de una forma independiente, obteniendo así flexibilidad y modularidad en el sistema.

El desarrollo de este segundo simulador se ha llevado a cabo mediante la herramienta MATLAB, permitirá la visualización de las trayectorias llevadas a cabo por cada uno de los UAVs, los distintos aspectos contemplados en el mismo (regiones permitidas, regiones no permitidas, velocidad objetivo, velocidad elegida, etc.) así como la comunicación de mensajes en caso de colisión u otro aspecto relevante.

A partir de este simulador se ha obtenido un conjunto de datos, que permiten analizar la eficiencia de este bajo diferentes situaciones.

* 1. Mecanismo para detección de colisiones

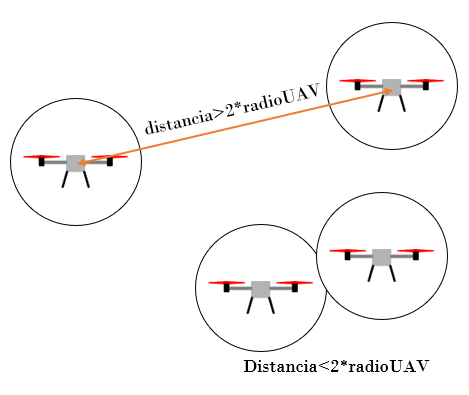
Como ya se había comentado con anterioridad, se entiende por colisión a la situación en la que dos o más agentes superan un umbral de seguridad preestablecido. Los UAVs son modelados como circunferencias para una mayor simplicidad en las ejecuciones.

En nuestro caso, se establece que dicha colisión se producida cuando la distancia existente entre los UAVs involucrados en la simulación supere al doble del radio de dichos UAVs.

Siendo *i* el identificador del UAV que lanza la ejecución de la función y *j* todos los agentes involucrados. Primero se establece la distancia entre *i* y *j*, mediante sus posiciones en el espacio, *=(x1,y1)* y *=(x2,y2)*.

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

En la Figura 25 puede observarse de forma clara cuando se produce dicha colisión.

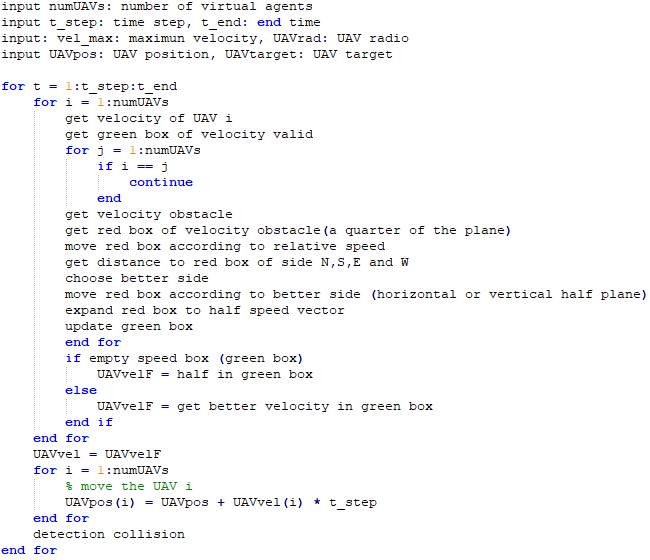


1. Mecanismo para detectar colisiones
   1. Mecanismo para evitación de colisiones (BBCA)

El mecanismo que permite evitar las colisiones entre los distintos UAVs 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. El nombre establecido a este mecanismo es “*Bounding Box Collision Avoidance”* (de ahora en adelante, BBCA, por sus siglas).

Este algoritmo es la propuesta de un mecanismo de menor complejidad en cuanto a coste computacional e implementación se refiere respecto a otros algoritmos ya implementados, como el famoso 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 proyecto, como mecanismo para la evitación de colisiones.



1. Pseudocódigo del algoritmo

Entradas

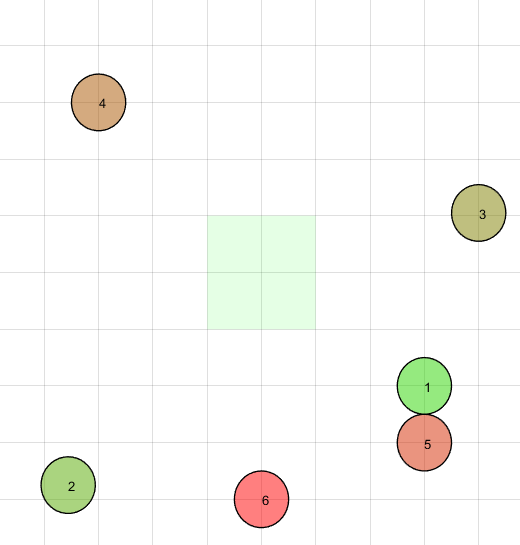
Existen un conjunto de inputs o entradas, empleadas por el algoritmo, y que permiten la simulación de diferentes escenarios. Estas entradas se dividen principalmente en:

* Número de agentes o UAVs
* Posición de inicio de cada agente o UAV
* Velocidad máxima
* Posición de destino de cada agente o UAV
* Radio de UAVs

Velocidad válida

Como se puede observar en la Figura 26 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.

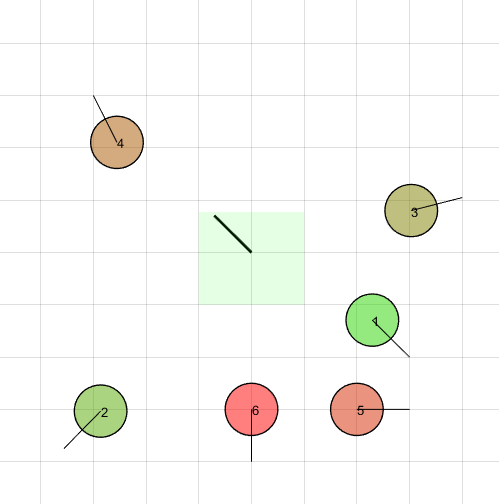
Inicialmente se configura por medio de la velocidad máxima y se actualiza en función de las restricciones introducidas.



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

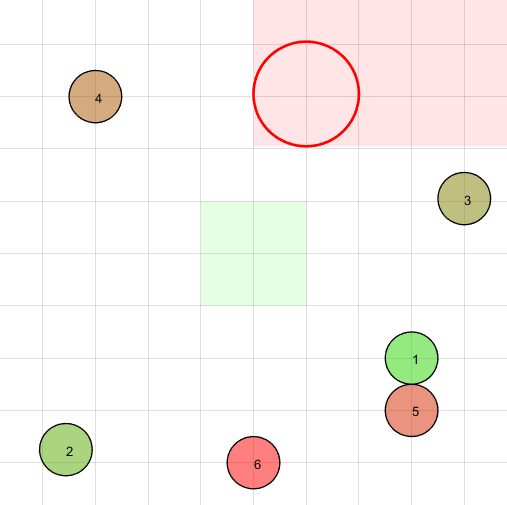
Una vez actualizado el conjunto de velocidades válidas, se comprueban dos posibles casos:

* 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 mejor velocidad entre las posibles.

Obstáculo de velocidad

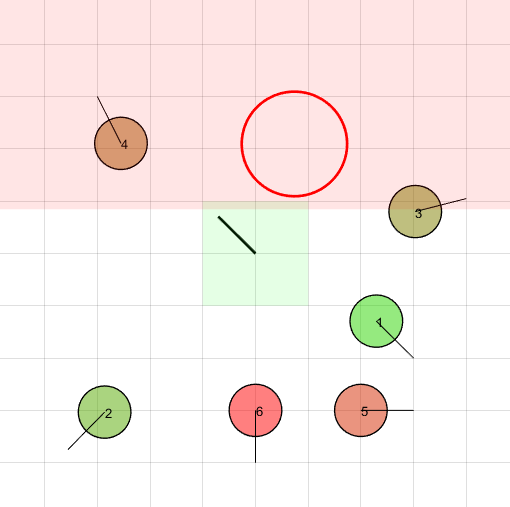
De la misma forma en la que se ha modelado la velocidad permitida o velocidad válida, se ha implementado el obstáculo de velocidad, pero en este caso mediante una caja roja con lados Norte (N), Sur (S), Este (E) y Oeste (W).

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.



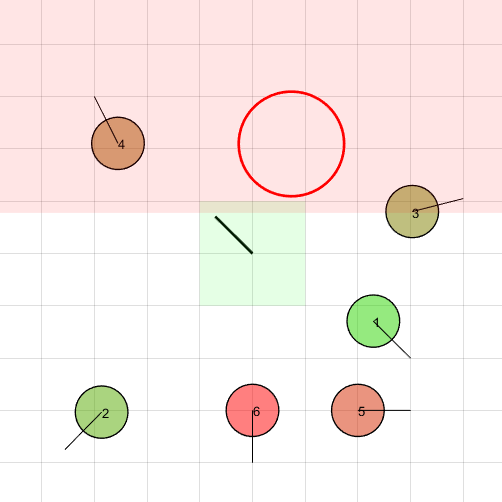
1. Obstáculo velocidad en simulador (agente 1 con 3)

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.

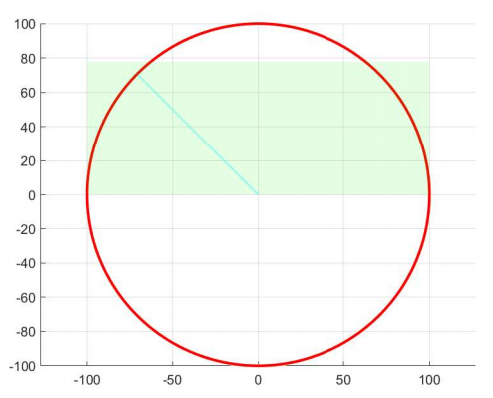


1. Ampliación de obstáculo según vector velocidad (agente 1 con 3)

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 32.

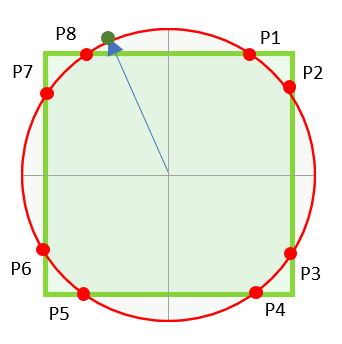


1. Cálculo de velocidad óptima

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

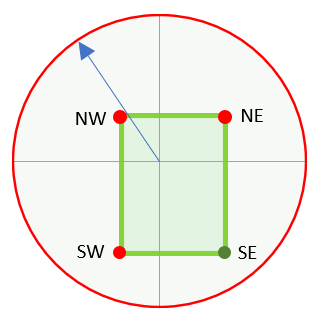
* La velocidad directa al objetivo se encuentra dentro de las velocidades válidas, por lo tanto, se establece como mejor opción dicha velocidad.
* La velocidad directa al objetivo no está dentro de las velocidades válidas: se trata mediante un método que hemos denominado “método de los doce puntos”.

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.



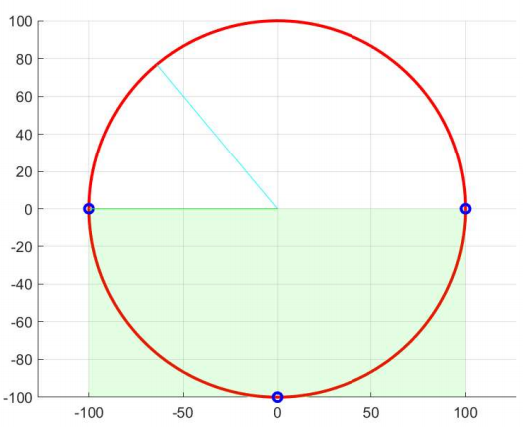
1. Ocho puntos de corte posibles

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 y permita avanzar en mayor medida hacia el objetivo.



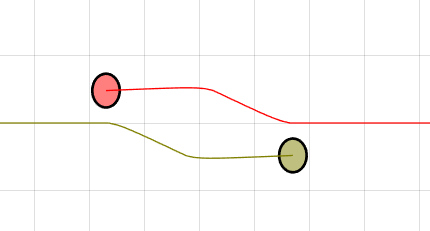
1. Elección de velocidad óptima

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 36, 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 UAVs se encuentran en el camino, se consiguen evitar, tal y como se muestra en Figura 37.

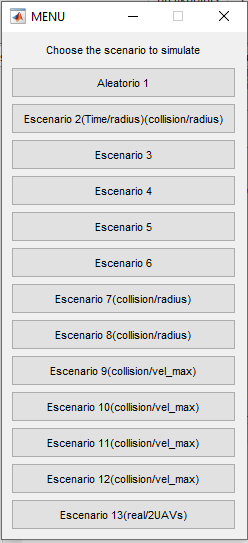


1. Simulación del caso ‘ángulos iguales’
   1. Escenarios

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

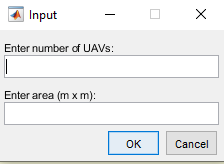
* Posiciones iniciales de UAVs
* Posiciones objetivo de UAVs
* Radio de UAV
* Velocidad máxima
* Número de UAVs

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 UAVs, 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 UAVs.

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

1. Pseudocódigo para generar un escenario aleatorio
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 6. 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”* 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 UAVs.
* Estudio de escenarios con múltiples UAVs.

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 UAVs

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

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 UAVs, 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 500 m x 500 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 UAVs 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 UAVs, el tiempo necesario para llegar a su destino es creciente en su mayoría. Esto ocurre porque en el escenario simulado los UAVs 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.

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 UAVs 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 UAVs, en que se varían sus radios, para poder determinar la influencia de este en las colisiones. Como el escenario se compone de 2 UAVs, 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 500 m x 500 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 UAVs 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 UAVs, el número de UAVs 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.

Análisis de colisiones con variación de velocidad máxima

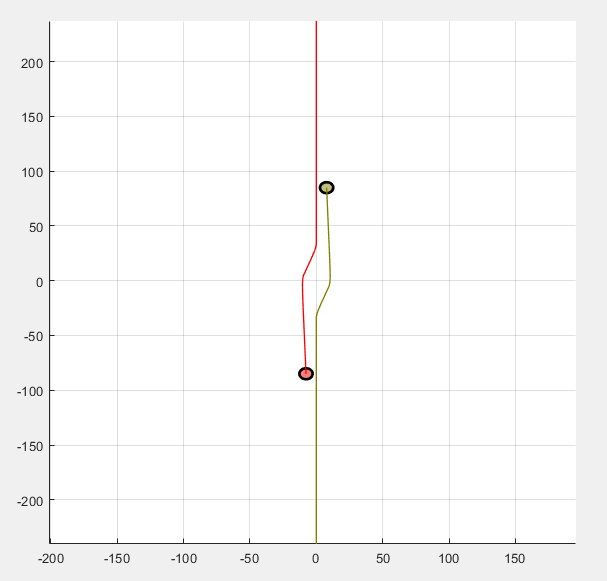
A continuación, se analizan los resultados obtenidos en simulaciones donde la velocidad máxima en cada escenario conflictivo se ha variado, para determinar el porcentaje de colisiones evitadas cuando este valor cambia.

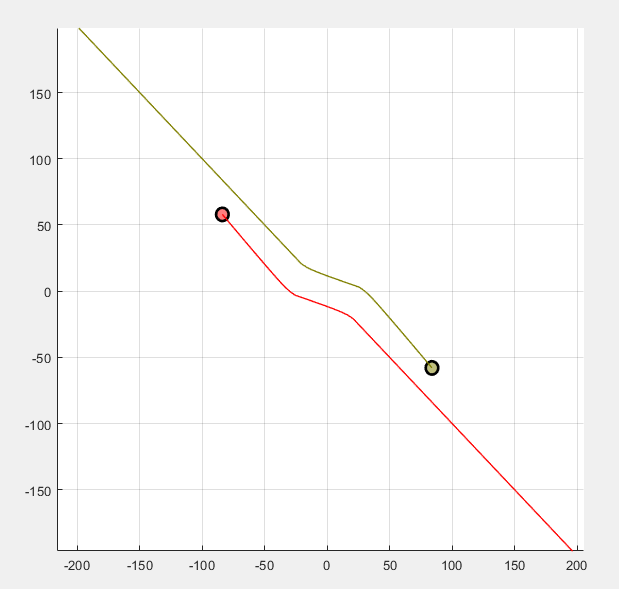
Los escenarios empleados representan el cruce entre 2 UAVs que deben evitarse para no colisionar, se ha empleado un radio de 5 m y área 500 m x 500 m.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 UAVs 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%.

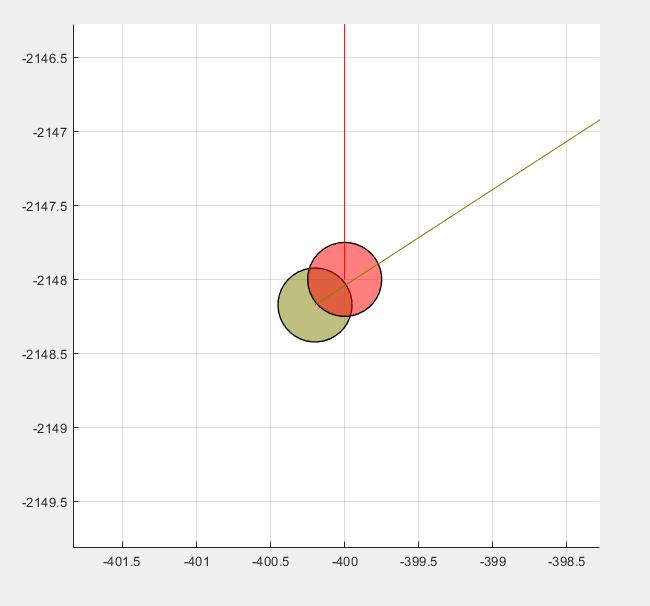
Análisis de escenario realista

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 (UAVs) deben ofrecer un servicio en un municipio/ciudad desde un punto inicial hasta su objetivo/destino.

Las características del escenario son:

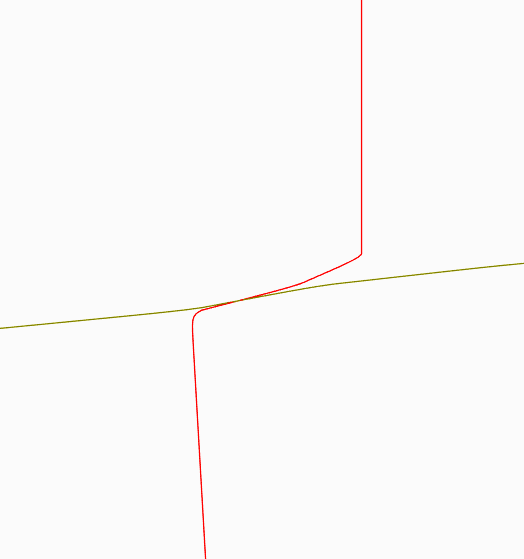
* Área de 5 Km x 5 Km.
* 2 UAVs (numUAVs).
* UAV con radio 0.25 m (UAVrad).
* Velocidad máxima 100 km/h, equivalente a 27.8 m/s (vel\_max).
* Las rutas suponen el cruce de los UAVs.
* Cada paso de simulación en 0.05 s (t\_sim\_step).
* Intervalo para recalcular la navegación (t\_nav\_step) en 0.5.
* 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.

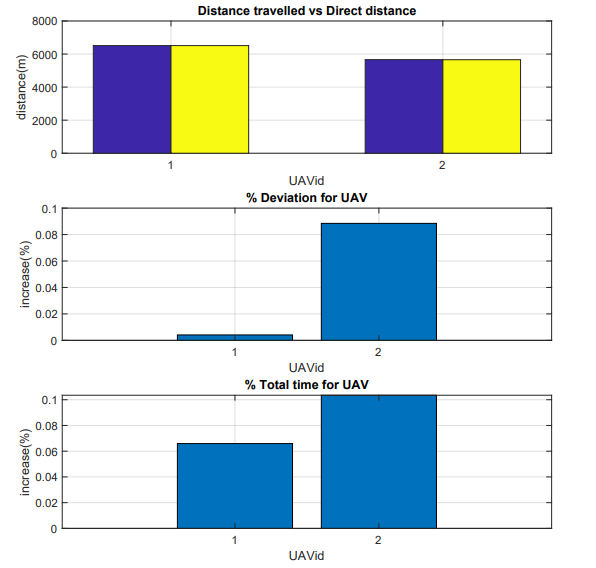


1. Colisión entre 2 UAVs sin empleo de BBCA

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

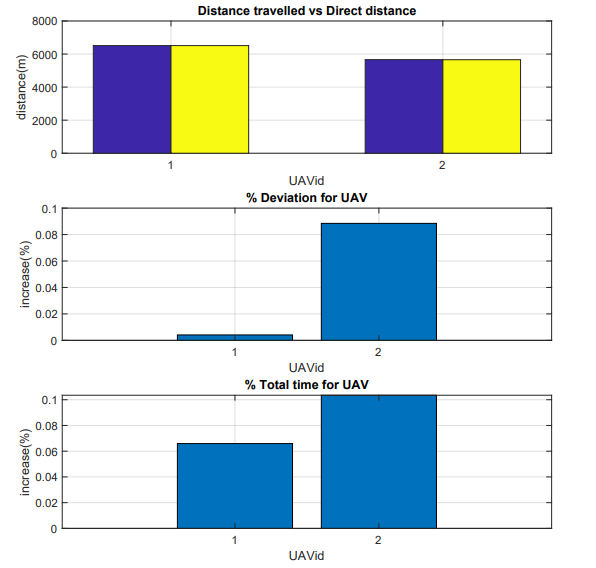


1. Simulación de escenario con 2 UAVs empleando BBCA(izquierda) y la desviación realizada en la simulación (derecha)



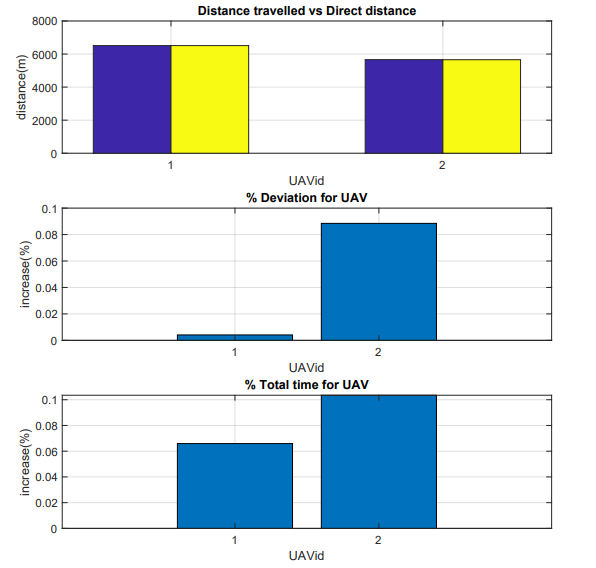
1. Distancia recorrida con BBCA vs distancia directa sin BBCA

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.



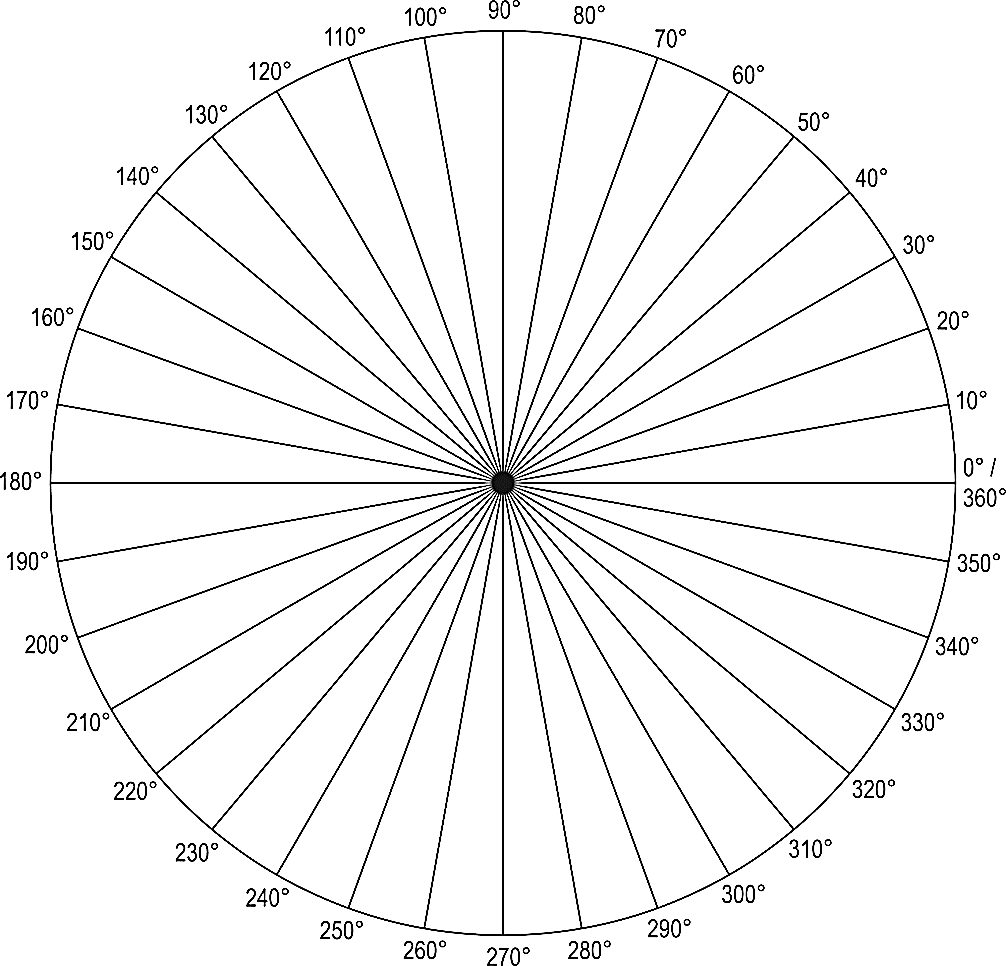
1. Incremento del tiempo en porcentaje por el uso de BBCA vs sin BBCA

Escenario realista con impacto total

Cuando hablamos de impacto total, nos referimos a que el punto de colisión es alcanzado por los UAVs 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 UAVs 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 UAVs 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 UAVs se encuentran entorno a 280º y 350º, el algoritmo funciona peor.

1. Conflictos evitados según los grados de las rutas de los UAVs 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 UAVs 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 UAVs 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 UAVs 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 UAVs

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

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] “‘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/.

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

[3] 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.

[4] H. Zhang, P. Wang, Y. Zhang, B. Li, and Y. Zhao, “A Novel Dynamic Path Re-Planning Algorithm with Heading Constraints for Human following Robots,” *IEEE Access*, vol. 8, pp. 49329–49337, 2020, doi: 10.1109/ACCESS.2020.2979867.

[5] 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.

[6] “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.

[7] 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.

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

[9] 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.

[10] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, “Reciprocal n-body Collision Avoidance.”

[11] Adam T Lake, “Reciprocal Collision Avoidance and Navigation for Video Games,” 02-Mar-2012. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/reciprocal-collision-avoidance-and-navigation-for-video-games.html.

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

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

[14] “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

adflñkajf qelrkj qer lqewrj hqlkrj qlhr lqjr lkqrk ckzfjasdlfh qenrl jqelrkj qleh hnlqwerj qw

lajfa lnfladsjf asdfn ladfj aldfladjf ladjflñaeyrqehrn lqwerj oqewrh nqer

* + 1. Manual de usuario

lkadfjla jdflqjer qertkj qer'ijqtej qoier hnqert

kllñkalñkdfg laesrfj lqñwer hnladf fjowqehtn lkwrjeoyhk lñkjje wlkjwenr ñljwer k

1. Un ejemplo de tabla

|  |  |
| --- | --- |
| ewqr qwe | 90 |
| llkadsfn | 10 |