



**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 1 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

Resumen v

Agradecimientos vii

CAPÍTULO 1. Introducción 1

1.1.1. Motivación 1

1.1.2. Objetivos 3

1.1.3. Estructura de la memoria 4

CAPÍTULO 2. Estado del arte 5

2.1.1. Regulación UAVs 6

2.1.2. Evitación de obstáculos 6

2.1.3. Algoritmos para la deteción y evitación de colisiones entre múltiples UAVs 7

2.1.4. Sense and avoid 7

2.1.5. Optimal Reciprocal Collision Avoidance (ORCA) 8

CAPÍTULO 3. Herramientas empleadas 9

3.1.1. Matlab 9

3.1.2. Simulink 10

3.1.3. App Designer 11

3.1.4. Github 11

3.1.5. Git Bash 11

3.1.6. Mendeley 12

CAPÍTULO 4. Metodología y desarrollo 13

CAPÍTULO 5. Simulador 14

5.1. Aproximación al problema 14

5.2. Entidades involucradas 15

5.3. Subsistema UAV 16

5.3.1. Dynamics 17

5.3.2. Pilot 18

5.3.3. Radar 21

5.4. Subsistema ATC 21

5.5. Interfaz de usuario 23

5.6. Simulador para testear EL algoritmo 23

5.6.1. Mecanismo para detección de colisiones 24

5.6.2. Mecanismo para evitación de colisiones 25

5.6.2.1 Entradas 26

5.6.2.2 Velocidad válida 26

5.6.2.3 Obstáculo de velocidad 28

5.6.2.4 Velocidad óptima 30

CAPÍTULO 6. Experimentos y resultados 33

6.1.1. Una sección 33

6.1.2. Otra sección 34

CAPÍTULO 7. Conclusiones y propuestas 37

7.1.1. Conclusiones 37

7.1.2. Trabajo futuro 37

Bibliografía 39

Anexos 41

A.1.1. Ejemplo de uso de la herramienta 41

A.1.2. Manual de usuario 41

Índice de Figuras

[Figura 1. UAV 1](#_Toc39656928)

[Figura 2. UAV de mensajería en DHL 2](#_Toc39656929)

[Figura 3. Globo aerostático de 1849 5](#_Toc39656930)

[Figura 4. Sense and Avoid 8](#_Toc39656931)

[Figura 5. MATLAB logo 9](#_Toc39656932)

[Figura 6. Simulink & MATLAB logo 10](#_Toc39656933)

[Figura 7. GitHub logo 11](#_Toc39656934)

[Figura 8. Mendeley logo 12](#_Toc39656935)

[Figura 9. Elemento UAV en el simulador 16](#_Toc39656936)

[Figura 10. Elemento ATC en el simulador 16](#_Toc39656937)

[Figura 11. Subsistema UAV en Simulink 17](#_Toc39656938)

[Figura 12. Dynamics en subsistema UAV 18](#_Toc39656939)

[Figura 13. Máquina de estado de UAV 19](#_Toc39656940)

[Figura 14. Bloque comunicación de UAV 20](#_Toc39656941)

[Figura 15. Esquema tratamiento de mensajes en UAV 21](#_Toc39656942)

[Figura 16. Radar en subsistema UAV 21](#_Toc39656943)

[Figura 17. Bloque control de ATC 22](#_Toc39656944)

[Figura 18. Esquema tratamiento de mensajes ATC 22](#_Toc39656945)

[Figura 19. Interfaz de usuario del simulador 23](#_Toc39656946)

[Figura 20. Mecanismo para detectar colisiones 25](#_Toc39656947)

[Figura 21. Pseudocódigo del algoritmo 26](#_Toc39656948)

[Figura 22. Velocidad válida inicial en simulador 27](#_Toc39656949)

[Figura 23. Obstáculo velocidad en simulador (agente 1 con 3) 28](#_Toc39656950)

[Figura 24. Semiplano de obstáculo velocidad (agente 1 con 3) 29](#_Toc39656951)

[Figura 25. Ampliación de obstáculo según vector velocidad 29](#_Toc39656952)

Índice de Tablas

[Tabla 1. Un ejemplo de tabla 17](#_Toc480805014)

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 prevee 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 4 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 tiempor 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 la ya existentes e incluyo ofrecer una propuesta de calidad.

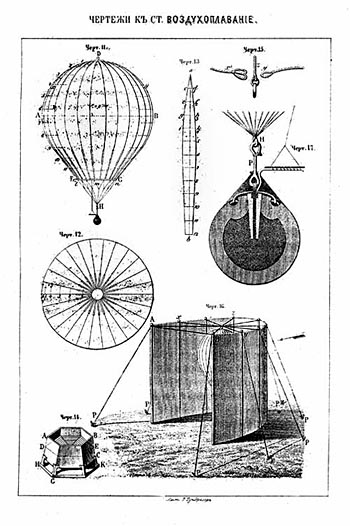
* + 1. Estructura de la memoria

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

* Capítulo 1:
* Capítulo 2:
* Capítulo 3:
* Capítulo 4:
* Capítulo 5:
* Capítulo 6:
* Capítulo 7:

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 tirpulados al cielo cargados de bombas, para misiones de vigilancia militar, para 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 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 metereológicas 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, permitendo la deteción 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 trabajar en 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:

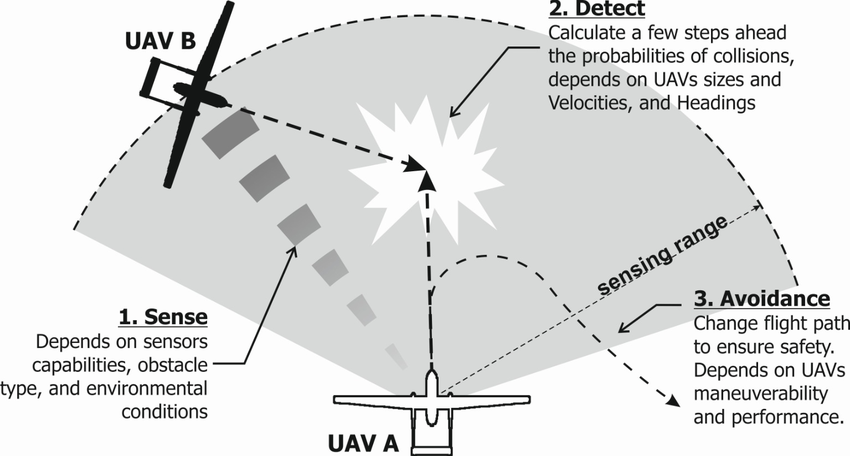
* Los obstáculos se encuentran en movimiento.
* Alta velocidad de movimiento.
* Entidades con toma de decisiones.
* Información en tiempo real, donde pueden producirse retrasos.

Existen multitud de algoritmos ya diseñados para este fin, por lo tanto, se ha creado una taxonomía de los mismos.

* + 1. 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 literura más de 60 métodos diferentes [3]. 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
   * 1. Optimal Reciprocal Collision Avoidance (ORCA)

Evitación óptima y recíproca de colisión (ORCA, de sus siglas en inglés, Optimal Reciprocal Collision Avoidance), se basa en elegir la velocidad óptima del campo de velocidad, de tal manera que el obstáculo sea evitado.

Este algoritmo parte de la hipótesis de que cada UAV es independiente y no se comunica con otro. Cada UAV se encuentra continuamnete en un ciclo de deteción 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.

ORCA garantiza la navegación libre de colisiones [2].

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 princiaplamente 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 [4].

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 librerias 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 [5]. 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 record.

* + 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.

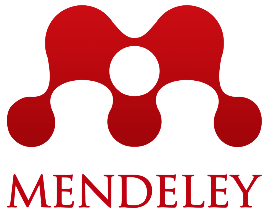
Por lo tanto, este proyecto será mantenido a través de GitHub en el repositorio almacenado en <https://github.com/PalomaSanx/UAVsimulation_TFG>.

* + 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 [6].

* + 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 CAPÍTULO 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 colisón, 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.

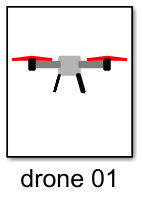
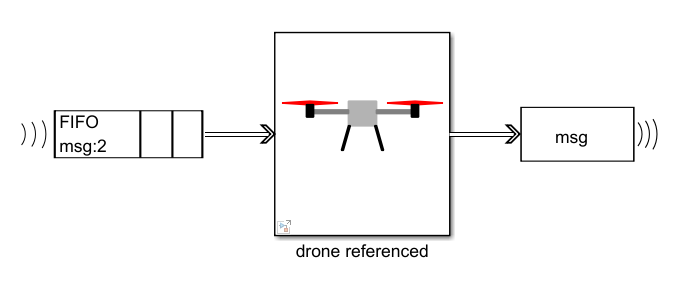
Por último, se han establecido cada uno de los estados por los que puede pasar un UAV:

* 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).

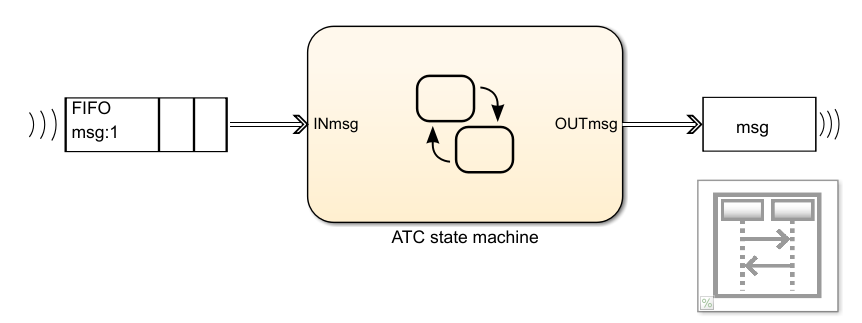
Ambos elementos emiten y reciben mensajes de forma contínua. La recepción de los mensajes se realiza con el bloque ‘*Queue’* que facilita Simulink y mediante el método FIFO (por sus siglas en inglés, First-In First-Out) se antienden y procesan cada mensaje almacenado.

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

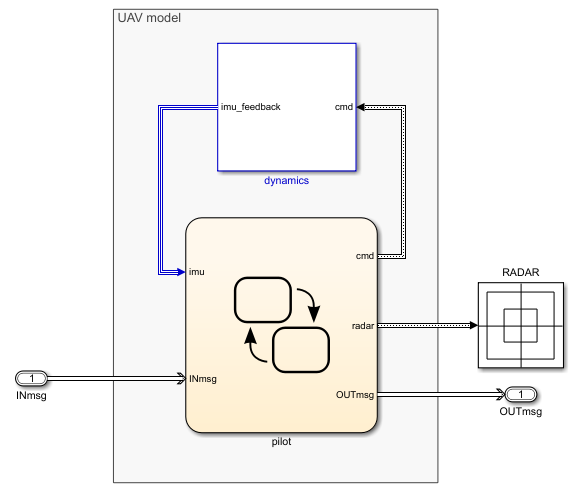
A continuación, se detalla tanto la composición como el funcionamiento de ambos subsistemas para el desarrollo del simulador. Estos subsistemas han sido implementados mediante el entorno gráfico de Simulink que permite construir el modelo a simular mediante bloques.

* 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 ordenes de entrega por parte del ATC.

Los elementos principales que componen este subsistema se pueden observar a través de la Figura 11 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.

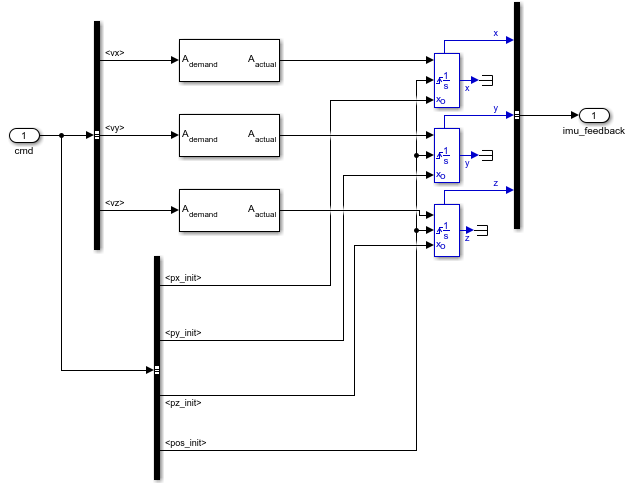


1. Subsistema UAV en Simulink
   * 1. Dynamics

Como se puede observar en la Figura 12 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*’.

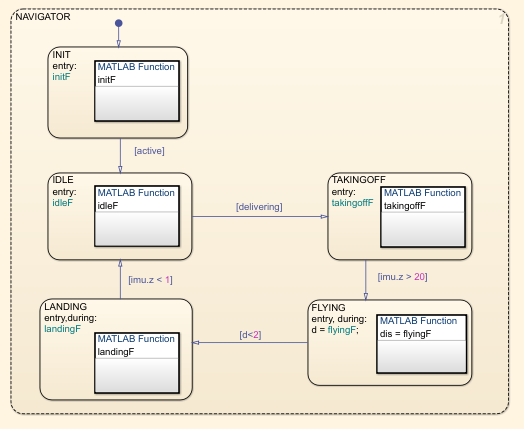
Como se puede observar en la Figura 11 ambos puertos (*cmd* e *imu\_feedback*) son utilizados en ‘*pilot’* que es el corazón del subsistema UAV, contiene la lógica del mismo.



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 13) y el otro la comunicación (ver Figura 14) de cada UAV.

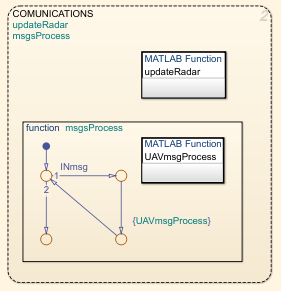


1. Máquina de estado de UAV

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:

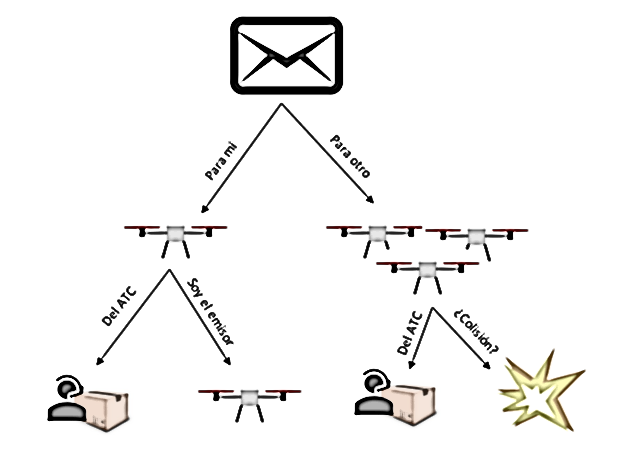
* Init: es el estado inicial y solo se ejecutará la primera vez en cada simulación. A través de este estado se incializará la estructura de estructura de datos local a cada UAV, las posiciones iniciales, el número de UAVs involucrados en la simulación y la variable que permite la transición entre estados.
* 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 esta 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 pararelo al bloque de navegación, aunque con menor prioridad, se encarga del tratamiento de los distintos mensajes recibidos.



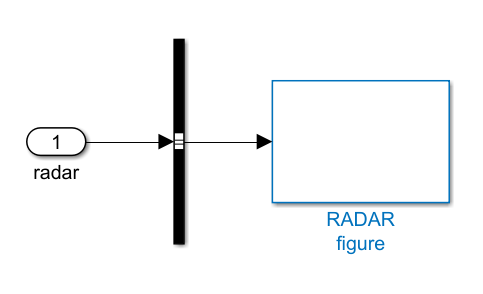
1. Bloque comunicación de UAV

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

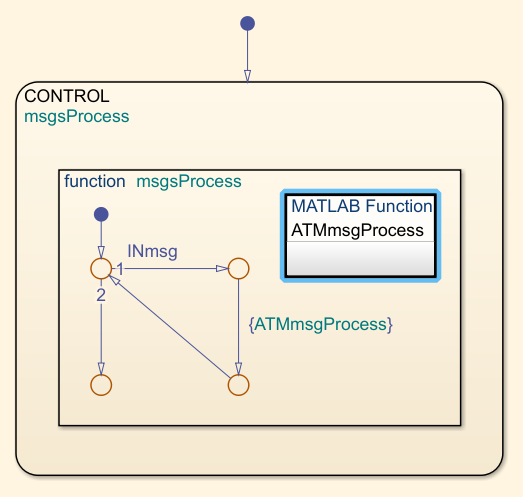
Como puede observarse en la Figura 16, el gráfico 3D que permitirá ver el transcurso de la simulación, es alimentado a través del puerto ‘*radar*’. A través de este puerto se actualiza la información necesaria tal como la posición o el id de cada UAV.



1. Radar en subsistema UAV
   1. Subsistema ATC

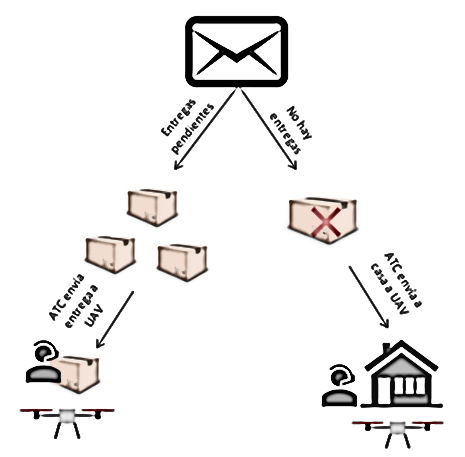
Al igual que ocurre en el subsistema UAV, el ATC también recibe y emite mensajes, principalmente para poder ejecutar órdenes de entrega a cada uno de los UAV.

El subsistema ATC se compone de un único bloque de control (ver Figura 17) que se encarga del procesamiento de dichos mensajes, además incializa la estructura de datos que maneja el ATC para el control sobre las entregas.



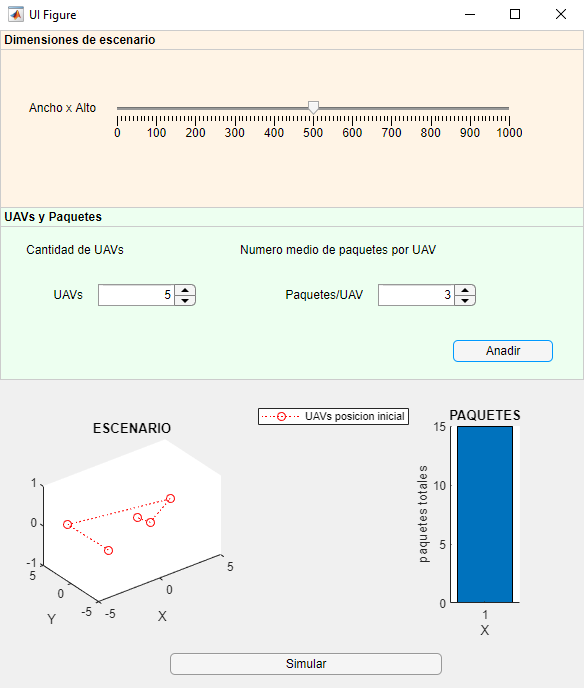
1. Bloque control de ATC

El procesamiento de los mensajes por parte del ATC puede observarse en la Figura 18. 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 19).



1. Interfaz de usuario del simulador
   1. Simulador para testear EL algoritmo

Debido a la complejidad del sistema desarrollado, se ha optado por la implementación de un segundo simulador 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, logramdo un ahorro en tiempo tanto en el desarrollo del algoritmo como en la equicución 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 acabo con la herramienta MATLAB, incluyendo este en un único fichero que 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

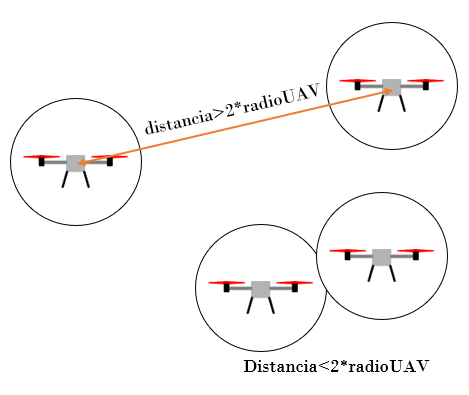
Como ya se había comentado con anterioridad, se entiende por colisión a la situación en la que dos o más agentes superan un umbral de seguridad preestablecido. Los UAVs son modelados como circunferencias para su simplecidad y dotándoles así de dicho margen de seguridad.

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 UAVs existentes. Primero se establece la distancia entre *i* y *j*, mediante sus posiciones en el espacio, *Pi=(x1,y1)* y *Pj=(x2,y2)*.

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

En la Figura 20 puede observarse de forma clara cuando se produce dicha colisión.

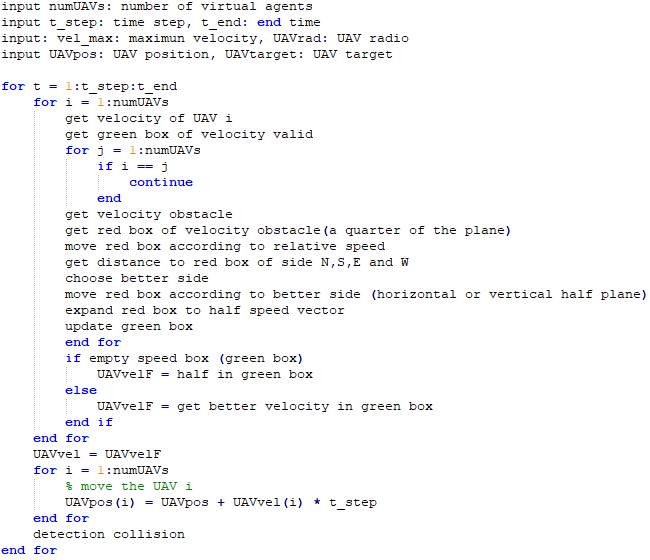


1. Mecanismo para detectar colisiones
   * 1. Mecanismo para evitación de colisiones

El mecanismo que permite evitar las colisiones entre los distintos UAVs que sobrevuelan el espacio aéreo con un determiando fin, como sería el de entrega de paqueteria a diferentes destinos, es una de las partes más importantes e interesantes que engloban a este TFG.

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 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 proyecto, como mecanismo para la evitación de colisiones.



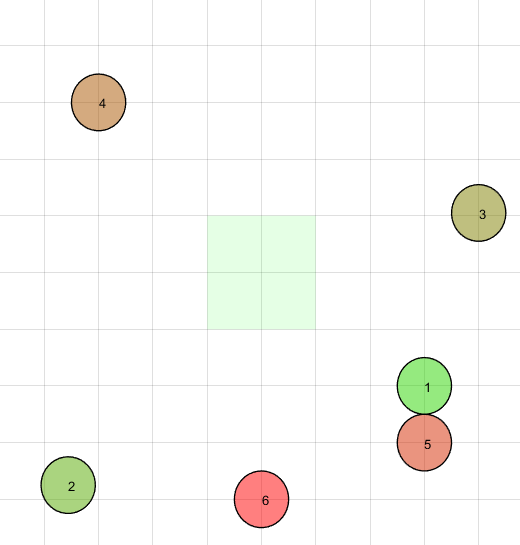
1. Pseudocódigo del algoritmo
   * + 1. Entradas

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 cuatro:

* Número de agentes o UAVs
* Posición de inicio de cada agente o UAV
* Velocidad máxima
* Posición de destino de cada agente o UAV
  + - 1. Velocidad válida

Como se puede observar en la Figura 21 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.

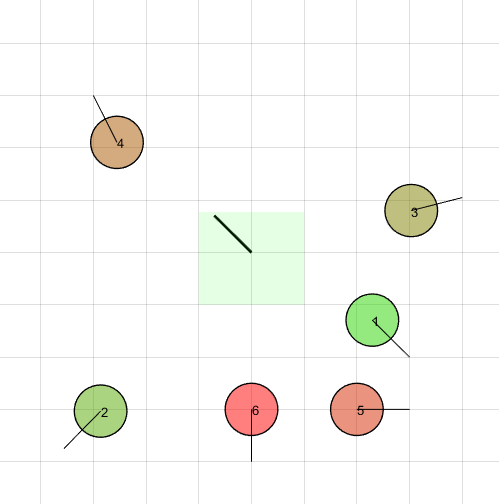
Inicialmente se configura por medio de la velocidad máxima y se actualiza en función de las restricciones introducidas.



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}:



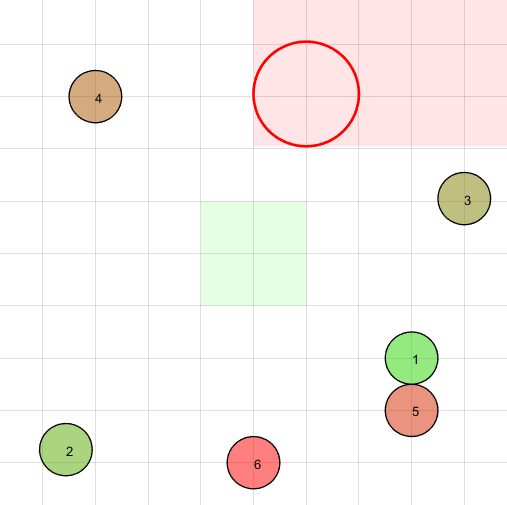
1. Actualización de velocidad válida

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.
  + - 1. Obstáculo de velocidad

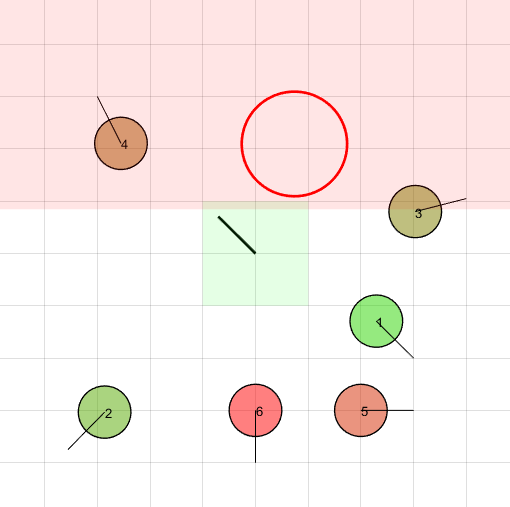
De la misma forma en la que se ha modelado la velocidad permitida o velocidad válida, se ha implementado el obstáculo de velocidad, pero en este caso mediante una caja roja con lados Norte (N), Sur (S), Este (E) y Oeste (W).

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.



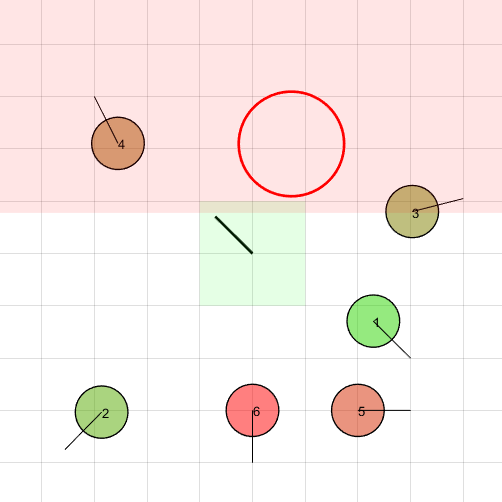
1. Obstáculo velocidad en simulador (agente 1 con 3)

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)

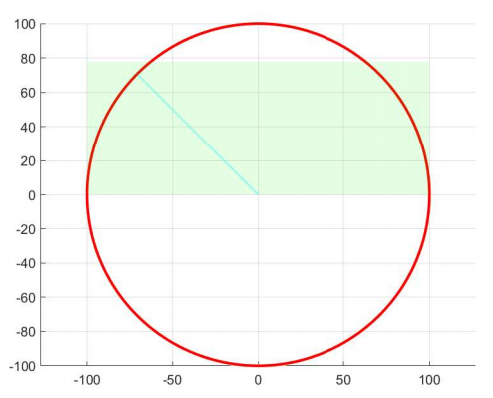
El tercer paso se basa en ampliar dicha restricción (obstáculo velocidad) la mitad del vector velocidad del agente con la dirección óptima calculada anteriormente.



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 27.

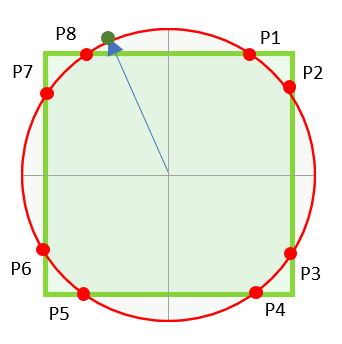


1. Cálculo de velocidad óptima

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

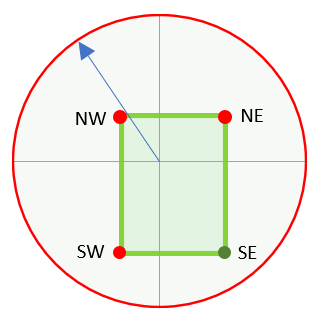
* La velocidad directa al objetivo se encuentra dentro de las velocidades válidas, por lo tanto, se establece como mejor opción dicha velocidad.
* La velocidad directa al objetivo no está dentro de las velocidades válidas: se trata mediante un método que hemos denominado “método de los doce puntos”.

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.



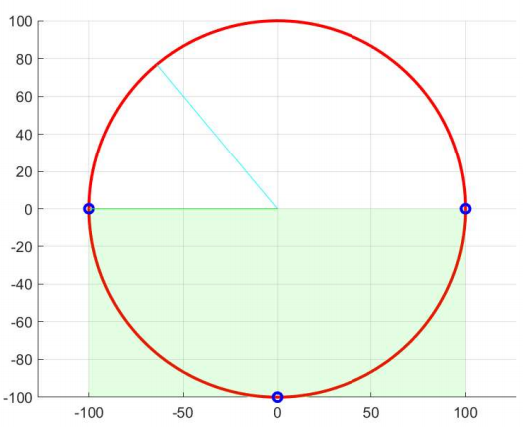
1. Ocho puntos de corte posibles

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 cual 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
2. Experimentos y resultados

Uno de los aspectos más importantes consiste en el estudio y análisis de los datos obtenidos. Este estudio determinará la viabilidad de la solución, o la posible mejora respecto a otros algoritmos, así como aquellos aspectos que caben destacar.

Para ello se ha construido un banco de pruebas por medio de la modificación de aquellas variables que interesan de cara a comparativas o análisis del comportamiento de dicho algoritmo.

* + 1. Simulaciones algoritmo

owpqeu rwiqehr qw ejroñ

* + 1. Otra sección

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

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/. [Accessed: 21-Mar-2020].

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

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

[4] “MATLAB - El lenguaje del cálculo técnico - MATLAB & Simulink.” [Online]. Available: https://es.mathworks.com/products/matlab.html. [Accessed: 21-Mar-2020].

[5] “Simulink - Simulación y diseño basado en modelos - MATLAB & Simulink.” [Online]. Available: https://es.mathworks.com/products/simulink.html. [Accessed: 21-Mar-2020].

[6] “Crear un repositorio Git en Github y subir el código.” [Online]. Available: https://desarrolloweb.com/articulos/crear-repositorio-git-codigo.html. [Accessed: 21-Mar-2020].

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 |