Trabajo Fin de Máster
Máster en Ingeniería Aeronáutica

Simulación de Vuelo de Múltiples UAVs de Ala Fija

Autor: Alberto Lobit Cerrato

Tutor: Aníbal Ollero Baturone

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Máster
Máster en Ingeniería Aeronáutica

Simulación de Vuelo de Múltiples UAVs de Ala Fija

Autor:
Alberto Lobit Cerrato

Tutor:
Aníbal Ollero Baturone
Catedrático

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Máster: Simulación de Vuelo de Múltiples UAVs de Ala Fija

Autor: Alberto Lobit Cerrato
Tutor: Aníbal Ollero Baturone

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Resumen

El objetivo de este Trabajo Fin de Máster es la simulación en el entorno de programación MATLAB® de distintos casos de vuelo con múltiples vehículos aéreos no tripulados (UAVs). Para ello se han diseñado tres programas diferentes en MATLAB® cada uno con distintas simulaciones para ver como actuarían en la vida real los UAVs ante diferentes perturbaciones. Hay un primer programa para barrer áreas rectangulares, pudiéndose desactivar un UAV en plena simulación. Un segundo programa para ver como interaccionan entre sí varios conjuntos de UAVs al localizar objetivos para después profundizar más en esos objetivos con otro UAV y un tercer programa que se parece al primero, pero en el cual, los UAVs de un único conjunto despegan y aterrizan de una base para barrer un área cuya única condición es que sea convexa. Se han utilizado diversas interfaces gráficas para modificar cómodamente los parámetros de los programas por parte del usuario.

Abstract

The aim of this Master's Work is the simulation programming environment MATLAB® different flight cases with multiple unmanned aerial vehicles (UAVs). This work has been designed three different programs in MATLAB® , each one with different simulations to see how they would act in real life UAVs to various disturbances. There is a first program to sweep rectangular areas, being able to deactivate a UAV in full simulation. A second program to see how they interact with each other several sets of UAVs to locate targets for later deeper into these objectives with other UAVs and a third program that looks like the first, but in which UAVs of a single set of UAVs off and landing of a base to sweep an area whose only condition is that is convex. They have used various graphical interfaces to easily modify the parameters of the programs by the user.

Índice Abreviado

Índice

1 Introducción

El presente documento trata sobre simulaciones del vuelo de múltiples UAVs en el entorno MATLAB® (versión R2015a). Se ha decidido realizar diferentes programas para simular diferentes acciones sobre múltiples UAVs, para comprobar como funcionarían en la vida real dichas situaciones. Hay que tener en cuenta que al volar múltiples UAVs hay muchas formas de hacerlo, desde volar en formación, hasta volar como si todos los UAVs fueran uno solo. En este trabajo, los UAVs van a volar como si no supieran de ningún otro UAV, sino recibiendo información del sistema y completando la misión que se le asigna, ya será el sistema el encargado de pensar por cada uno de los UAVs.

Figura 1.1 Múltiples UAV en vuelo.

En este capítulo se va a proceder a explicar la historia de los UAVs, desde el inicio de los mismos hasta el siglo XXI, se va también a realizar un estudio sobre la mejora que hay entre utilizar un UAV o múltiples UAVs y los tipos de múltiples UAVs que hay. Por último se va a explicar el enfoque del trabajo comentando capítulo a capítulo el desarrollo del trabajo.

1.1 Historia de los UAVs

La idea del avión no tripulado es más antigua de lo que uno pudiera pensar. Hoy en día se piensa que un UAV (*drone*) son los aviones militares de hoy, pero más allá de esa afirmación, hay que decir que se han utilizado durante décadas. En la historia de los UAVs se ha pasado por varias fases de utilización de UAVs:

- En primer lugar, se utilizaban los UAVs como blanco de práctica para las fuerzas militares a principios del siglo XX.
- Después, en el período de entre las dos Guerras Mundiales y durante el transcurso de la Segunda Guerra Mundial, los UAVs fueron usados como especie de bombas volantes con el fin de dañar las líneas enemigas.
- En tercer lugar, durante la Guerra Fría, el avión no tripulado fue visto como una plataforma de vigilancia viable capaz de capturar datos de inteligencia en áreas de difícil acceso.
- Por último, los UAVs se usan como un arma que fusiona la capacidad de vigilancia y la de matar.

A continuación se va a realizar un resumen de la historia de los UAVs [1] [2] [13].

- Julio de 1849: los austriacos estaban en guerra con Venecia, y decidieron lanzar un ataque de doscientos globos aerostáticos no tripulados cargados con bombas.

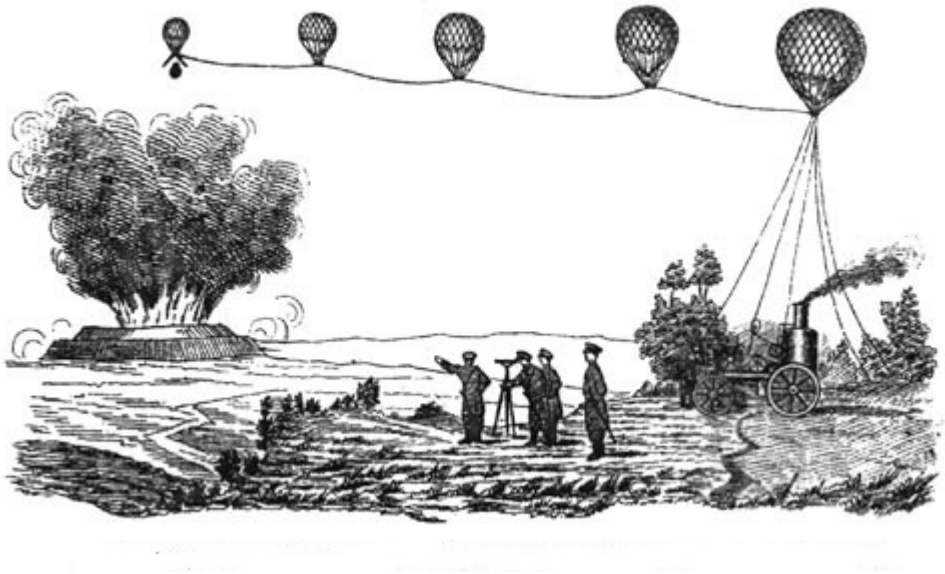

Figura 1.2 Globos de Austria contra Venecia.

- En 1898: en la Guerra Hispano-Americana, los militares de Estados Unidos, equiparon una cámara a una cometa, dando lugar a una de las primeras fotografías de reconocimiento aéreo.

Es evidente que estos dos precedentes no tienen nada que ver con la idea que se tiene actualmente de los UAV, pero eso no quita que los precursores de los UAVs actuales fueran estos casos.

- Antes de proseguir hablando de UAVs es necesario hacer una parada en Nikola Tesla porque en 1898 en Nueva York, fue capaz de controlar a distancia un barco con una señal de radio, es decir, el radio control. Esta nueva etapa abría numerosos caminos para seguir evolucionando los UAVs.
- En la Primera Guerra Mundial, se utilizó ampliamente la vigilancia aérea. Los militares utilizaban cometas para obtener fotografías aéreas y seguir los movimientos de los enemigos. También, el inventor del giroscopio Elmer Ambrose Sperry, desarrolló una plataforma de aeronaves sin piloto con un dispositivo para lanzar torpedos con una catapulta. La idea era que estas *aeronaves*, fueran guiadas hasta el objetivo y volaran hacia abajo, en vertical, con una carga de TNT.

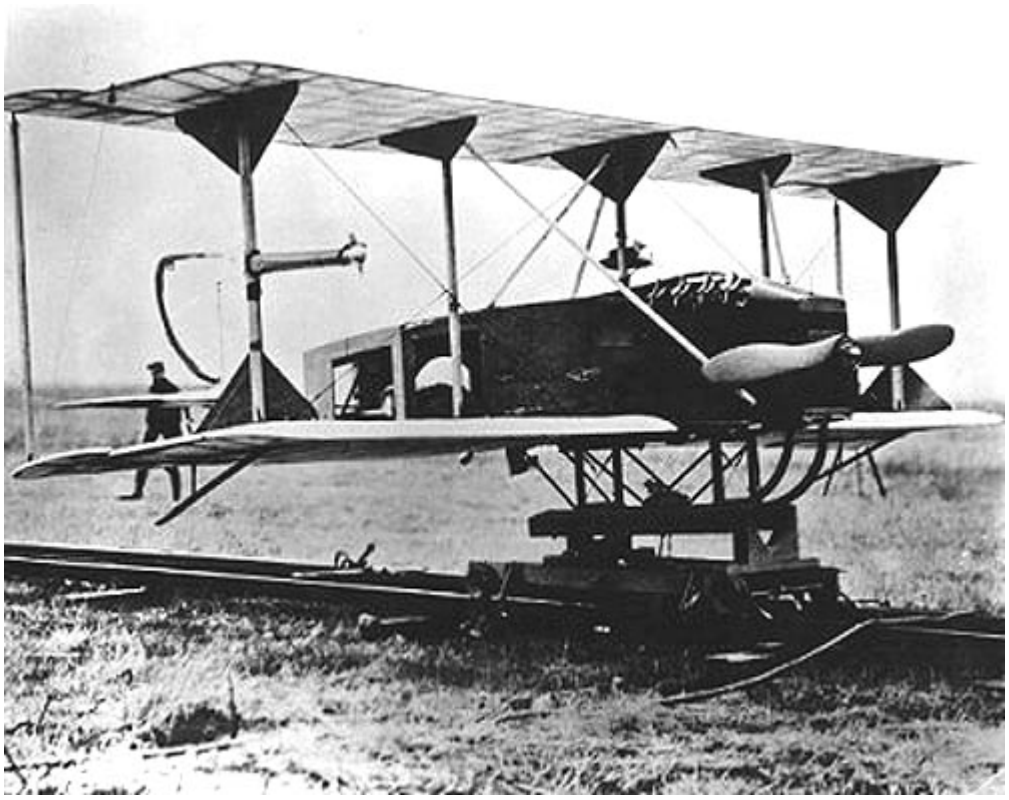

Figura 1.3 Hewitt-Sperry.

Al tener tanto éxito este avión al introducirle la tecnología giroscópica, se decidió llevar a cabo un segundo proyecto que se denominó *Kettering*, que era un torpedo sin piloto y guiado por controles preestablecidos. En Alemania, también se estaba llevando a cabo un proyecto similar que dio lugar al *Siemens Torpedo Planeador* que era un misil que se valía de un Zeppelin y luego se guiaba hasta su objetivo por radio. Estos casos son los precursores de los misiles actuales.

- Durante la Segunda Guerra Mundial, se desarrolló el GB-1 Glide que era un sistema de bombardero cuya función era planear hasta su objetivo guiado por radio.

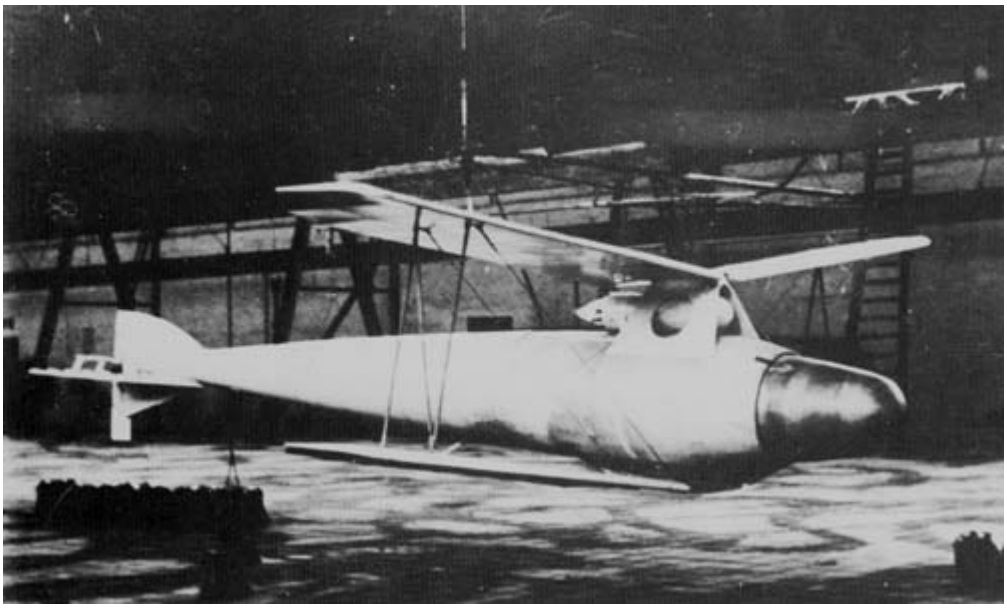

Figura 1.4 Siemens Torpedo Planeador.

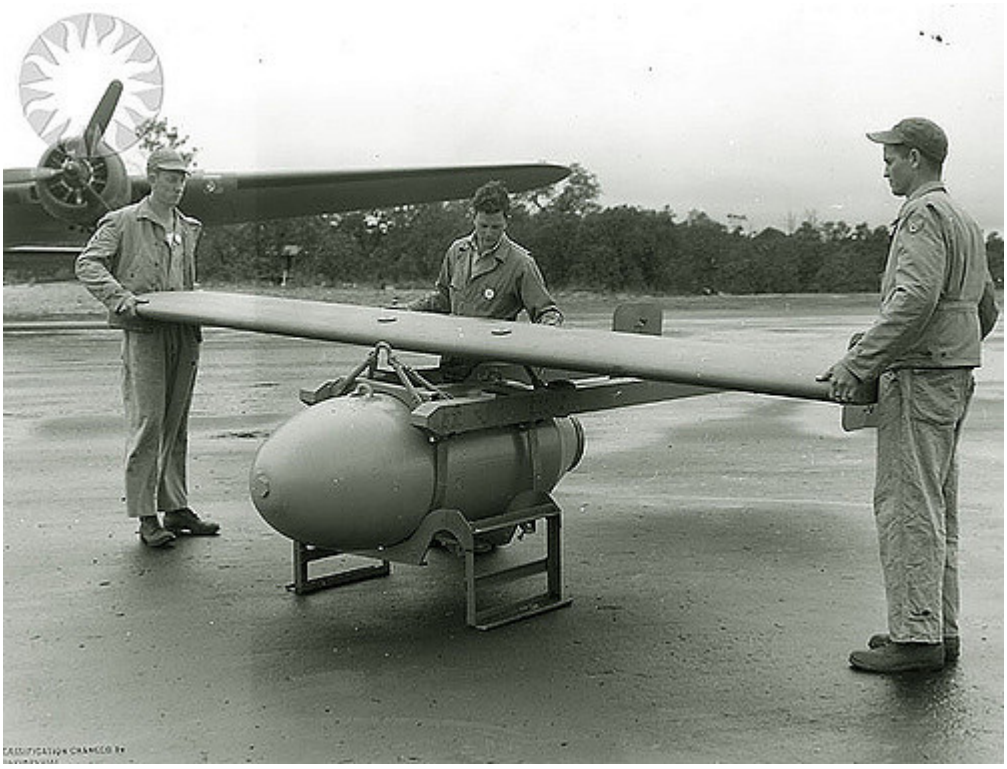

Figura 1.5 GB-1 Glide.

La pérdida de 40.000 aviones y 80.000 personas de tripulación durante la Segunda Guerra Mundial fueron motivos que hicieron centrar el interés en conformar una fuerza aérea robótica en Estados Unidos, una cuestión financiera y humana que había que resolver.

- En 1946, una vez finalizada la Guerra, en Estados Unidos se aprobó el desarrollo de tres tipos de aviones no tripulados para su uso como *diana* para ayudar en la

formación.

- La Guerra de Vietnam fue la causa más importante para el desarrollo de los UAVs porque esta guerra fue la primera guerra tecnológica, siendo la primera en la cual se automatizó y informatizó el campo de batalla con sensores y ordenadores.

  En 1962, se desarrollo el *Gran Safari* que consistía en que un *C-130 Hércules* actuaba como nave nodriza para coordinar a todo el enjambre de aviones no tripulados (*Firebees*) que se soltaban desde el propio avión nodriza. Estos aviones no tripulados volaban en rutas preprogramadas y también podían ser programados por los operadores de radios que se encontraban dentro de la nave nodriza. Después de la misión de vigilancia, caían al suelo en paracaídas y se recuperaban con un helicóptero.

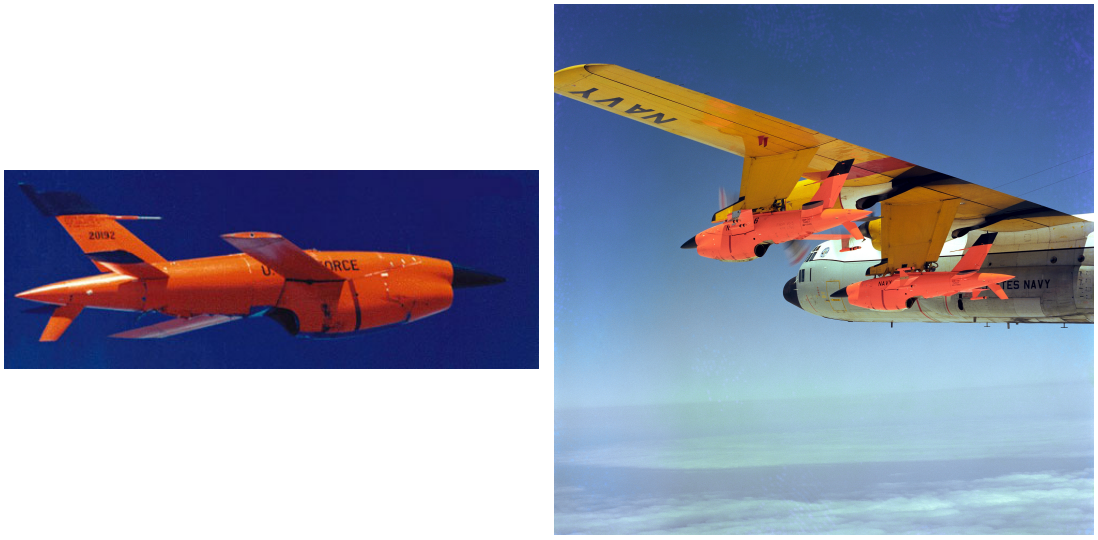

Figura 1.6 Firebees desplegado y en la nave nodriza.

- En los años posteriores, década de 1970, se siguió investigando en mejorar los UAVs porque los ya existentes, tenían las capacidades para llevar cámaras y sistemas de guiado sencillos, pero no tenían las capacidades suficientes para volver a la base. Se hicieron prototipos teledirigidos de vigilancia que eran capaces de aguantar más de 24 horas en el aire, siendo pilotados desde el suelo. También se realizaron prototipos de aviones no tripulados pequeños que eran capaces de llevar láseres y cámaras de vídeos.

  En esta década, concretamente en 1971, fue la primera derrota de un humano frente a un UAV porque no era capaz de seguir el ritmo de los giros y vueltas del mismo.

- Durante la década de los 80 y 90 se desarrolló fuertemente la computación y los sistemas de control electrónicos y fue a finales de los 90 que se empezaron a dotar los UAVs de misiles.

- En en siglo XXI, apareció el predator, el primer UAV diseñado para vigilancia de larga duración y a gran altitud. Con el paso de los años se ha ido modificando hasta ser una mezcla entre UAV de vigilancia y un avión furtivo.

  Aunque los avances en aviones no tripulados fueron impulsados por las exigencias de la inteligencia cartográfica, estos objetos no tripulados estaban muy ligados a una

Figura 1.7 Predator.

serie de objetos legales que permitieron su despliegue, es decir, la relación entre la tecnología y la ley es extremadamente importante en el trazado de la escalada del UAV Predator. Ambos elementos van de la mano en la adquisición de conocimiento geográfico y vigilancia, consecución de objetivos, y en última instancia, la toma de decisiones sobre la eliminación de objetivos.

1.2 Un UAVs vs. múltiples UAVs

A la hora de realizar una misión, es necesario conocer si es mejor hacerla con un solo UAV o con varios UAVs porque cada manera de actuar tiene sus ventajas y desventajas [8] [11] [12] [14]. A continuación se muestran las ventajas y desventajas de usar múltiples UAVs respecto de un sólo UAV.

- Ventajas:
  - Intervenciones simultáneas en distintas localizaciones.
  - Mayor eficiencia en la exploración y búsqueda de objetivos.
  - Más seguridad, por ejemplo, al haber varios UAVs si falla alguno de ellos, no implica que no se pueda llevar a cabo la misión.
- Desventajas:
  - Sistemas más complicados porque hay que tener en cuentas las variables de cada uno de los UAVs.
  - Mayor coste respecto al de un solo UAV.

Aún así, un buen sistema de múltiples UAVs debe tener las siguientes características:

- Robusto, es decir, que si falla en algún punto, el sistema pueda seguir operando aunque pierda propiedades.

- Eficiencia, aún cuando el sistema esté inmerso en condiciones dinámicas.
- Rápida respuesta a los cambios.
- Capacidad de funcionamiento aun cuando las comunicaciones no sean perfectas.
- Capacidad de asignar recursos limitados.
- Capacidad de hacer uso de las capacidades de los robots aún cuando éstas no sean las mismas para cada uno de ellos.

1.3 Clasificación de múltiples UAVs

A continuación se van a desarrollar dos tipos de múltiples UAVs, que son el vuelo en formación y swarm.

1.3.1 Vuelo en formación

Se considera vuelo en formación al vuelo disciplinado de dos o más aeronaves bajo el mando de un líder de vuelo. Es una de las maniobras más delicadas en el mundo de la aviación debido a la proximidad entre las aeronaves, por lo que en la realidad sólo se utiliza en el ámbito militar y únicamente se emplea en el ámbito civil como espectáculo aéreo. En el ámbito de los UAVs sí se puede utilizar y tiene las siguientes características:

- Los movimientos relativos entre aeronaves están fuertemente limitados para no romper la formación.
- El plan de movimiento es requerido por parte de todos los UAVs que componen la formación.
- La evitación de colisiones debe estar introducido en el control de los UAVs.

1.3.2 Swarm

Swarm o *enjambre* se refiere a las redes de UAVs capaces de reconfigurarse dinámicamente a medida que cambia la situación de la red [6] [7] [10]. Esto proporciona una gran flexibilidad a la hora de realizar misiones dinámicas. Los UAVs son capaces de realizar una misión conjuntamente, decidiendo en cada momento que es lo mejor para el sistema de manera conjunta.

- Las interacciones generan comportamientos colectivos emergentes.
- No es obligatorio el movimiento en formación.

1.4 Enfoque del trabajo

En este trabajo no se ha desarrollado un sistema swarm de UAVs, pero sí se ha desarrollado un sistema más complejo que el del vuelo en formación, es decir, en este trabajo el sistema da órdenes a cada UAV por separado evitando tener que realizar la misión en formación,

Figura 1.8 Vuelo en formación.

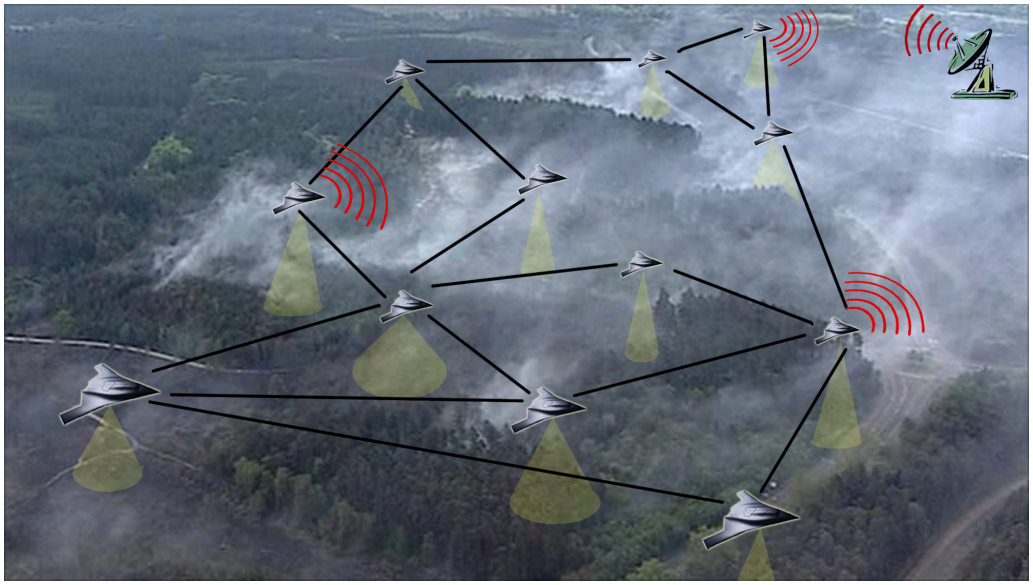

Figura 1.9 Swarm.

pero no llega a ser un sistema swarm porque cada UAV no *sabe* que misión está realizando conjuntamente con los demás, sino que cada UAV realiza su misión y no es capaz de saber qué les pasa a los otros UAVs.

Con el objetivo del desarrollo de programas de simulación de múltiples UAVs, el primer paso es conocer las ecuaciones mediante las cuales se han desarrollado todos los programas del trabajo (capítulo 2). También es necesario el conocimiento de la base de programación a objetos (capítulo 3) para aportar una mayor sencillez al código de los programas realizados.

En este trabajo se han realizado tres programas totalmente diferentes unos con otros. El primero de ellos, capítulo 4, se basa en un conjunto de UAVs, es decir, se va a simular un único conjunto de UAVs para ver como barre un área descrita con anterioridad, dicha área

va a resultar un rectángulo cuyas dimensiones son modificables.

En este programa se explica el modelo seguido para los UAVs, incluyendo el movimiento que va a realizar cada uno de ellos, y el movimiento del conjunto de UAVs. Se le ha incorporado al programa la capacidad de desactivar un UAV en mitad de la simulación lo que confiere al programa una mayor realidad porque si se desactiva un UAV, el resto debe reajustarse para completar el objetivo.

Se explica como se ha modelado en MATLAB® siguiendo los valores del capítulo de programación a objetos y mediante una interfaz gráfica que se explica en el capítulo se pueden modificar los parámetros del programa para aportar una mayor sencillez al usuario. Por último, en este capítulo se ha añadido una sección de posibles resultados que arroja el programa ante distintas simulaciones y se comprueba que dichas simulaciones se parecen a la realidad.

El segundo programa, capítulo 5, consiste en la búsqueda de objetivos por parte de cuatro conjuntos de UAVs diferentes, en las simulaciones se aprecia un UAV líder que realiza circunferencias completas en torno al área que se quiere buscar, dicha área debe ser un rectángulo. Del UAV líder salen los diferentes grupos de UAVs de uno en uno y cada uno tiene asignado un número concreto de UAVs. Dichos UAVs barren el área que le toca a cada uno en busca de uno de los objetivos. Cuando un objetivo es encontrado, del UAV líder sale un UAV hunter que se encarga de dirigirse a cada uno de los objetivos encontrados y realizar una inspección más en profundidad para saber qué clase de objetivo es.

La manera de explicar este programa es similar al del anterior programa, es decir, primero se define el modelo que se ha seguido para los UAVs (líder, hunter y los de barrido) y se explica el modelo seguido por los conjuntos de UAVs. Después se aclara el modelo en el entorno MATLAB® que se ha seguido para hacer el código del programa.

Por último, se han realizado una serie de interfaces gráficas para ayudar al usuario en el manejo de las variables y se ha añadido una última sección de resultados del programa que, al igual que en el programa anterior, se centra en comparar los resultados con la realidad para saber si está correctamente realizado.

El último programa realizado, capítulo 6, es el más realista de los tres programas, y se basa en el barrido de áreas por parte de un conjunto de UAVs, pero la diferencia con el primer programa radica en que en éste los UAVs salen de una base desde la cual despegan y vuelven a aterrizar una vez completada su misión. Para la realización de este programa ha sido necesario tomar los datos de un UAV real, en este caso, se ha optado por tomar los datos del *Aerosonde UAV*, apéndice A. El área de este programa también se puede definir y no tiene por qué ser un rectángulo, pero sí tiene que ser un área convexa.

Se ha explicado primero el modelo utilizado para los UAVs, cómo se ha repartido el área a cada UAV, cómo se han dado los waypoints según el área a cada UAV. Para finalizar la parte del modelo explicando el movimiento de los UAVs. El modelado en MATLAB® ha seguido las directrices del capítulo 3 y se explica con todo el detalle posible.

Al igual que en los capítulos anteriores se ha optado por crear unas interfaces gráficas que ayudan a manejar todos los datos del programa por parte del usuario y se ha finalizado el capítulo con los resultados que arroja el programa para compararlos con la realidad y con los programas anteriores. También se han comparado los resultados con distintas posibles misiones de vuelo de UAVs.

Las conclusiones, así como propuestas para trabajos futuros, se incluyen en el capítulo 7.

2 Ecuaciones

En este capítulo se van a explicar las ecuaciones usadas para el desarrollo del trabajo. Se ha seguido como guía para la implementación de las mismas los desarrollos del libro de mecánica de vuelo [15].

2.1 Hipótesis

Para estudiar las actuaciones de un avión, se va a considerar el movimiento del centro de masas del mismo, considerado como un cuerpo puntual de masa constante con tres grados de libertad. Para la formulación que a continuación se presenta, se han considerado las siguientes hipótesis.

- El cuerpo del avión es rígido.
- El avión es simétrico.
- El motor es fijo respecto del avión.
- La tierra es plana, que dado las distancias en las que se va a mover el trabajo tiene sentido aplicarla.
- La gravedad es constante.
- La atmósfera está en calma, lo que implica que no hay viento.

2.2 Sistema de referencia

Para elegir el sistema de referencia, hay que ver qué se quiere hacer en el trabajo exactamente para poder elegirlo correctamente [4]. En este trabajo se va ha simular una serie de UAVs en vuelo, así que interesa un sistema de referencia global y no uno por cada UAV que haya en vuelo. Por lo tanto, el sistema de referencia elegido es el sistema inercial topocéntrico.

2.2.1 Sistema inercial topocéntrico

El sistema inercial topocéntrico consiste en un sistema de referencia inercial newtoniano solidario a La Tierra (figura 2.1). Los elementos de este sistema de referencia se definen como:

- El centro del sistema de ejes se encuentra en cualquier punto de la superficie terrestre.
- x: dirección norte.
- y: dirección este.
- z: completa un triedro a derechas (dirigido hacia abajo).

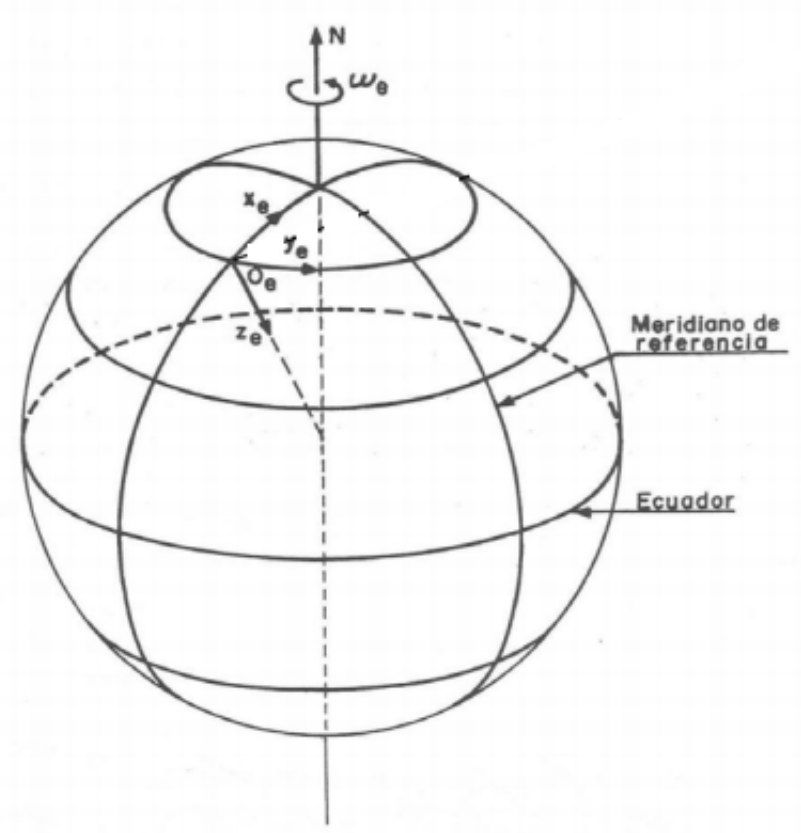

Figura 2.1 Sistema inercial topocéntrico.

En este trabajo se ha modificado ligeramente la definición del sistema inercial topocéntrico, quedando de la siguiente manera.

- x: dirección este.
- y: dirección norte.
- z: completa un triedro a derechas (dirigido hacia arriba).

También se va a presentar el sistema de ejes de horizonte local porque a partir de este sistema se van a definir las ecuaciones generales del avión.

2.2.2 Sistema de ejes horizontal local

Este sistema se caracteriza porque las direcciones de los tres ejes son paralelas a las del sistema anterior (figura 2.2). Los tres ejes en este sistema de referencia se definen como:

- El centro de este sistema de ejes se encuentra en el centro de masas de la aeronave.
- x_H: eje paralelo al eje x del sistema inercial topocéntrico.

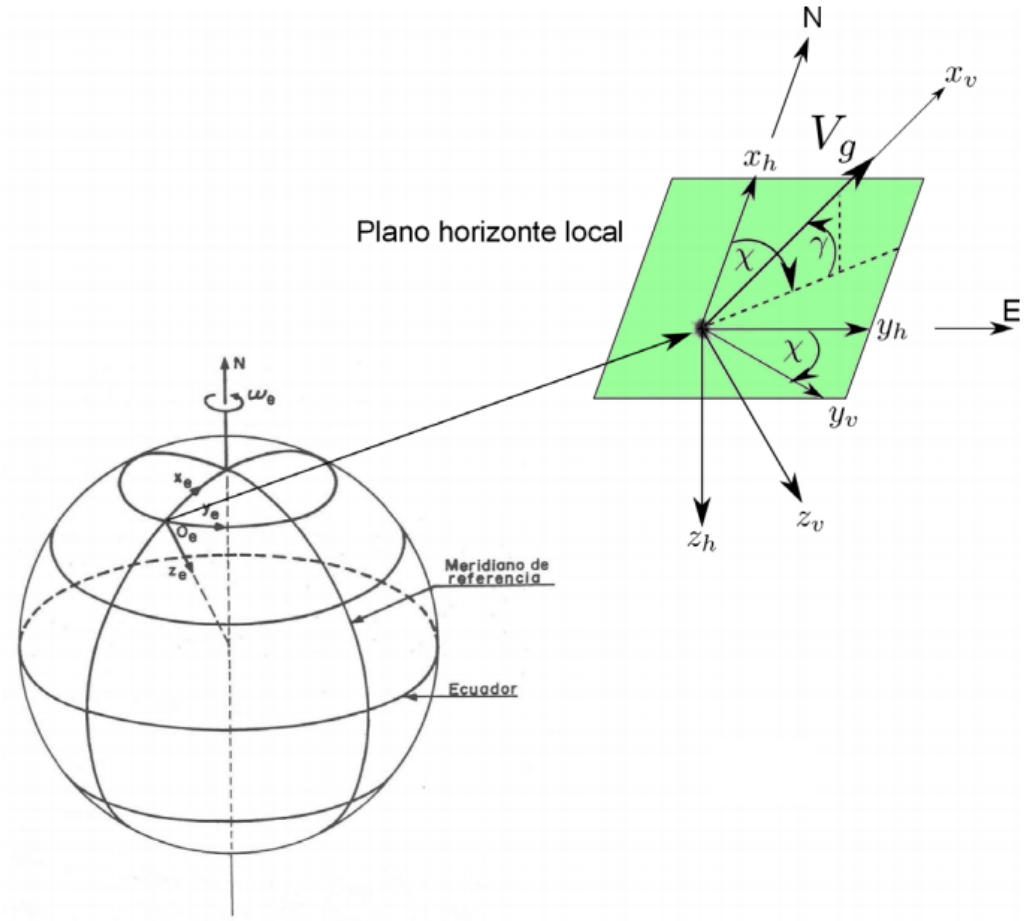

Figura 2.2 Sistema de ejes horizontal local.

- y_H: eje paralelo al eje y del sistema inercial topocéntrico.
- z_H: completa un triedro a derechas (paralelo al eje z del sistema inercial topocéntrico).

El plano horizonte local lo forman el eje x_H y el y_H.

2.3 Modelo de atmósfera

El modelo de atmósfera proporciona la temperatura, la presión y la densidad del aire en función de la altitud. Para este trabajo se va a considerar el modelo de atmósfera estándar internacional, modelo ISA[1]. A continuación se muestran las características de este modelo. Se va a suponer que en este trabajo no se va a subir por encima de una altitud superior a

[1] ISA: International Standard Atmosphere.

11.000 metros.

$$\begin{aligned} T &= T_0 - \alpha_T h \\ P &= P_0 \left(1 - \frac{\alpha_T h}{T_0}\right)^{\frac{g}{R_a \alpha_T}} \\ \rho &= \rho_0 \left(1 - \frac{\alpha_T h}{T_0}\right)^{\frac{g}{R_a \alpha_T} - 1} \end{aligned} \tag{2.1}$$

en dónde los valores de las constantes son:

- T_0: 288,15 K.
- P_0: $1{,}01325 \cdot 10^5$ N/m^2.
- ρ_0: 1,225 kg/m^3.
- α_T: $6.5 \cdot 10^{-3}$ K/m.
- R_a: 287,05 J/(kgK).
- g: 9,80665 m/s^2.

2.4 Modelo de avión

Aquí se van a modelar las fuerzas aerodinámicas y propulsivas, que van a depender en mayor o menor medida de la altitud a la que nos encontremos.

Las fuerzas aerodinámicas son la resistencia y la sustentación, que se caracterizan de la siguiente manera:

$$L = \frac{1}{2}\rho(h)V^2 S C_L \tag{2.2}$$

$$D = \frac{1}{2}\rho(h)V^2 S C_D \tag{2.3}$$

siendo S la superficie alar del avión.

Para obtener el coeficiente de sustentación (C_L) y el coeficiente de resistencia (C_D), es necesario seleccionar un avión del cual se tengan todos los datos aerodinámicos. En el apéndice A están expuestos todos estos datos.

Dados estos valores, se definen los coeficientes aerodinámicos como:

$$C_L = C_{L0} + C_{L\alpha}\alpha \tag{2.4}$$

$$C_D = C_{D0} + C_{D\alpha}\alpha \tag{2.5}$$

El modelo aerodinámico esta resuelto, ahora queda por resolver el modelo propulsivo, que se va resolver con la ayuda de [5]. Obteniéndose un empuje de:

$$T = \delta_T T_{SL} \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma - 1}{\gamma}} \left(1{,}00 - 0{,}49\sqrt{M}\right) \frac{\rho}{\rho_{SL}} \tag{2.6}$$

en donde:

- Posición de la palanca de gases (δ_T): varía entre 0 y 1,15.
- Empuje a nivel del mar (T_{SL}).
- Coeficiente de dilatación adiabática (γ): 1,4.
- Densidad a nivel del mar (ρ_{SL}): 1,225 kg/m^3.
- Mach de vuelo (M).

El Mach de vuelo se puede descomponer en:

$$M = \frac{V}{a} = \frac{V}{\sqrt{\gamma R_a T(h)}} \tag{2.7}$$

siendo a la velocidad del sonido para una altura determinada.

Una vez que se tienen resueltos los modelos aerodinámicos y propulsivos del avión, se puede empezar a introducir las ecuaciones del avión, que se van a dividir en:

- Ecuaciones en crucero.
- Ecuaciones en viraje.
- Ecuaciones de despegue.
- Ecuaciones de aterrizaje.

Pero primero se van a exponer las ecuaciones generales que gobiernan el movimiento de un avión, basándose en la figura 2.2 que se corresponde con el sistema de ejes horizontal local y en la figura 2.3 que hace referencia a la orientación del empuje con respecto a los ejes viento, pero como se ha supuesto atmósfera en calma como hipótesis, entonces, se está refiriendo a los ejes horizontal local.

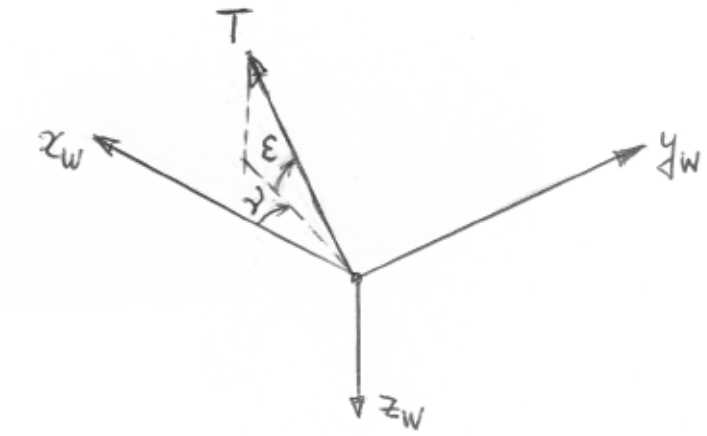

Figura 2.3 Orientación del empuje.

Las ecuaciones están bajo la hipótesis de vuelo simétrico, lo que hace que ν (figura 2.3) sea nulo.

$$\begin{aligned}
\frac{dx}{dt} &= V cos\gamma cos\chi \\
\frac{dy}{dt} &= V cos\gamma sin\chi \\
\frac{dh}{dt} &= V sin\gamma \\
m\frac{dV}{dt} &= T cos\varepsilon - D - mgsin\gamma \\
mV cos\gamma\frac{d\chi}{dt} &= (L + T sin\varepsilon)sin\mu \\
mV\frac{d\gamma}{dt} &= (L + T sin\varepsilon)cos\mu - mgcos\gamma
\end{aligned} \tag{2.8}$$

siendo μ el ángulo de balance de velocidad o ángulo de alabeo.

Teniendo éstas ecuaciones generales, se pueden deducir las ecuaciones para vuelo simétrico en un plano vertical y en un plano horizontal.

2.4.1 Ecuaciones generales en un plano vertical

El vuelo en un plano vertical está definido por la condición $\chi = cte$. Por comodidad se va a tomar $\chi = 0$ y por tanto $y = cte$. Al ser en un plano vertical el ángulo de alabeo es nulo ($\mu = 0$). Así pues, las ecuaciones quedan como:

$$\begin{aligned}
\frac{dx}{dt} &= V cos\gamma \\
\frac{dh}{dt} &= V sin\gamma \\
m\frac{dV}{dt} &= T cos\varepsilon - D - mgsin\gamma \\
mV\frac{d\gamma}{dt} &= (L + T sin\varepsilon) - mgcos\gamma
\end{aligned} \tag{2.9}$$

2.4.2 Ecuaciones generales en un plano horizontal

EL vuelo horizontal está definido por $h = cte$. Por lo que se deduce que si no hay cambio de altitud, el ángulo de ascenso/descenso es nulo ($\gamma = 0$). Las ecuaciones quedan:

$$\begin{aligned}
\frac{dx}{dt} &= V cos\chi \\
\frac{dy}{dt} &= V sin\chi \\
m\frac{dV}{dt} &= T cos\varepsilon - D \\
mV\frac{d\chi}{dt} &= (L + T sin\varepsilon)sin\mu \\
0 &= (L + T sin\varepsilon)cos\mu - mg
\end{aligned} \tag{2.10}$$

De aquí en adelante, se considera que el empuje está alineado con la velocidad con lo que $\varepsilon = 0$.

2.5 Ecuaciones en crucero

En el estudio de las actuaciones de punto se considera el problema casi estacionario, es decir, se desprecian las aceleraciones tangencial y normal. Teniendo en cuenta ésto las ecuaciones 2.10 pasan a ser:

$$\begin{aligned} T - D - W sin\gamma &= 0 \\ L - W cos\gamma &= 0 \end{aligned} \tag{2.11}$$

Dado que en el crucero la altitud es constante, se cumple que $\gamma = 0$. Por lo que las ecuaciones 2.11 pasan a ser de la forma:

$$\begin{aligned} T - D &= 0 \\ L - W &= 0 \end{aligned} \tag{2.12}$$

2.6 Ecuaciones en viraje

En viraje se va a suponer que la velocidad es constante, con lo que partiendo de las ecuaciones 2.10 se llega a:

$$\begin{aligned} \frac{dx}{dt} &= V cos\chi \\ \frac{dy}{dt} &= V sin\chi \\ T &= D \\ mV\dot{\chi} &= L sin\mu \\ L cos\mu &= W \end{aligned} \tag{2.13}$$

en dónde la aceleración norma $V\dot{\chi}$ no es despreciable, ya que las variaciones de χ son importantes. El factor de carga durante la maniobra es:

$$n = \frac{1}{cos\mu} \tag{2.14}$$

Las ecuaciones del movimiento también pueden escribirse como:

$$\begin{aligned} \frac{dx}{dt} &= V cos\chi \\ \frac{dy}{dt} &= V sin\chi \\ T &= D \\ tan\mu &= \frac{V\dot{\chi}}{g} \\ n &= \frac{1}{cos\mu} \end{aligned} \tag{2.15}$$

El radio de giro viene definido por:

$$\frac{1}{r} = \frac{d\chi}{ds} = \frac{\dot{\chi}}{V} \tag{2.16}$$

siendo s el parámetro de longitud de arco.

2.7 Ecuaciones de despegue

La maniobra de despegue va desde la suelta de frenos en cabecera de pista hasta que el avión alcanza una velocidad y altura definidas en las normas de aeronavegabilidad. Esta maniobra se efectúa con empuje máximo, flaps en posición de despegue (si la aeronave dispone de dispositivos hipersustentadores) y el tren de aterrizaje extendido. El despegue se compone de varias fases, como muestra la figura 2.4.

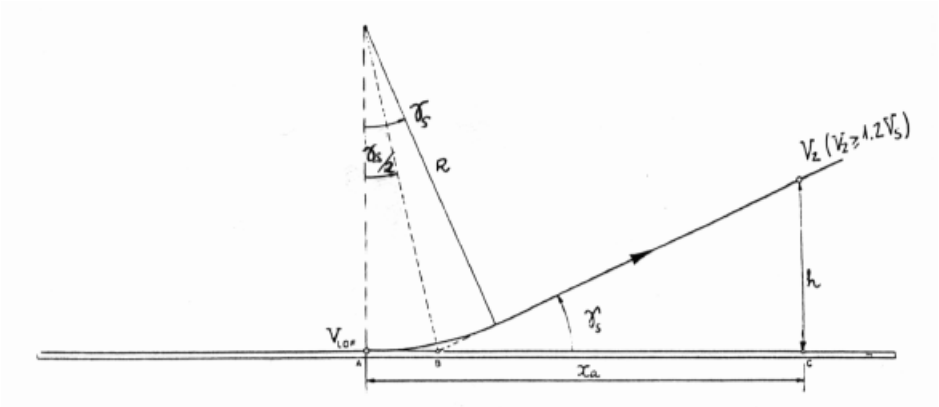

Figura 2.4 Fases en el despegue.

- Rodadura en el suelo: desde la suelta de frenos hasta que el avión alcanza la velocidad de despegue, V_{LOF}, y deja de estar en contacto con la pista.
- Recorrido en el aire: desde que el avión se va al aire hasta alcanzar una altura de 35 pies respecto del suelo y una velocidad $V_2 = 1{,}2V_s$. Esta fase consta de dos partes:
  - Transición curvilínea: desde que el avión deja de estar en contacto con la pista hasta que alcanza el ángulo de subida deseado.
  - Subida rectilínea: el avión se acelera en una subida rectilínea hasta alcanzar la velocidad V_2 a la altura deseada.

siendo V_s la velocidad de entrada en pérdida del avión y $V_{LOF} = 1{,}1V_s$.

$$V_s = \sqrt{\frac{2W}{\rho S C_{L_{max}}}} \tag{2.17}$$

Se va a proceder a explicar las ecuaciones de cada uno de los tres segmentos que hay, sabiendo que:

- El peso se va a suponer constante durante los tres segmentos.
- La contribución del empuje es siempre paralela a la dirección de la velocidad.

2.7.1 Recorrido en tierra

Para analizar el recorrido en tierra del avión se va a considerar un modelo simplificado definido por las siguientes características:

- La pista es horizontal, por lo que se tiene que en todo el recorrido en tierra $\gamma = 0$.
- Un sólo segmento de aceleración, en el que se acelera desde $V = 0$ hasta $V = V_{LOF}$ con todas las ruedas en el suelo.
- Al ser $\gamma = 0$, provoca que el ángulo de ataque que ven las alas (α) sea nulo, por lo que de la ecuación 2.4 y de la ecuación 2.5, se obtienen que los coeficientes aerodinámicos son constantes.
- El empuje es independiente de la velocidad.

Las ecuaciones del movimiento son:

$$\begin{aligned} \frac{dx}{dt} &= V \\ m\frac{dV}{dt} &= T - D - \mu_r(N_1 + N_2) \\ L + N_1 + N_2 &= W \end{aligned} \tag{2.18}$$

2.7.2 Recorrido en el aire

Además de las hipótesis anteriores, se añade la suposición de que la densidad es constante durante el ascenso. Las ecuaciones que definen el movimiento provienen de las ecuaciones 2.9.

$$\begin{aligned} \frac{dx}{dt} &= V cos\gamma \\ \frac{dh}{dt} &= V sin\gamma \\ m\frac{dV}{dt} &= T - D - W sin\gamma \\ mV\frac{d\gamma}{dt} &= L - W cos\gamma \end{aligned} \tag{2.19}$$

El recorrido en el aire está formado por dos partes como se aprecia en la figura 2.5.

La primera de las partes es una transición curvilínea, en la cual se consideran las siguientes hipótesis:

- La velocidad es constante durante este segmento y vale V_{LOF}.
- La velocidad angular de giro en ascenso ($\dot{\gamma}$) es constante y vale 2,5 deg/s, con lo que el radio de giro es:

$$R = \frac{V}{\dot{\gamma}} \tag{2.20}$$

por lo que se tiene una trayectoria circular.

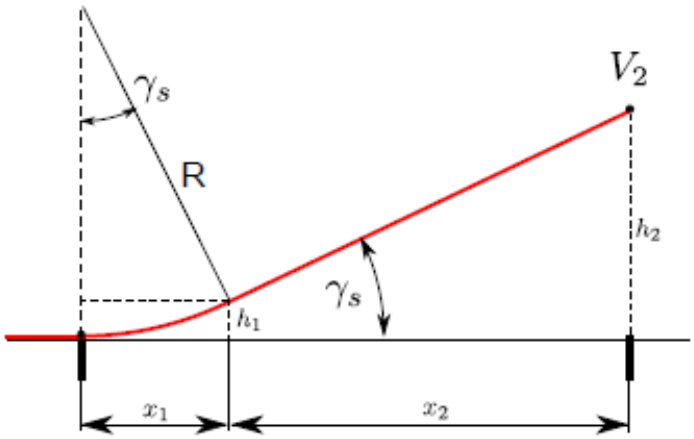

Figura 2.5 Segmentos del recorrido en el aire en despegue.

En este segmento, γ varía entre cero y γ_s que es el ángulo de ascenso del avión, que viene definido por diseño. Teniendo ésto se puede obtener la distancia horizontal recorrida x_1, la altura h_1 y el tiempo empleado t_1, aplicando estas ecuaciones:

$$x_1 = Rsin\gamma_s \tag{2.21}$$

$$h_1 = R(1 - cos\gamma_s) \tag{2.22}$$

$$t_1 = \frac{\gamma_s}{\dot{\gamma}} \tag{2.23}$$

El siguiente segmento, subida rectilínea, se realiza a empuje constante y ángulo de ascenso constante (γ_s). La altura varía entre h_1 y $h_2 = 35$ pies y la velocidad varía entre V_{LOF} y V_2.

Para resolver este segmento es necesario resolver las ecuaciones 2.19.

2.8 Ecuaciones de aterrizaje

La maniobra de aterrizaje es similar al despegue pero invirtiendo los segmentos. Se realiza con empuje nulo o incluso negativo si se dispone de reversa, en el caso del trabajo, se ha supuesto el *Aerosonde UAV* y no tiene reversa, por lo que se considerará empuje nulo durante el aterrizaje. Se compone de varias fases como se aprecia en la figura 2.6.

- Recorrido en el aire: desde la velocidad y altura del avión hasta que entra en contacto con el suelo, suponiendo que la velocidad y altura del avión no son desproporcionadas. Esta fase se compone de dos partes:
  - Aproximación rectilínea: es una trayectoria rectilínea, siguiendo la senda de planeo. Acaba a una velocidad de $1{,}15V_s$.
  - Redondeo: trayectoria de transición curvilínea, que termina cuando el avión entra en contacto con la pista.
- Rodadura en suelo: desde que el avión toca suelo hasta que se para.

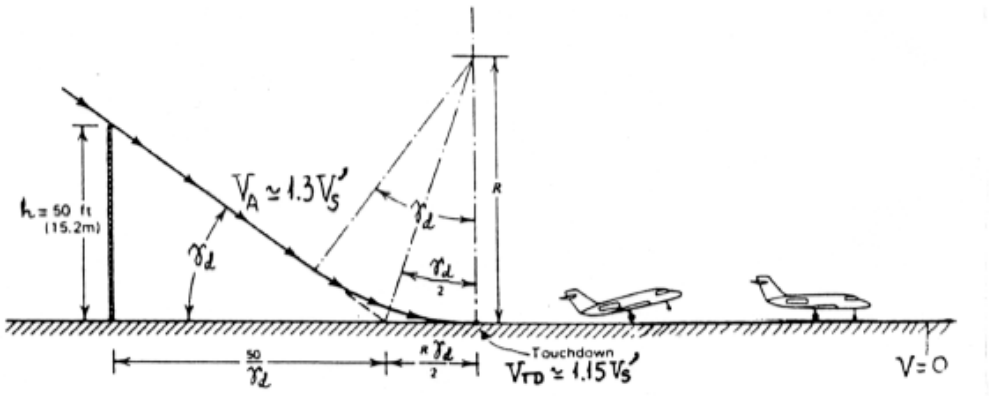

Figura 2.6 Fases en la maniobra de aterrizaje.

El desarrollo que continúa a partir de aquí se ha realizado para un peso constante durante todo el aterrizaje y un empuje nulo.

2.8.1 Recorrido en el aire

Se considera que la densidad durante este tramo es contante.

Las ecuaciones varían ligeramente respecto al despegue (ecuaciones 2.19) y a las ecuaciones 2.9, quedando finalmente como:

$$\begin{aligned} \frac{dx}{dt} &= V cos\gamma_d \\ \frac{dh}{dt} &= -V sin\gamma_d \\ m\frac{dV}{dt} &= -D + W sin\gamma_d \\ mV\frac{d\gamma_d}{dt} &= -L + W cos\gamma_d \end{aligned} \tag{2.24}$$

siendo $\gamma_d > 0$ el ángulo de descenso.

Esta fase de la maniobra se compone de dos partes, una aproximación rectilínea y un redondeo. En la aproximación rectilínea se desacelera hasta una velocidad igual a $1{,}15V_s$, a un ángulo de descenso constante. Para resolver este segmento es necesario resolver las ecuaciones 2.24.

Para el redondeo se va a considerar que la velocidad es constante y que el radio de giro es constante. En este segmento γ varía desde $-\gamma_d$ hasta cero. Además, es imprescindible que la altura y γ al final del segmento sean nulos.

La distancia recorrida viene dada por:

$$x_4 = R sin\gamma_d \tag{2.25}$$

siendo,

$$R = \frac{h_4}{1 - cos\gamma_d} \tag{2.26}$$

La velocidad de giro es constante de valor:

$$\dot{\gamma} = \frac{1{,}15V_s}{R} \tag{2.27}$$

por lo que el tiempo del redondeo viene dado por:

$$t_4 = \frac{R\gamma_d}{1{,}15V_s} \tag{2.28}$$

2.8.2 Recorrido en tierra

Para este segmento se pueden aplicar las mismas hipótesis que en la rodadura de despegue, exceptuando que la velocidad ahora se disminuye hasta ser cero y que el empuje también es nulo.

Para resolver este segmento, se aplican las ecuaciones 2.18, con empuje igual a cero, quedando como ecuaciones a resolver:

$$\begin{aligned} \frac{dx}{dt} &= V \\ m\frac{dV}{dt} &= -D - \mu_f(N_1 + N_2) \\ L + N_1 + N_2 &= W \end{aligned} \tag{2.29}$$

3 Programación orientada a objetos

El Trabajo Fin de Máster que aquí se presenta se ha desarrollado íntegramente en MATLAB® y se ha programado usando la *filosofía* de la programación orientada a objetos [9]. En este capítulo se va a explicar detalladamente en que consiste esta filosofía y cómo se ha implementado en el trabajo.

Para explicar en qué consiste la programación orientada a objetos hay que cambiar por completo la forma en que se programa normalmente porque cuando se refiere a ésta nueva filosofía no se está hablando de unas nuevas características que se han añadido a la programación, sino que se trata más bien, de una nueva forma de pensar.

Normalmente, si uno se enfrenta a la programación de un problema complejo y largo, se suele usar una programación estructurada que consiste en descomponer el problema en subproblemas cada vez más pequeños hasta que sean acciones muy simples y fáciles para programarse. En el entorno MATLAB® se utilizan las funciones (*function*) y el comando *struct*, que crea estructuras para el fin explicado anteriormente.

La programación orientada a objetos es otra forma de descomponer el problema, en concreto, se basa en la descomposición de objetos. La diferencia entre los dos modelos presentados es que el primero se descomponen las diferentes acciones a realizar hasta tener acciones muy simples a programar, mientras que este segundo método se basa en descomponer los objetos que hay en el problema, es decir, en intentar simular el escenario en nuestro programa.

Para programar esta filosofía se utilizan las clases y los objetos, que se explican a continuación, decir, que hay varias formas distintas de programación y que en este documento se explica una de ellas, que ha sido la usada para desarrollar el trabajo.

3.1 Clase

Una clase es una plantilla que define las variables y los métodos que son comunes para todos los objetos de un cierto tipo.

Por ejemplo, avión es una clase que tendrá asociado una serie de variables (color, envergadura, número de motores, etc.) y métodos (acelerar, frenar, descender, etc.) comunes a todos los objetos de la clase. Sin embargo, el estado particular que tiene cada avión que hay en el mundo es diferente entre sí, es decir, un avión puede ser de color blanco con dos motores, pero hay otro avión que tiene un solo motor y además es de color azul. De

éste modo se consigue definir una plantilla de variables y métodos para todos los aviones (figura 3.1). A ésto se le llama crear una clase llamada avión.

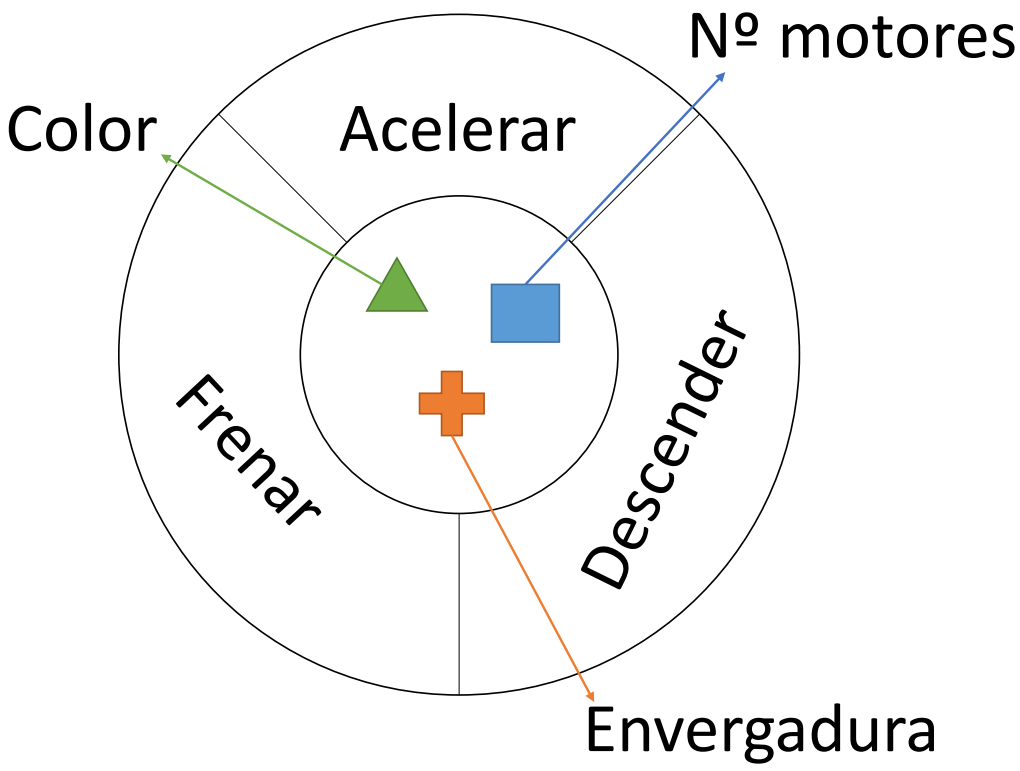

Figura 3.1 Clase.

3.2 Objeto

Se define objeto como un conjunto de variables y métodos relacionados entre sí. Por tanto, una vez definida una clase concreta, se puede crear un objeto específico de dicha clase.

Continuando el ejemplo anterior, se puede crear el objeto *Mi avión* (figura 3.2), añadiendo a cada variable un valor concreto, es decir, a color se le define el verde, a la variable envergadura se le aplica quince metros y a la variable número de motores se le aplica dos motores. Con ésto ya se tiene un objeto de la clase avión que se compone de una serie de variables y de una serie de métodos comunes a todos los aviones.

La utilidad de esta definición de clase y objeto, reside en que una vez definido ésto en cualquier programa para acelerar un objeto de la clase avión, basta con llamar al método acelerar del objeto en cuestión, lo que en un programa complejo ayuda bastante en cuanto a sencillez de escritura del programa.

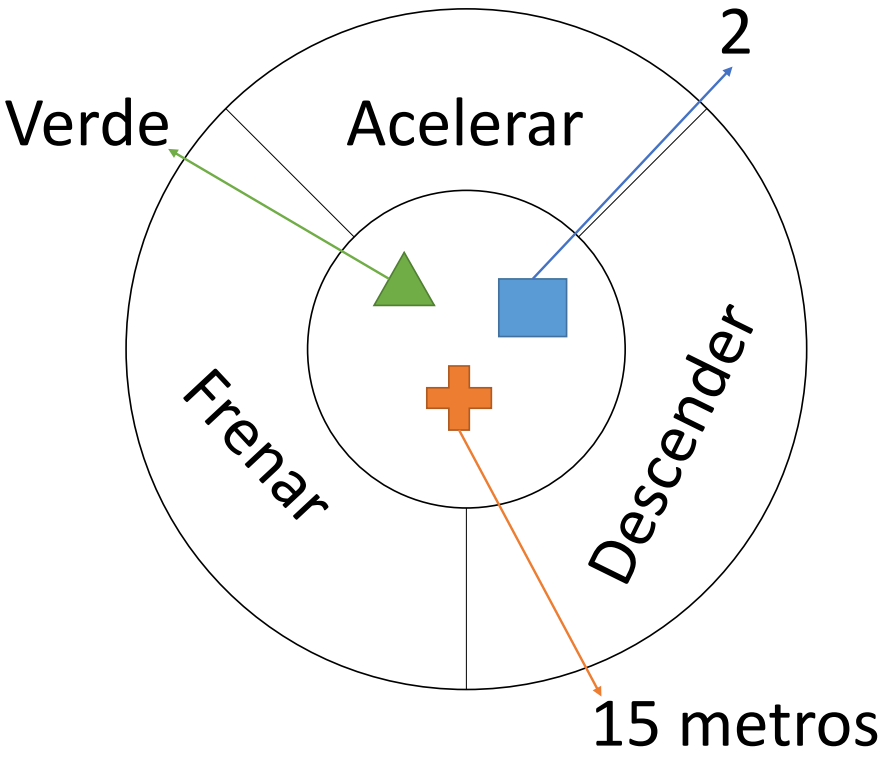

Figura 3.2 Objeto.

3.3 Definir en MATLAB® clases y objetos

Una vez explicado qué es una clase y un objeto en el ámbito de la programación orientada a objetos, hay que explicar como crear todo esto en el entorno de MATLAB® . Para ello se va a proceder a explicarlo utilizando el ejemplo anterior pero añadiendo una nueva variable a la clase que será la velocidad del avión.

Primero se va a explicar como crear una clase como la explicada anteriormente:

- En el directorio de trabajo se crea una carpeta con el nombre *@Nombre_de_la_clase*, en nuestro caso *@Avion*.
- Dentro de la carpeta, se crea un archivo con el nombre de la clase (*Avion.m*). Dicho archivo tiene en su interior el siguiente código, que es el que permite crear objetos de la clase *Avion*.

Código 3.1 Definir la clase Avion.

```
function Avion = Avion(col,mot,env,vel)

Avion.color = col;
Avion.motor = mot;
Avion.envergadura = env;
```

```
Avion.velocidad = vel;

Avion = class(Avion,'Avion');

end
```

Con esto ya se tendría la clase *Avion* creada en MATLAB® pero sólo cuando se trabaje en el directorio donde está la carpeta creada.

Si se quisiera crear un objeto de la clase creada, basta con ejecutar el código 3.2, en dónde *Avion1* es el nombre del objeto creado, y las variables introducidas a la función son el color, el número de motores, la envergadura y la velocidad respectivamente.

Código 3.2 Definir un Objeto de la clase Avion.

```
Avion1 = Avion('verde',2,15,20);
```

Si se ejecuta este comando en la ventana de comando de MATLAB® , se obtiene en el *Workspace* una variable llamada *Avion1* de la clase *Avion*, pero si intentamos mostrar esa variable en la ventana de comando aparacerá *Avion object: 1-by-1*, indicando que la variable a mostrar es una variable 1x1 de la clase *Avion*.

Para mostrar esta variable como cualquier otra variable en la ventana de comando es preciso detallar a MATLAB® qué y cómo tiene que mostrar, de un objeto de la clase *Avion*, las variables, para ello es necesario crear el archivo *display.m* en la carpeta de la clase *Avion* con el código 3.3.

Código 3.3 Función display de la clase Avion.

```
function display(Avion)

disp(' ');
disp([inputname(1),' = ']);
disp(' ');
fprintf('\tEl color del avión es %s.\n',Avion.color);
fprintf('\tEl avión tiene %1.0f motores.\n',Avion.motor);
fprintf('\tLa envergaura del avión es de %3.0f metros.\n',Avion.
   envergadura);
fprintf('\tLa velocidad del avión es %3.1f m/s.\n',Avion.velocidad)
   ;
disp(' ');

end
```

Este código escrito en una función que **obligatoriamente** se tiene que llamar *display.m* y estar en la carpeta de la clase, produce que cada vez que se ponga un objeto en la ventana de comando de MATLAB® devuelve como resultado el código 3.4.

Código 3.4 Resultado en la ventana de comando.

```
>> Avion1

Avion1 =

  El color del avión es verde.
  El avión tiene 2 motores.
  La envergaura del avión es de 15 metros.
  La velocidad del avión es 20.0 m/s.

>>
```

Ya se ha explicado como crear una clase y como crear objetos de dicha clase. También se ha modificado cómo MATLAB® va a mostrar los objetos de dicha clase, pero todavía no se ha visto para qué sirve realmente esta nueva filosofía de programación orientada a objetos, que es lo que se va a explicar a continuación.

Siguiendo con el mismo ejemplo que se ha expuesto hasta ahora, se va a proceder a introducir un método a la clase *Avion*, por ejemplo, se va a pedir que un objeto llamado *Avion1*, acelere un metro por segundo, para ello se utiliza el código 3.5 que se ha de encontrar en la carpeta de la clase.

Código 3.5 Función que acelera el avión.

```
function Avion = Acelerar(Avion)

Avion.velocidad = Avion.velocidad + 1;

end
```

Igualmente se puede crear una función similar que produzca el resultado inverso, es decir, frenar el avión (código 3.6).

Código 3.6 Función que frena el avión.

```
function Avion = Acelerar(Avion)

Avion.velocidad = Avion.velocidad - 1;

end
```

Ya desde la ventana de comando, o desde un programa, se puede llamar a la función *Acelerar* o *frenar* y ésta producirá en la ventana de comando (si se ejecuta desde ésta) el siguiente resultado.

Código 3.7 Resultado de ejecutar la función Acelerar.

```
>> Avion1 = Acelerar(Avion1)

Avion1 =
```

```
  El color del avión es verde.
  El avión tiene 2 motores.
  La envergaura del avión es de 15 metros.
  La velocidad del avión es 21.0 m/s.

>>
```

Se aprecia claramente como ha aumentado la velocidad del objeto *Avion1* en un metro por segundo.

Si se implementa éste método como se ha hecho en este proyecto, se consigue una disminución muy importante de código MATLAB® que se tiene que escribir y también facilita el proceso de escribir, evitando errores de escritura, puesto que se escribe código a más *alto nivel* ya que el código de *bajo nivel* sólo se escribe a cada clase una vez y cada objeto ya tiene implementado todos los métodos de su clase.

4 Conjunto de UAVs

En este primer programa realizado en MATLAB® se van a estudiar las características que tiene un conjunto de UAVs al barrer un área rectangular, para ello, es necesario explicar el modelo seguido para el conjunto de UAVs y las diferentes propiedades que se le han incluido al programa respecto a la programación a objetos explicada en el capítulo 3.

Este programa se basa en el barrido de un área rectangular por parte de un conjunto de UAVs con un número ilimitado de UAV que se podrá definir por parte del usuario. La idea es que todas las propiedades del programa las pueda modificar el usuario a partir de una interfaz gráfica diseñada para el caso.

Destacar la posibilidad que tiene el programa de que un UAV del conjunto pueda ser desactivado mientras se esté desarrollando el barrido del área por lo que es necesario que se reorganicen el resto de UAVs del conjunto para barrer cada uno su área y además el área que tenía que barrer el UAV desactivado y no le había dado tiempo.

4.1 Modelo

Se va a explicar qué características tiene el modelo del programa expuesto en este capítulo. Primero, se van a explicar las distintas hipótesis utilizadas para la realización del programa:

- El mapa donde se desarrolla la simulación se supone plano.
- Los UAVs se inician a una altura, posición horizontal y velocidad prefijados.
- La posición vertical inicial de los UAVs vendrá dada por el mapa.
- La velocidad lineal en los giros de los UAVs será constante.
- Cada UAV sigue sus propios waypoints, por lo que no tiene en cuenta a los otros UAVs. Para que no haya problemas, se deberá poner a cada UAV a una altura diferente y con un margen de seguridad coherente para que no haya interferencias entre ellos.

A continuación se va a desarrollar las características de los UAVs, el algoritmos de movimiento de los UAVs del conjunto y cómo sucede la desactivación del UAV.

4.1.1 Características de los UAVs

Los UAVs vienen definidos por las siguientes propiedades:

- Posición horizontal a la que empieza inicialmente el UAV.

- La altura del UAV.
- La velocidad del UAV.
- El ángulo de trayectoria inicial del UAV.
- El ángulo de apertura del sensor que tiene la cámara que lleva el UAV.
- El radio mínimo de giro del UAV.

Con estos parámetros se tiene definido perfectamente cada uno de los UAVs del conjunto.

4.1.2 Movimiento del UAV

El movimiento de cada UAV viene determinado por waypoints, de tal forma que lo primero que hay que hacer es darle a cada UAV una serie de waypoints que son los que va a seguir en orden. De tal forma, que si cada UAV sigue los waypoints asignados se barre por completo el área.

El movimiento de los UAV va a ser descendente en el eje vertical y barriendo en horizontal el área requerida (figura 4.1), porque es el algoritmo más rápido para barrer un área rectangular de manera óptima [16].

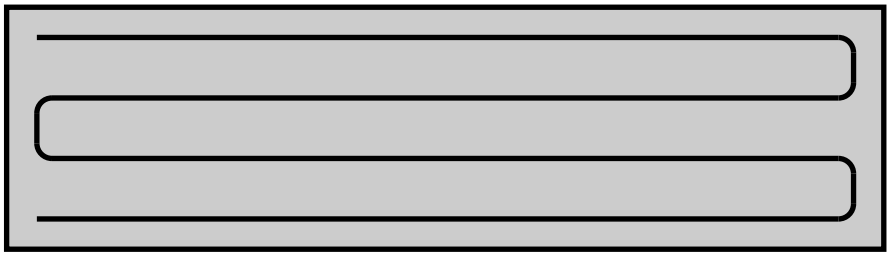

Figura 4.1 Algoritmo para barrer el área rectangular.

Para que este movimiento se haga correctamente, se permite modificar dos parámetros:

- La distancia a la que tiene que estar el UAV de un waypoint para que cambie de waypoint al siguiente.
- El número de waypoint que hay en cada fila de barrido.

4.1.3 Movimiento del conjunto de UAVs

Una vez que se ha visto como es el movimiento de un UAV, aquí se va a estudiar como se va a mover el conjunto, para ello, se va a intentar que dicho conjunto se mueva de forma óptima. Para el caso de un área rectangular y suponiendo que la velocidad de los UAVs es similar, la forma óptima para barrer el área es con todos los UAVs del conjunto en paralelo y barriendo igual que si fuera un único UAV (figura 4.1).

4.1.4 Desactivación del UAV

Para el caso en que la desactivación de un UAV del conjunto esté activa es necesario dar como parámetro qué UAV concreto es el que se va a desactivar y en que tiempo de la simulación va a tener lugar su desactivación. Estos dos parámetros van a tener dos posibles soluciones:

- Que el UAV desactivado haya acabado de barrer el área que tenía asignada.
- Que el UAV desactivado **no** haya acabado de barrer el área que tenía asignada.

En el primer caso, no hace falta modificar nada porque todo el área va a ser barrida por el conjunto de UAVs, sin embargo, en el segundo caso si hace falta modificar los waypoints de cada UAV para que se pueda barrer todo el área.

Se va a exponer el algoritmo realizado a la hora de reorganizar los waypoints cuando un UAV ha sido desactivado, para ello, se supone que la velocidad de los UAVs es la misma, puesto que si no lo fuera, se añadiría mucha más incertidumbre al problema porque no se sabría dónde se encuentra cada UAV cuando uno ha sido desactivado dando lugar a situaciones bastante caóticas en cuanto al movimiento de los UAVs.

- Se obtiene hasta qué waypoint ha barrido el UAV desactivado.
- Se reorganizan los waypoints de la siguiente manera.
  - Se mantienen los waypoints que han sido vistos por algún UAV.
  - Se mantienen también los waypoints de la fila que están barriendo en el momento de la desactivación.
  - Una vez hecho esto, queda un área rectangular más pequeña que la inicial y un área rectangular que ocupa desde el último waypoint visto por el UAV desactivado hasta el final de la fila donde se encuentra ese waypoint. Visto ésto, se asigna el área no vista del waypoint desactivado al UAV que vuela a mayor altura[1], para asegurarse que el área que barre el UAV es mayor o igual que el área que tiene que barrer y el área restante del rectángulo inicial se reparte volviendo a aplicar la sección 4.1.3 a los UAVs activos.
- En algunos casos puntuales (figura 4.3), se le da un waypoint de apoyo al UAV que tiene que barrer el área del UAV desactivado para que sea más fácil el barrido de ese área. Dicho waypoint de apoyo está a 200 metros de distancia del primer UAV al que se tiene que dirigir y en el mismo eje vertical.

En las figuras 4.2 y 4.3 se ha tomado una fotografía de como quedarían los waypoints del área justo antes y justo después de la desactivación de un UAV. En este caso, se ha desactivado el UAV azul, por tanto, el UAV verde, que es el de mayor altura, es el encargado de barrer el área que no ha barrido el UAV azul, siendo ayudado por un waypoint que se le da para su ayuda. También se aprecia cómo todo el área restante cambia, desapareciendo los waypoints del UAV azul y repartiéndose el resto de UAVs el área no barrida.

4.2 Modelado en MATLAB®

Una vez explicado que hace este programa, se va a estudiar como se ha programado esto en el entorno MATLAB® . Para ello, se van a detallar cada una de las funciones utilizadas en este programa.

[1] Si el UAV desactivado fuera el UAV de mayor altura de todos los UAVs del conjunto, se tomaría el UAV con la segunda mayor altura y se aumentaría su altura a la del UAV desactivado y se continuaría con el proceso.

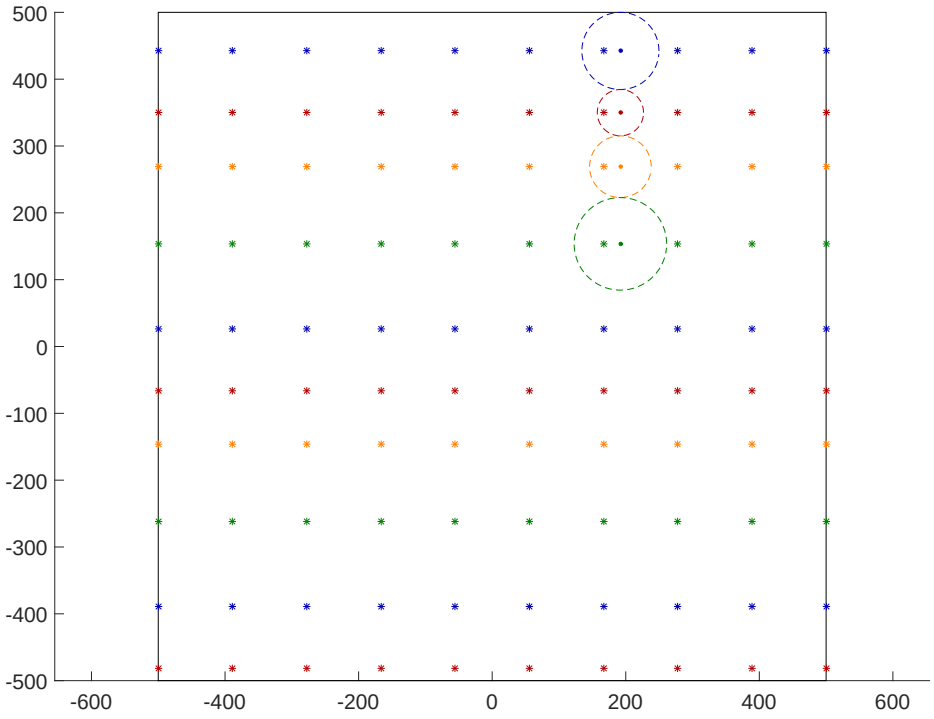

Figura 4.2 Desactivación al inicio.

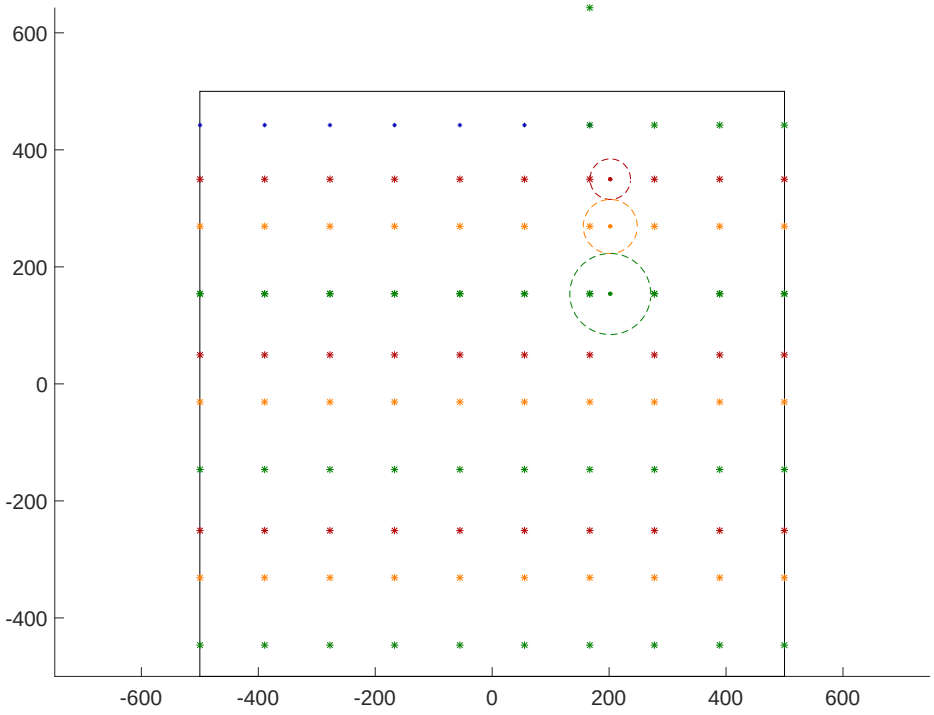

Figura 4.3 Desactivación al final.

Lo primero es decir, que hay dos interfaces gráficas que se estudiarán posteriormente, aquí se van a explicar los archivos .m que no tienen que ver con las interfaces.

Explicado el capítulo 3 de programación a objetos, se va a ver como ha influido dicha filosofía de programación en este programa. Pues bien, se ha creado una clase que se llama *UAV* y que tiene las siguientes características:

- x: posición x del UAV.
- y: posición y del UAV.
- h: altura del UAV.
- v: velocidad del UAV.
- θ: ángulo de trayectoria del UAV, medido desde el eje horizontal. Dicho ángulo está en radianes y siempre tiene un valor entre 0 y 2π.
- R_{min}: radio mínimo de giro del UAV.
- fov: apertura del sensor de la cámara en radianes.
- act: variable que indica si el UAV está en vuelo o si no está activo, tomando el valor 1 cuando está activo y el valor 0 cuando no lo está.

Dentro de la carpeta donde se encuentra esta clase hay también varios archivos de MATLAB® que sirven para diferentes tareas. A continuación se van a detallar algunos de los archivos que hay dentro de la carpeta y que son métodos de la clase *UAV*.

Código 4.1 Función display de la clase UAV.

```
function display(UAV) %#ok.

% Función que muestra el objeto de la clase UAV en la ventana de
    comandos.

disp(' ');
disp([inputname(1),' = ']);
disp(' ');
fprintf('\tPosición x: %5.2f m.\n',UAV.x(end));
fprintf('\tPosición y: %5.2f m.\n',UAV.y(end));
fprintf('\tAltura: %5.2f m.\n',UAV.h(end));
fprintf('\tVelocidad del UAV: %3.2f m/s.\n',UAV.v(end));
fprintf('\tÁngulo de la velocidad: %3.2fº.\n',rad2deg(UAV.theta(end
    )));
fprintf('\tRadio mínimo de giro: %3.2f m.\n',UAV.Rmin);
fprintf('\tApertura del sensor: %3.2fº.\n',rad2deg(UAV.fov));
if UAV.act == 1,
fprintf('\tEstá activo.\n');
else
fprintf('\tNo está activo.\n');
end
disp(' ');
```

```
end
```

Código 4.2 Función Datos de la clase UAV.

```
function varargout = Datos(UAV,varargin)

% Función que devuelve los datos introducidos en varargin del UAV
    seleccionado.

if nargin > 1,
varargout = cell(1,length(varargin));
for k=1:length(varargin),
varargout{k} = UAV.(varargin{k});
end
end

end
```

Código 4.3 Función Guardar de la clase UAV.

```
function UAV = Guardar(UAV,varargin)

% Función que guarda los datos introducidos en la variable varargin
     en el UAV seleccionado.

if nargin > 1,
if mod(length(varargin),2) ~= 0,
error('Error en el input de los datos. VARARGIN.');
end
for k=1:length(varargin)/2,
if strcmp(varargin{2*k-1},'theta'),
if varargin{2*k} >= 2*pi,
varargin{2*k} = varargin{2*k} - 2*pi;
elseif varargin{2*k} < 0,
varargin{2*k} = varargin{2*k} + 2*pi;
end
end
UAV.(varargin{2*k-1}) = varargin{2*k};
end
end

end
```

Estas tres funciones de MATLAB® , sirven para cosas completamente diferentes que se van a detallar:

- *display*: sirve para mostrar por pantalla los datos de un objeto de la clase *UAV* como ya se explicó en el capítulo 3.
- *Datos*: función que sirve para coger datos de un objeto de la clase *UAV*. Por ejemplo, si se quisiera obtener la velocidad de un objeto de dicha clase, no habría más que poner el código 4.4 en el programa.
- *Guardar*: función que sirve para guardar datos en un objeto. Por ejemplo, en el código 4.5 hay descrito cómo se llamaría a la función para guardar en el objeto *UAV1* la variable *Velocidad*.

Código 4.4 Coger la velocidad de un objeto.

```
% Siendo UAV1 el objeto de la clase UAV.

Velocidad = Datos(UAV1,'v');
```

Código 4.5 Guardar la velocidad de un objeto.

```
% Siendo UAV1 el objeto de la clase UAV.

UAV1 = Datos(UAV1,'v',Velocidad);
```

Hay más métodos dentro de la clase *UAV* que sirven para el movimiento de un objeto de la clase *UAV*, en concreto, hay un método más que consiste en una especie de control muy simple para que el objeto (UAV en vuelo) sea capaz de seguir los waypoints. El código 4.6 es el citado código y no hace falta explicación porque es muy fácil de entender, quizás, lo más interesante del código es cómo utiliza las propiedades del objeto *UAV* que se aprecia en la línea que pone $R = UAV.Rmin$, es decir, cuando se está ejecutando un método de la clase no es necesario coger los datos del objeto, sino que basta con poner el objeto.propiedad para obtenerla.

Código 4.6 Ángulo de trayectoria nuevo del objeto.

```
function theta = Theta_new(UAV,Wp,D,paso)

% Función que devuelve el nuevo ángulo de la velocidad que debe
    llevar el UAV para dirigirse al waypoint.

theta_wp = atan2(Wp(2)-UAV.y,Wp(1)-UAV.x);

if theta_wp < 0,
theta_wp = theta_wp + 2*pi;
end

R = UAV.Rmin;
if Dist_waypoint(UAV.x,UAV.y,Wp(1),Wp(2)) >= 25 && abs(D/2) > UAV.
    Rmin,
```

```
R = abs(D/2);
end

omega = UAV.v/R;
theta_max = omega*paso;

theta_giro = theta_wp - UAV.theta;

if theta_giro > pi,
theta_giro = theta_giro - 2*pi;
elseif theta_giro < -pi,
theta_giro = theta_giro + 2*pi;
end

if theta_giro > theta_max,
theta_giro = theta_max;
elseif theta_giro < -theta_max,
theta_giro = -theta_max;
end

theta = UAV.theta + theta_giro;

end
```

El programa en sí consta de varios archivos de MATLAB® que se van ejecutando para obtener el resultado deseado, no tiene sentido poner el código entero porque ocuparía demasiado espacio y no es el propósito del trabajo, pero sí explicar el algoritmo en general de cómo se ha desarrollado este programa.

El código principal es un archivo que se llama *TFM.m* que lo primero que hace es ejecutar llamar a otro archivo de MATLAB® que se llama *Swarm_datos.m*. Este archivo es el encargado de generar los datos que se van a necesitar para la ejecución correcta del programa. El algoritmo 1 se corresponde con el algoritmo utilizado en dicho archivo. Conviene recordar que cuando se habla de objeto, en realidad, cada uno de los objetos es un UAV diferente.

Algoritmo 1 Archivo Swarm_datos.m

1: Se genera cada uno de los objetos de la clase *UAV*.
2: Se calcula el paso de tiempo que se va a utilizar en la simulación.
3: Se calcula los waypoints de cada uno de los objetos.

Los demás datos necesarios por la simulación, por ejemplo, el tiempo y el UAV que se va a desactivar o los datos iniciales de cada uno de los UAV se van a introducir a partir de la interfaz creada para tal caso, pero esto se explicará más adelante.

Una vez vista la función *Swarm_datos.m* volvemos a la función *TFM.m*, la cual tiene el grueso del programa de la simulación. El algoritmo de esta función es el algoritmo 2. Antes de explicar el algoritmo de esta función es necesario indicar que lo primero que hay que hacer es crear una matriz por cada UAV de ceros con tantos filas como iteraciones vaya

a haber y tres columnas (posición x, y, z), para que MATLAB® guarde los espacios en la memoria y así la rapidez del programa aumentará notablemente.

Algoritmo 2 Archivo TFM.m

```
1:  para k = 1 hasta Inf hacer
2:    para j = 1 hasta número UAVs hacer
3:      si UAV == Activo entonces
4:        Coger datos del UAV.
5:        Actualización del movimiento del UAV.
6:        Guardar los datos del UAV.
7:      fin si
8:    fin para
9:    Se comprueba si el programa ha finalizado.
10:   si Tiempo == Desactivación entonces
11:     Se activa la desactivación.
12:   fin si
13: fin para
```

Éste sería el algoritmo del programa principal del cual se van a explicar las diferentes acciones que se van haciendo.

- La actualización del movimiento de los UAVs sigue el algoritmo 3. La nueva posición viene dada por las ecuaciones 4.1, y la nueva velocidad y altura son un control muy simple para satisfacer la velocidad y la altura deseada.

$$\begin{aligned} x_{new} &= x_{old} + cos(\theta) \cdot v \cdot paso \\ y_{new} &= y_{old} + sen(\theta) \cdot v \cdot paso \end{aligned} \tag{4.1}$$

  siendo θ el nuevo ángulo de trayectoria, v la velocidad y $paso$ el paso de tiempo que tiene la simulación.

- El programa finaliza cuando todos los UAVs han visto todos sus waypoints.

- Si se activa la desactivación, se aplica el algoritmo 4.

Algoritmo 3 Actualización de los UAVs

```
1: Se selecciona el waypoints al que se tiene que dirigir.
2: Se calcula la nueva velocidad.
3: Se calcula la nueva altura.
4: Se calcula el nuevo ángulo de trayectoria.
5: Se calcula la nueva posición.
```

Con todo esto quedaría claro como se ha modelado y programado en MATLAB® el programa 1 del presente trabajo, ya sólo quedaría ver cuales son las interfaces creadas para la resolución de este programa.

Algoritmo 4 Desactivación de un UAV

1: Se obtienen los waypoints, de la misma fila que estaba barriendo, que no han sido vistos por el UAV desactivado.
2: Se calcula cual es el UAV volando a mayor altura[a].
3: Se introducen los waypoints no vistos por el UAV desactivado al UAV seleccionado de mayor altura.
4: Se vuelven a repartir los waypoints del área no visto por ningún UAV entre los UAV activos.

[a] Si el que vuela a mayor altura es el UAV desactivado, el segundo que vuela a mayor altura debe aumentar su altura hasta la del UAV desactivado.

4.3 Interfaces gráficas (GUIDE)

Una interfaz gráfica debe servir para ayudar al usuario a manejar un programa determinado de una forma sencilla y eficaz, por ello, se han decidido crear dos interfaces complementarias entre sí para aumentar la facilidad de manejar todos estos datos ya vistos en secciones anteriores.

Mencionar que ejecutar y visualizar el programa no es lo mismo, de hecho, para este programa en concreto, la manera de ejecutar la simulación es pulsar el botón *Ejecutar*, que dará como resultado un archivo llamado *Sim.mat* con los resultados de la misma. Y es entonces cuando si se pulsa el botón *Pintar*, se pinta los datos guardados en el archivo mat. De tal forma que se ejecuta la simulación una única vez pero permite visualizar la misma ejecución tantas veces como se quiera.

4.3.1 Interfaz principal

La interfaz principal (figura 4.4) consta de diferentes botones que se van a explicar a continuación:

- Botón *Menú*: se encuentra en la parte superior izquierda, y es un desplegable que desvela dos botones:
  - Botón *Datos de entrada*: abre la interfaz secundaria que sirve para modificar los datos de entrada a la simulación.
  - Botón *Salir*: cierra el programa.
- Botón *Ejecutar*: inicia la simulación, pero no la pinta.
- Texto escrito en la parte inferior del botón *Ejecutar*: sirve para saber cuanto ha tardado la simulación en tiempo de simulación, no real. Mientras se ejecuta la simulación también se va modificando el número.
- Botón *Pintar*: Pinta la simulación en la parte derecha de la interfaz.
- *Sombreado*: si se activa antes de pulsar el botón *Pintar*, por donde pase cada UAV irá dejando un rastro para saber si de verdad se ha barrido todo el área. Para este tipo de visualización se requiere un ordenador con buenas características.
- Panel de vistas: tiene tres opciones para modificar el tipo de visualización. No se puede cambiar de visualización, de tal forma que al pulsar el botón *Pintar* la vista que esté puesta será la que se muestre hasta que acabe la visualización.

En las figuras 4.4, 4.5, 4.6 y 4.7 se muestras como quedaría la interfaz antes las distintas variaciones según el tipo de visualización.

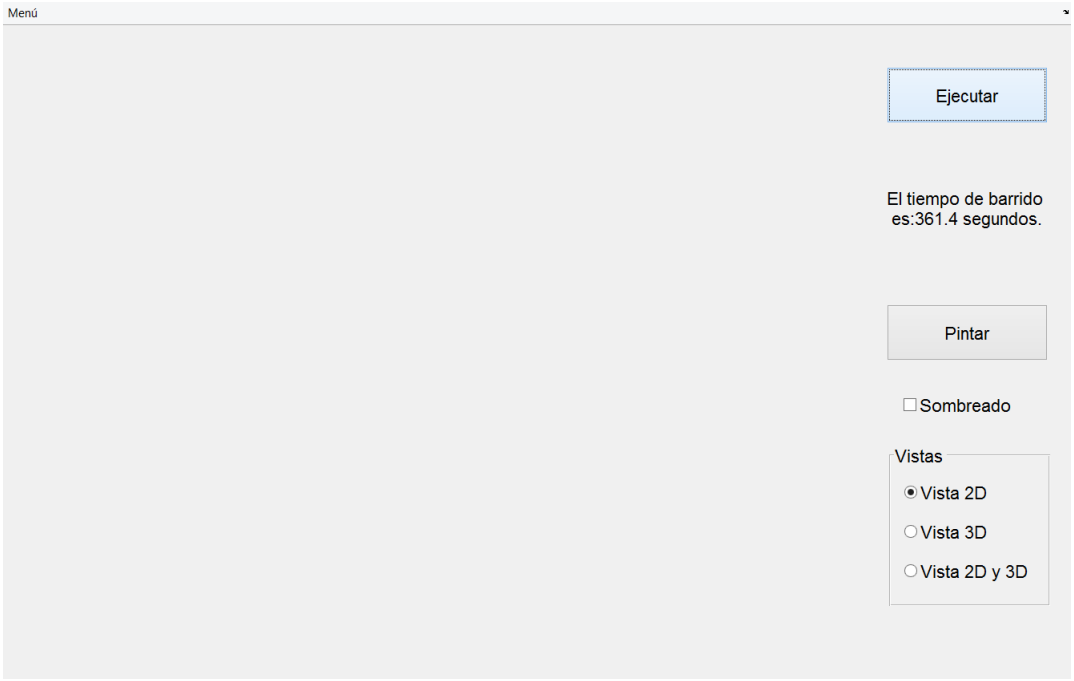

Figura 4.4 Interfaz principal.

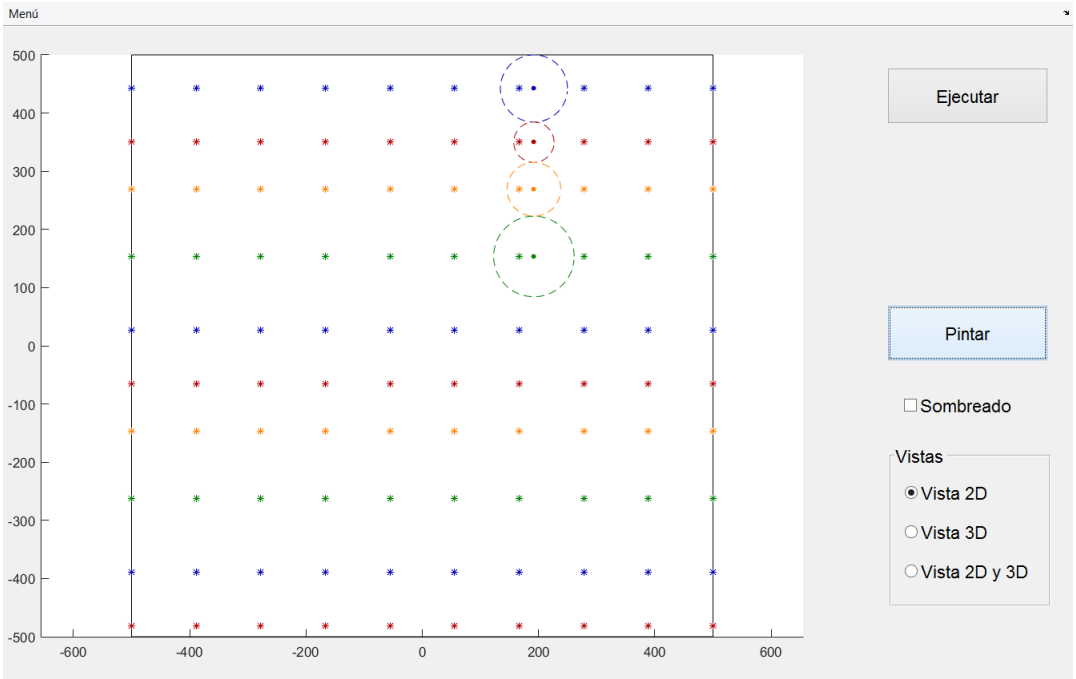

Figura 4.5 Interfaz con las vista 2D.

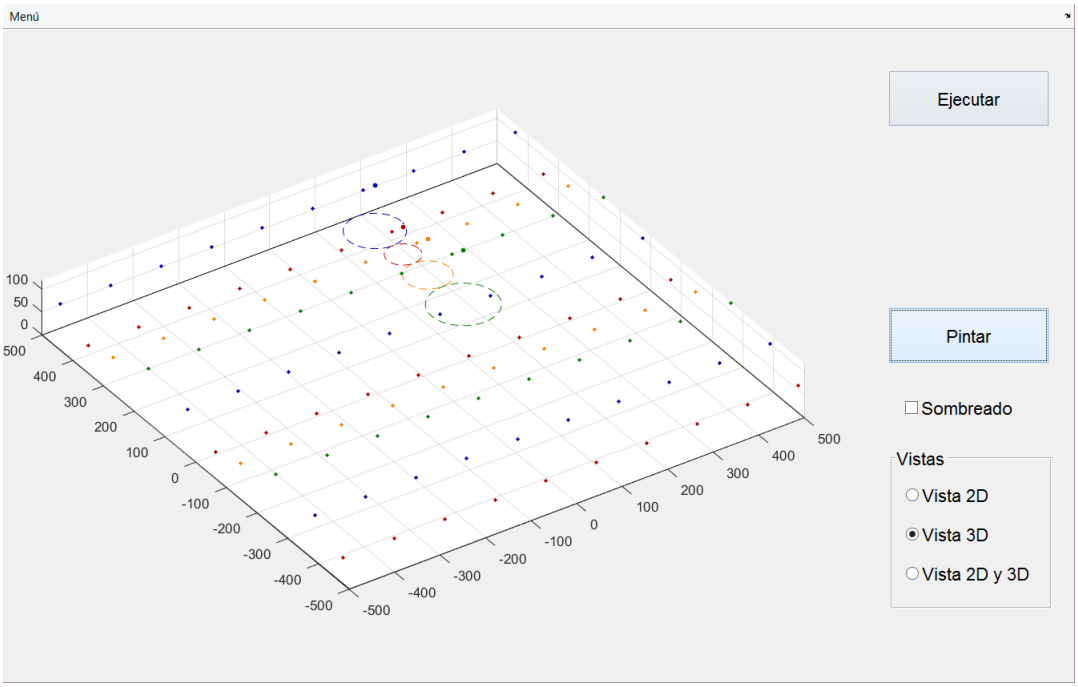

Figura 4.6 Interfaz con las vista 3D.

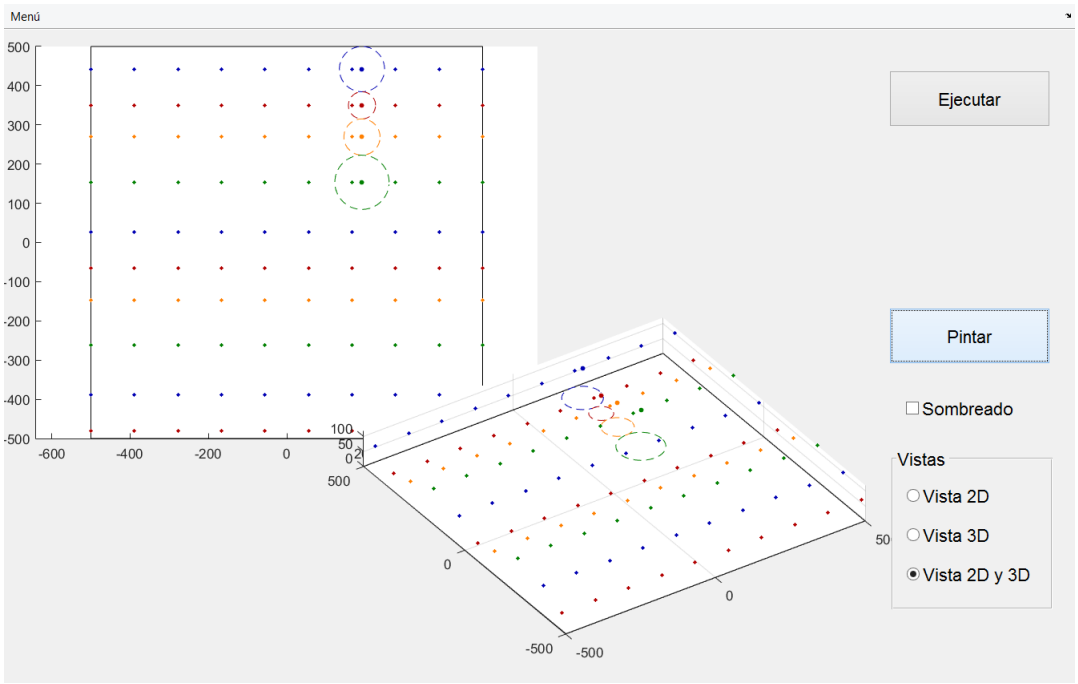

Figura 4.7 Interfaz con las vista 2D y 3D.

4.3.2 Interfaz secundaria

La interfaz secundaria (figura 4.8) sirve para manejar todos los datos de la simulación y es la encargada de facilitar esos datos al programa principal, para ello guarda todos los datos en un archivo que se llama *Datos_ini.mat* y de este archivo es de donde los lee el programa principal. Esta interfaz consta de:

- Panel del tamaño del mapa: está en la parte superior izquierda y consta de dos entradas, el tamaño según el eje horizontal, x, y el tamaño según el eje vertical, y. Decir que el mapa siempre va a estar centrado respecto el origen de coordenadas.
- Respecto a los waypoints, se pueden modificar dos valores, la distancia que tiene que haber entre el waypoints y el UAV para que el UAV cambie al siguiente waypoint, y los waypoints que se ponen en cada fila de barrido. En cuanto al primer parámetro y debido a que la simulación es de paso discreto, no conviene darle valores inferiores a 0,8. De todas formas, el valor de un metro parece razonable. Respecto a los waypoints por fila, con que hubiera dos sería suficiente, pero conviene darle más valores para que el UAV no se salga del camino previsto.
- Panel de la *desactivación*: tiene dos valores como se ha venido comentando, el UAV que se va a desactivar y el tiempo de simulación en que se va a desactivar el UAV.
- Panel de los UAVs: este panel tiene todos los valores de los UAVs creados. Contiene dos botones que son los que permiten crear nuevos UAVs, o eliminar los ya existentes. Hay que decir, que en teoría no hay un máximo de UAV, pero ese factor va a depender de la capacidad de procesamiento que tenga el ordenador con que se quiera realizar la simulación.

  Una vez que se quiera modificar los valores de un UAV, basta con modificar en la tabla el valor, cada columna significa:

  - *Orden*: indica el orden de los UAVs de arriba hacia abajo, siendo 1 el UAV superior y el último será el UAV inferior.
  - x: posición x del UAV. Conviene que dicha posición sea coherente con el ángulo de trayectoria y el tamaño, para que no haga cosas raras el UAV al inicio de la simulación.
  - h: altura inicial del UAV.
  - v: velocidad inicial del UAV.
  - $Trayectoria$: ángulo de trayectoria inicial del UAV.
  - fov: apertura del sensor que tiene la cámara que lleva montada el UAV.
  - *Radiomin.*: es el radio mínimo de giro del UAV.
- Botón *Guardar*: guarda los valores que aparecen actualmente en pantalla en el archivo *Datos_ini.mat*.
- Botón *Reset*: vuelve los valores que hay en la pantalla a los que hay guardados en el archivo *Datos_ini.mat*.
- Botón *Salir*: sale de esta interfaz sin guardar ningún dato.

El botón *Guardar* y el botón *Reset* requiere de confirmación para llevar a cabo la orden, el botón *Salir* no requiere de dicha confirmación.

4.4 Resultados del programa

Ya se ha explicado también el modelado de la interfaz gráfica, por lo que ahora se van a explicar algunas de las posibles aplicaciones que tiene este programa.

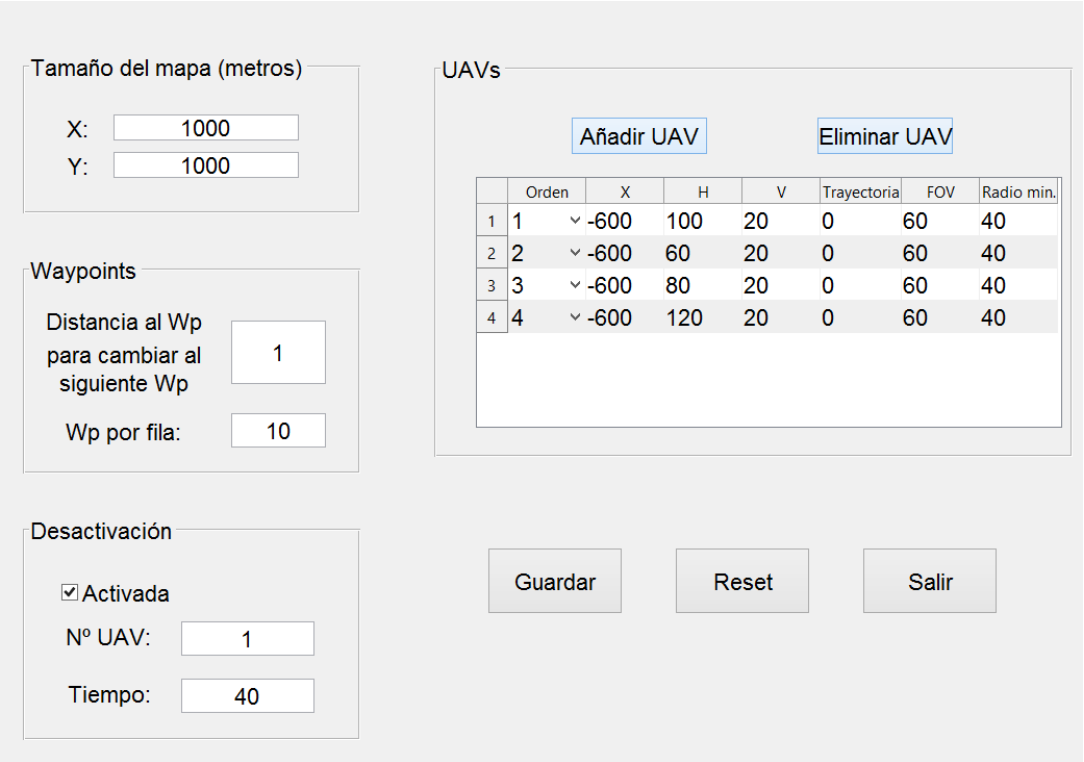

Figura 4.8 Interfaz secundaria.

Se han realizado tres tipos de simulaciones posibles y se han estudiado los resultados para ver la fiabilidad del programa. Los tres tipos de simulaciones son el número de UAVs, el área barrida y la apertura del sensor, todos estos parámetros frente al tiempo de barrido. Se va a estudiar cada simulación por separado.

4.4.1 Número de UAVs

Para esta simulación se ha supuesto que el área que se tiene que barrer es de 9 km^2, es decir, un cuadrado de 3x3 kilómetros. Las características de los UAVs son:

- La posición horizontal inicial de los UAVs se supone a 100 metros a la izquierda del inicio del cuadrado.
- La altura es la misma para todos los UAVs y vale 100 metros.
- La velocidad es de 20 m/s.
- La trayectoria inicial de los UAVs es horizontal con sentido hacia la derecha.
- La apertura del sensor es de 60°.
- El radio mínimo de giro es de 40 metros.

La figura 4.9 muestra como al aumentar el número de UAVs, el tiempo de barrido para un mismo área y sin cambiar ningún parámetro va disminuyendo lo cual es lógico. Se aprecia que para unos ciertos valores del número de UAVs el tiempo de barrido no cambia, por ejemplo, para siete y ocho o desde nueve y hasta doce UAVs, esto se debe a que al barrer horizontalmente el área y no de forma cooperativa, la forma de medir el tiempo es por pasadas, es decir, a mayor número de pasadas que haya que hacer, mayor será el tiempo

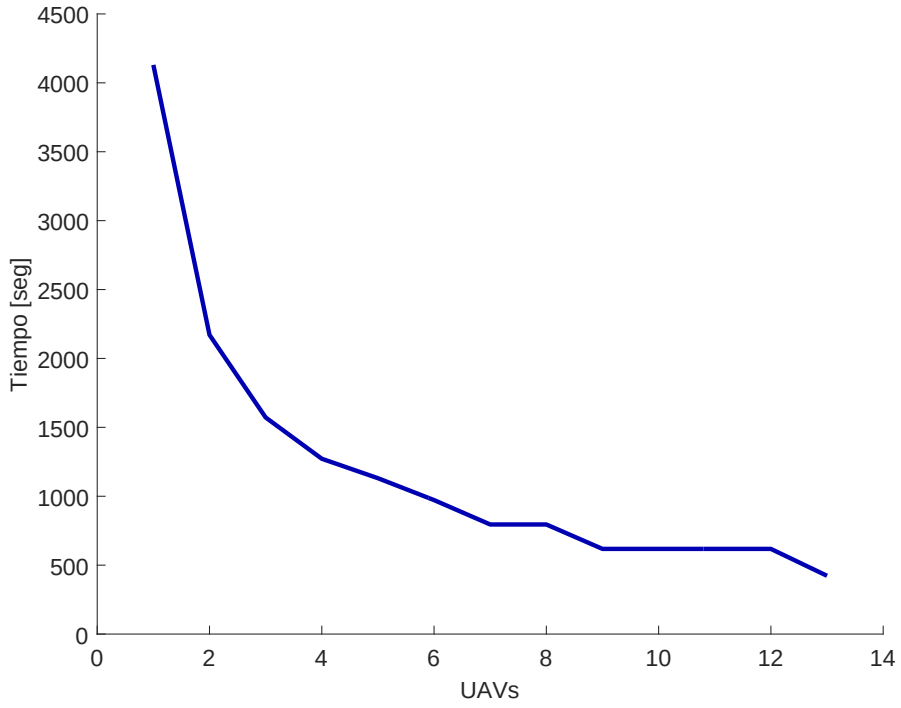

Figura 4.9 Número de UAVs frente a tiempo de barrido.

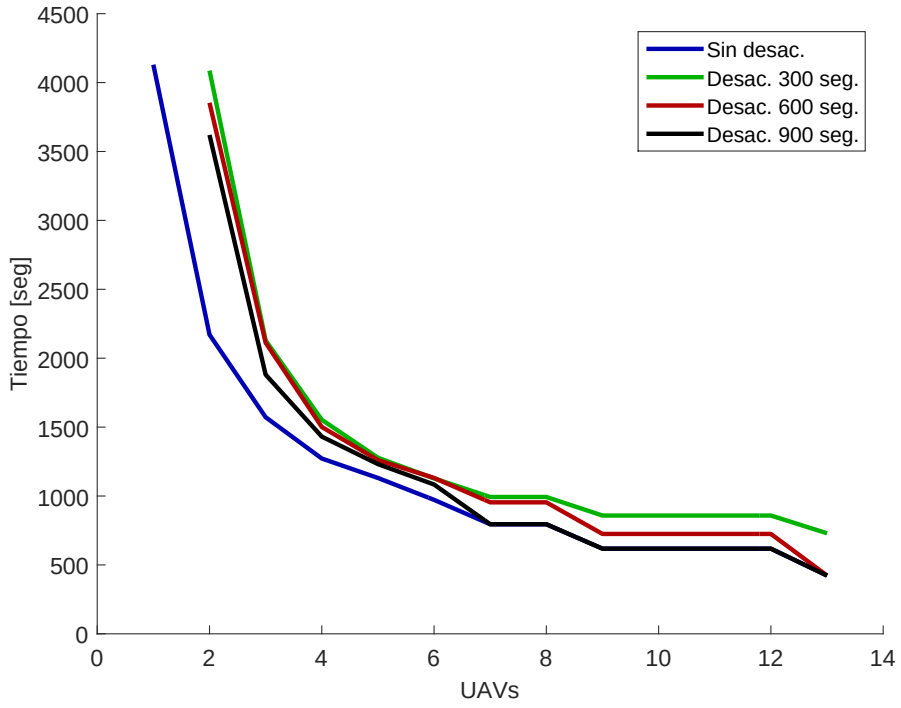

Figura 4.10 Número de UAVs frente a tiempo de barrido con desactivación.

de barrido. Si por ejemplo, con nueve UAVs se dan x pasadas, para doce UAVs se da el mismo número de pasadas por lo que el tiempo es el mismo en ambos casos.

Dado que este programa admite la eliminación de un UAV durante la simulación, se ha obtenido la figura 4.10, la cual muestra la misma simulación que la figura 4.9, pero con desactivación de UAVs. Decir que siempre se ha desactivado el mismo UAV, el que empieza a barrer inicialmente por la parte superior, y lo que cambia es el instante de tiempo en que se ha desactivado el UAV.

Se ha obtenido la figura para tres tiempos distintos, 300, 600 y 900 segundos. Se aprecian las siguientes características:

- Cuando hay un único UAV, no tiene sentido hablar de desactivación.
- Cuanto antes se desactive un UAV, mayor será el tiempo de barrido, es decir, en la figura 4.10 se ve que la línea de desactivación de 300 segundos es siempre superior a la línea de 900 segundos. Esto tiene sentido puesto que cuanto más avance la simulación mayor área se ha barrido con el plan inicial y será un menor área la que tiene que ser modificada con los nuevos UAVs.
- A partir de siete UAVs, no tiene sentido hablar de la desactivación de 900 segundos porque tardan menos de 900 segundo en barrer el área entera, por lo que la desactivación no afecta a la simulación.
- Lo mismo pasa en la simulación de trece UAVs con la desactivación programada a los 600 segundos.

A priori se puede aceptar que el programa funciona correctamente en cuanto al número de UAVs y el tiempo de barrido del área porque devuelve resultados que parecen bastantes razonables.

4.4.2 Área barrida

En esta otra simulación, se modifica el área de barrido y se ve como evoluciona el tiempo de barrido para cada área. Para esta simulación se han usado tres UAVs que son los que tienen que barrer el área, y sus características son:

- La posición horizontal inicial de los UAVs es 100 metros a la izquierda del área.
- La altura de los UAVs es 100 metros.
- La velocidad inicial es de 20 m/s.
- La trayectoria inicial de los UAVs es horizontal con sentido hacia la derecha.
- La apertura del sensor de los UAVs es 60°.
- El radio mínimo de giro de los UAVs es 40 metros.

El área que se barre en esta simulación es siempre cuadrada y únicamente varía su área.

En la figura 4.11 se aprecia que al aumentar el área a barrer aumenta el tiempo de barrido, lo cual tiene sentido. La aplicación que tiene esta gráfica es que se puede saber con antelación si los UAVs van a ser capaces de barrer un área con la autonomía que tienen, por ejemplo, si se tiene que barrer un área de 5 km^2, la autonomía de cada UAV debe ser superior a 1.000 segundos más la parte de autonomía que gastan en ir desde el despegue

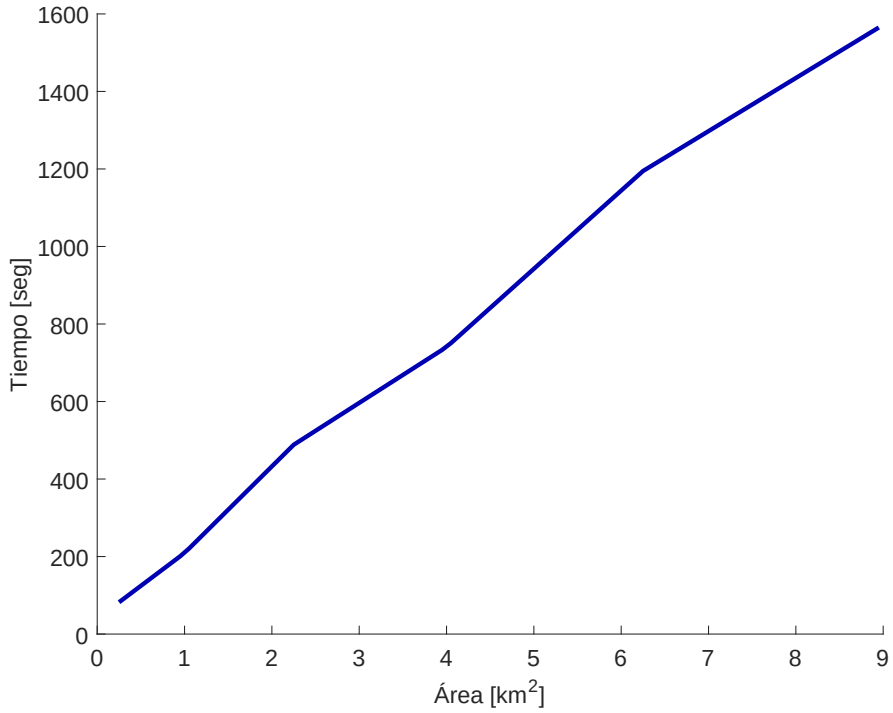

Figura 4.11 Área frente a tiempo de barrido.

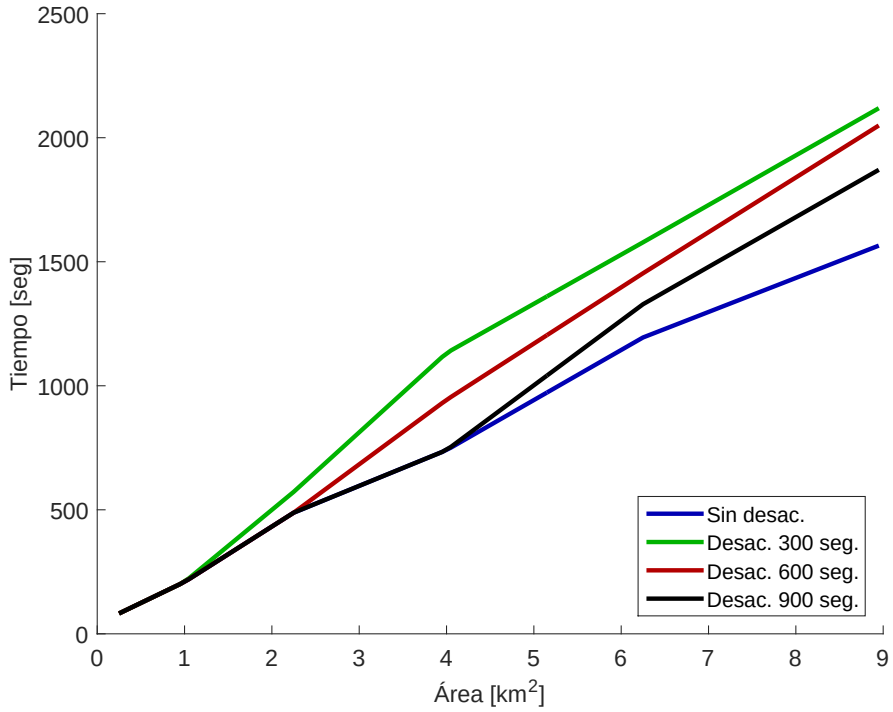

Figura 4.12 Área frente a tiempo de barrido con desactivación.

al inicio del área y la vuelta a la zona de aterrizaje. Así pues, con este programa se puede saber si es posible con los medios que se tiene barrer un área determinada.

Para el caso en que haya alguna desactivación de un UAV (figura 4.12), se ve que la gráfica es la misma pero aumenta el tiempo de barrido para un mismo área, las características de la figura son:

- Se ha simulado cada área con una desactivación del primer UAV a los 300, 600 y 900 segundos.
- Para áreas pequeñas de barrido, no hay desactivación, porque el área se barre antes de los 300 segundos y no da tiempo a alcanzar el tiempo para la desactivación del UAV.
- Cuanto antes sucede la desactivación, mayor es el tiempo de barrido. Al igual que en la simulación anterior, esto tiene sentido, porque cuanto antes se desactive mayor área será la que no ha barrido el UAV y mayor área es la que tienen que barrer los otros UAVs.

Vistas las características de estas simulaciones no hay razón para decir que el programa no funciona correctamente.

4.4.3 Apertura del sensor

Esta es la última simulación que se va a estudiar para este programa, en la cual se va a variar la apertura del sensor (FOV) y ver cómo varía el tiempo de barrido. Se va a realizar con tres UAVs con las siguientes características:

- El área a barrer es un cuadrado de 4 km^2.
- La posición horizontal inicial de los UAVs es 100 metros a la izquierda del área.
- La altura de los UAVs es 100 metros.
- La velocidad inicial es de 20 m/s.
- La trayectoria inicial de los UAVs es horizontal con sentido hacia la derecha.
- El radio mínimo de giro de los UAVs es 40 metros.

En las simulaciones se ha variado el ángulo de FOV desde 30° hasta 70°.

En la figura 4.13 se observa que al aumentar el campo de visión de la cámara, disminuye notablemente el tiempo de barrido como debería ser.

En la figura 4.14 se aprecia que el tiempo de barrido aumenta al desactivarse un UAV. Se ha desactivado el mismo UAV que en las dos simulaciones anteriores y se han realizado simulaciones para la desactivación a los 300, 600 y 900 segundos. Vista la gráfica se pueden sacar las siguientes conclusiones:

- A partir de 60°, al tardar la simulación menos de 900 segundos, ya no da tiempo al desactivarse el UAV en ese tiempo.
- Al igual que en los casos anteriores, cuanto antes ocurra la desactivación, peores resultados arroja la simulación por el hecho de que tienen que barrer un mayor área un número inferior de UAVs.

Vista las gráficas se puede afirmar que esta simulación da resultados coherentes y a priori, válidos.

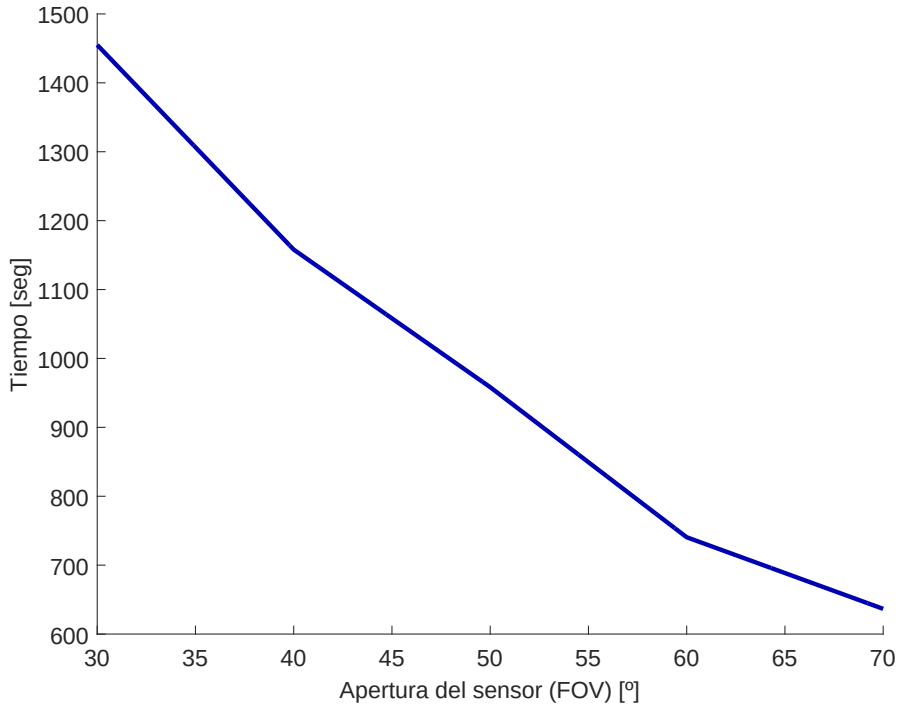

Figura 4.13 Campo de visión frente a tiempo de barrido.

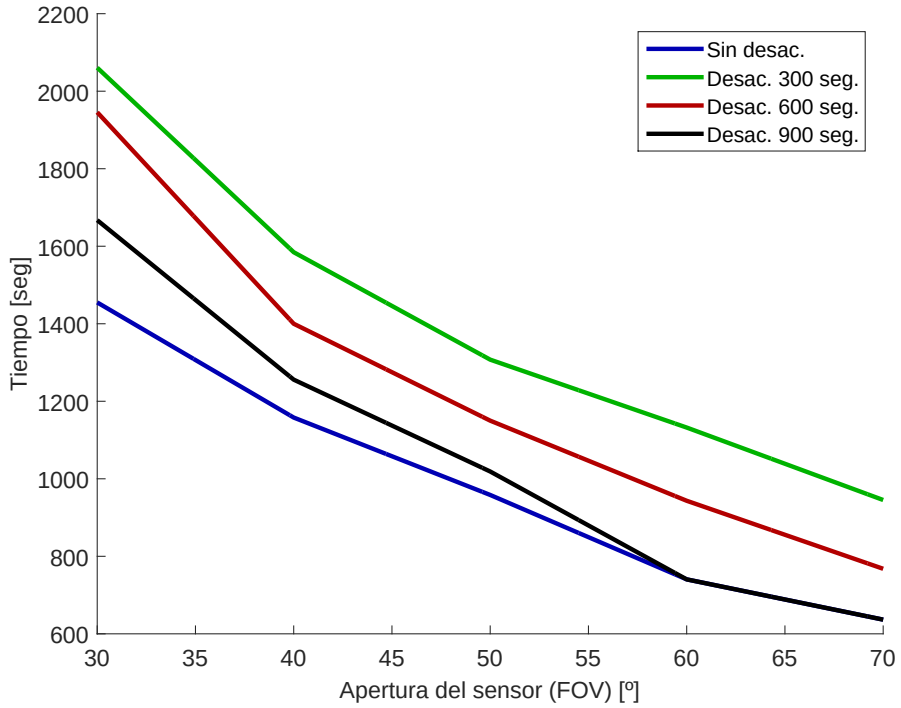

Figura 4.14 Campo de visión frente a tiempo de barrido con desactivación.

5 Búsqueda de objetivos

En el segundo programa del presente trabajo se van a estudiar las características que tienen varios conjuntos de UAVs al barrer un área rectangular, para ello se van a utilizar los mismo métodos seguidos en la explicación del programa anterior (capítulo 4).

Este programa se basa en el barrido de un área rectangular por parte de varios conjuntos de UAVs, para ello se ha barrido el área en cuatro zonas idénticas para ser barridas cada una por un conjunto diferente de UAVs. Al igual que en el programa anterior, también se tiene la posibilidad de controlar todos los parámetros desde las interfaces gráficas.

En este programa a diferencia del anterior, hay un UAV líder, más grande que los UAVs de barrido, el cual suelta a los UAVs uno a uno para que barran cada uno su área, también se tiene la posibilidad de poner objetivos por el mapa para que cuando los UAVs de barrido los localicen, salga un último UAV cuya función es ir a cada localización de los objetivos y dar vueltas a cada uno de ellos para su reconocimiento en una mayor profundidad.

5.1 Modelo

Se van a explicar las características que tiene el modelo del programa explicado. Antes de nada, se exponen las diferentes hipótesis utilizadas en el programa:

- El mapa donde se desarrolla la simulación se supone plano.
- El UAV líder se inicia en el punto más al norte del mapa con dirección oeste. El movimiento del líder es dar vueltas en torno al área general de barrido.
- La velocidad lineal de los giros de todos los UAVs será constante.
- Cada UAV sigue sus propios waypoints, por lo que no tiene en cuente a los otros UAVs. Para que no haya colisiones en vuelo se deberá colocar a cada UAV a una altura diferente y con un margen de seguridad para que no haya interferencias de vuelo entre ellos.

A continuación se van a desarrollar las características de los UAVs y de los conjuntos.

5.1.1 Características de los UAVs

En el programa hay tres tipos de UAVs, el líder, los de barrido y el hunter, cada uno de ellos con sus diferentes características que se van a detallar explicando cada uno de ellos por separado.

UAV líder

Este UAV es el principal del programa y su movimiento consiste en un giro continuo en sentido antihorario y con un radio igual al tamaño del mapa en dirección vertical dividido entre tres (figura 5.1).

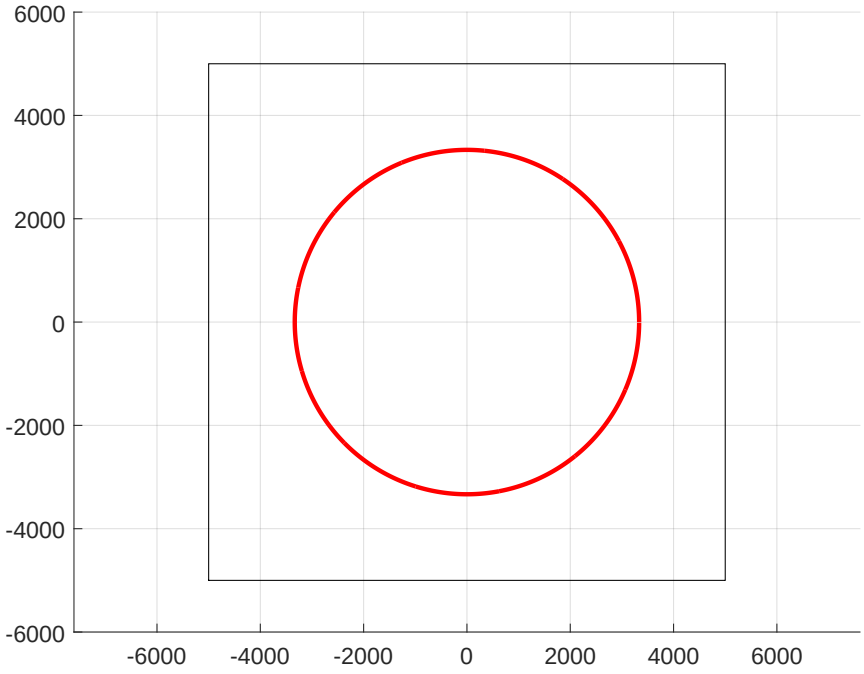

Figura 5.1 Movimiento del UAV líder.

Este UAV vuela a una altura superior a todos los demás UAVs del programa, en concreto, se ha introducido que vuele a una altura de 1.000 metros, aunque este valor se puede modificar desde la interfaz gráfica. Es el encargado de soltar a todos los demás UAVs del programa, para ello, cuando éste UAV alcance un cuadrante el rectángulo del mapa empieza a soltar a los UAVs del conjunto de dicho cuadrante y los suelta de uno en uno con un margen de unos segundos. Dicho margen se puede modificar desde las interfaces gráficas.

UAVs de barrido

Los UAVs de barrido son los encargados, como su propio nombre indica, de barrer todo el área en busca de objetivos. La particularidad de estos UAVs a diferencia del programa anterior, es que parten del UAV líder, por lo que las condiciones iniciales son las que tenga dicho UAV, esto provoca que tengan que disminuir su altura hasta la altura deseada para barrer el área y la velocidad también será necesaria que la disminuyan porque el UAV líder vuela a otras condiciones de vuelo.

Estos UAVs forman cuatro conjuntos que se encargan de barrer los cuatro cuadrantes del rectángulo. La forma de hacerlo es aplicando los mismos algoritmos que se aplicaron en el capítulo anterior, puesto que cada conjunto tiene que barrer la misma área que se barrió en el capítulo anterior.

UAV hunter

Este UAV es el encargado de identificar en mayor profundidad cada objetivo localizado por los UAVs de barrido, así pues entra en funcionamiento cuando se ha encontrado algún objetivo y también es soltado por el UAV líder, por lo que tiene que cambiar sus condiciones de vuelo a unas que le resulten óptimas para la identificación precisa de objetivos.

Su movimiento consiste en dirigirse hacia un objetivo y darle cinco vueltas, una vez dada las cinco vueltas se supone que ha sido suficiente para identificar al objetivo y se va hacia otro objetivo descubierto. Si no hubiera ningún otro objetivo o no estuviera identificado por los UAVs de barrido, permanecería dando vueltas en torno al objetivo encontrado, si existiese otro objetivo iría alternando entre los dos objetivos con cinco vueltas a cada uno de ellos.

5.1.2 Características de los conjuntos de UAVs

Los conjuntos están compuestos de UAVs de barrido, y su movimiento es como el de la figura 5.2, cada una de ellas es la encargada de barrer un cuadrante del rectángulo inicial y estará compuesta de varios UAVs.

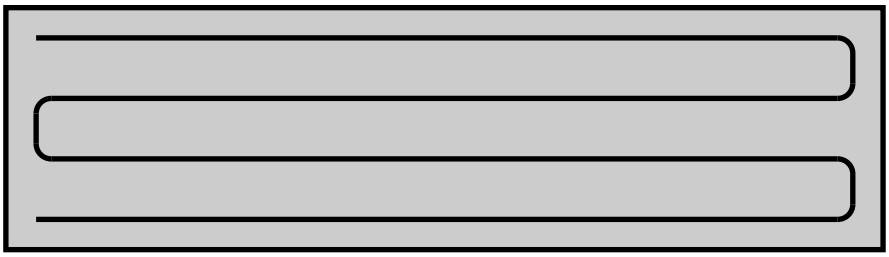

Figura 5.2 Movimiento de un conjunto de UAVs.

Cuanto mayor sea el número de UAVs que compongan el conjunto, mayor será la rapidez con la que se barrerá la misma.

En la figura 5.3 se aprecia el movimiento de todos los conjuntos y de cómo se ha definido cada cuadrante del rectángulo. Decir que cuando un conjunto gira, se encuentra con otra conjunto, por lo que es necesario que los UAVs vuelen a diferentes alturas, o bien, que se disponga de un buen control en altura de los UAVs para evitar colisiones entre ellos.

5.2 Modelado en MATLAB®

En esta sección se va a explicar cómo se ha programado todo lo visto anteriormente en el entorno MATLAB® , decir que no se van a explicar aquí las interfaces gráficas, que se hará posteriormente.

Siguiendo con lo visto en el capítulo 3, se van a explicar las clases utilizadas para la programación de este programa, que no son más que la clase de los UAVs y la clase de los conjuntos:

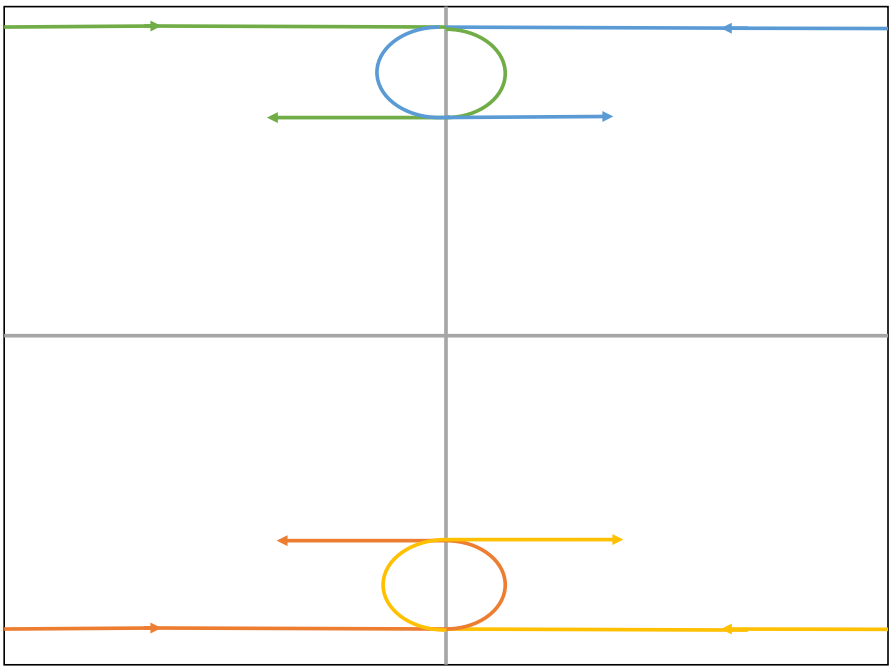

Figura 5.3 Movimiento de los conjuntos de UAVs.

5.2.1 Clase UAV

Esta clase de MATLAB® define a cada uno de los UAVs, es similar a la del capítulo 4, pero tiene algunas diferencias. A continuación se definen cada una de las variables de la clase:

- x: posición horizontal del objeto.
- y: posición vertical del objeto.
- h: altura del objeto.
- H_{obj}: altura objetivo del objeto. La diferencia con la anterior es que esta es la deseada por el objeto.
- v: velocidad del objeto.
- V_{obj}: velocidad objetivo del objeto. Similar al caso de las alturas.
- θ: ángulo de trayectoria del objeto medido desde la horizontal. Se mide en grados y tiene un valor entre 0 y 2π.
- $Rmin$: radio mínimo de giro del objeto.
- fov: apertura del sensor que lleva la cámara del objeto, medido en grados.
- act: variable que indica si el objeto está activo, tomando el valor uno cuando si está activo y cero cuando no lo está.

- *tierra*: variable que indica si el objeto está en tierra, tomando el valor uno cuando si lo está y el valor cero cuando no lo está.
- *vuelo*: variable que indica en que fase de vuelo está el objeto, siendo el valor uno si está en vuelo, el valor cero si está despegando y el valor dos si está aterrizando.
- Wp: variable con todos los waypoints por los que tiene que pasar el UAV.
- Wp_{act}: variable que indica a qué waypoints de los expuestos en la variable anterior se tiene que dirigir.

Esta clase también dispone de diferentes métodos que sirven para una programación más limpia en MATLAB® . Dichos métodos son:

- *Alt_new*: es un controlador muy simple en altura, para que la altura del objeto tienda a la altura objetivo del mismo.
- *Vel_new*: es un controlador similar al de la altura pero en velocidad.
- *Datos*: sirve para coger cualquier dato del objeto.
- *Guardar*: sirve para guardar datos en el objeto.
- *display*: método que sirve para que muestre correctamente los datos del objeto por pantalla.
- θ*_new*: es un controlador simple para que el objeto se dirija al UAV establecido.

Con esto ya se tiene definido a cada UAV del programa, incluido al UAV líder y al UAV hunter porque las características de todos los UAVs son comunes.

5.2.2 Clase Swarm

La clase *Swarm* programada en MATLAB® se corresponde con los conjuntos de UAV del programa. En este programa sólo se ha creado una única variable de la clase *Swarm* y en esa variable se han introducido todos los conjuntos que hay en el programa, uno por cada cuadrante, en forma de vector. Esto provoca que todos los conjuntos al estar en una misma variable en forma de vector, tienen que tener obligatoriamente las mismas variables. Así pues, las variables de esta clase son:

- *num*: número de UAVs que forma el conjunto.
- *UAV*1: primer UAV del conjunto.
- *UAV*2: Segundo UAV del conjunto.
- Y así sucesivamente hasta tener todos los UAVs del conjunto en forma de variables dentro de la clase.

Esto provoca que todas los conjuntos tienen que tener el mismo número de UAVs para que el número de sus variables coincidan. Para el caso que ocupa al programa, se han introducido UAVs *vacíos* para el caso en que un conjunto tenga menos UAVs que el resto, así el programa no da errores.

El código 5.1 es el empleado para crear la clase *Swarm* en MATLAB® , se aprecia en él que la entrada a la función es un vector con todos los UAVs que van a formar la clase.

Código 5.1 Creación de la clase Swarm.

```
function Swarm = Swarm(UAVs)

% Función que crea el objeto Swarm, que contiene las caracterí
    sticas de x UAVs, siendo x cualquier número.

Swarm.num = length(UAVs);

for k=1:Swarm.num,
Swarm = setfield(Swarm,['UAV' num2str(k)],UAVs(1));
end

Swarm = class(Swarm,'Swarm');

end
```

Los métodos de esta clase son:

- *display* (código 5.2): muestra por pantalla los datos del objeto. En este caso sólo muestra cuantos UAVs tiene dentro el conjunto, aunque no hay que tener en cuenta este dato, porque el conjunto puede tener UAVs *vacíos* que a efectos del programa no tienen valor pero si que lo tienen para el objeto creado de esta clase.
- *Guarda_uav* (código 5.3): guarda los datos del UAV en el objeto de la clase *Swarm*. Tiene como entradas el objeto de la clase *Swarm*, el objeto de la clase *UAV* que se va a guardar y el número de UAV que es dentro del conjunto.
- *Swarm_uav* (código 5.4): método que devuelve el objeto UAV deseado del objeto de la clase *Swarm*. Tiene como entradas el objeto de la clase *Swarm* y el número dentro del conjunto del UAV que se quiere.

Código 5.2 Función display de la clase Swarm.

```
function display(Swarm)

% Función que muestra el objeto Swarm en la ventana de comandos.

disp(' ');
disp([inputname(1),' = ']);
disp(' ');
fprintf('Número de UAVs en el enjambre: %d\n',Swarm.num);
disp(' ');

end
```

Código 5.3 Función Guarda_uav de la clase Swarm.

```
function Swarm = Guarda_uav(Swarm,UAV,num)

% Función que guarda el UAV en el enjambre y en el sitio que se le
    diga.

% Swarm = setfield(Swarm,['UAV' num2str(num)],UAV);
S.type = '.';
S.subs = ['UAV' num2str(num)];
Swarm = subsasgn(Swarm,S,UAV);

end
```

Código 5.4 Función Swarm_uav de la clase Swarm.

```
function UAV = Swarm_uav(Swarm,D)

% Función que devuelve las características del UAV que está
    guardado en la posición correspondiente en el enjambre. Si se
    introduce como variable D el string 'num', la función devuelve
    el número de UAVs que hay en el enjambre.

if strcmp(D,'num'),
UAV = Swarm.num;
else
UAV = eval(['Swarm.UAV' num2str(D)]);
end

end
```

5.3 Interfaces gráficas (GUIDE)

Para manejar todos los datos se ha decidido crear varias interfaces gráficas sencillas desde las cuales se muy simple modificar cualquier parámetro de la simulación. Se ha optado por crear tres interfaces gráficas, cada una destinada a un objetivo diferente, la interfaz principal para ejecutar y visualizar la simulación, la interfaz de los datos de entrada para controlar todos los parámetros que no tienen que ver con los UAVs y la interfaz de los UAVs para controlar los parámetros que tienen que ver con los UAVs.

Al igual que en el programa anterior, no es lo mismo ejecutar que visualizar la simulación, por lo que la interfaz principal (figura 5.4) del programa es muy similar a la del programa anterior exceptuando dos diferencias:

- Se ha añadido al botón *Menú* un botón para que abra la interfaz de los datos de los UAVs.
- Se ha eliminado el botón del *Sombreado* porque en esta simulación al ser cuatro conjuntos va a haber muchos UAVs en vuelo y si se sombrea por donde pasa cada uno de ellos se colapsaría MATLAB® .

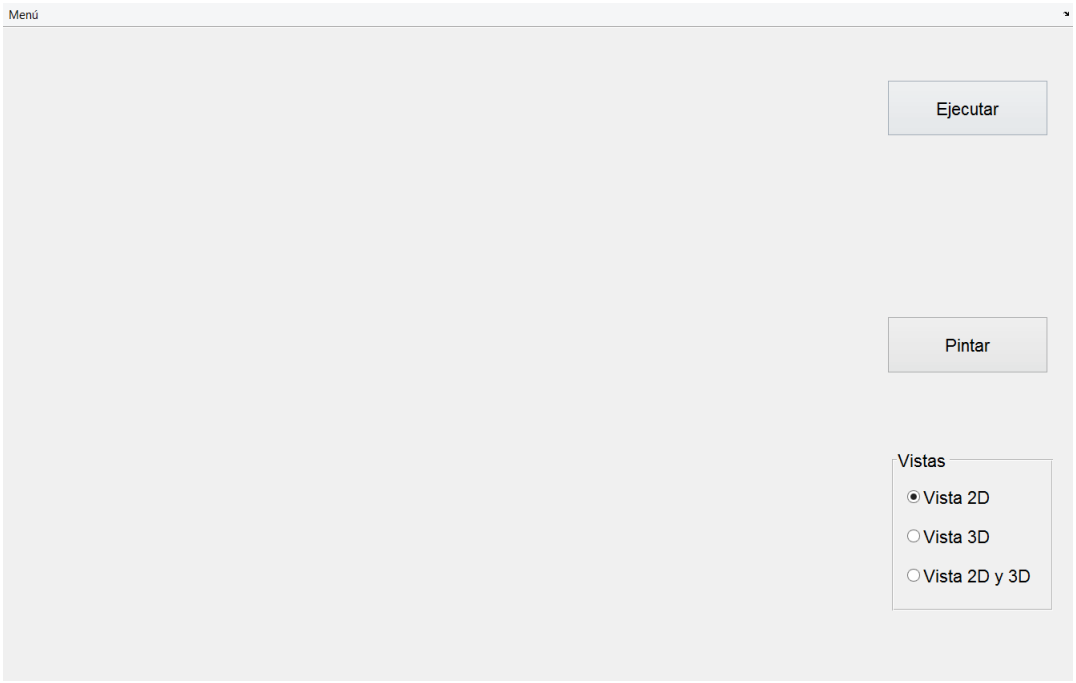

Figura 5.4 Interfaz principal.

5.3.1 Interfaz de los datos de entrada

Esta interfaz (figura 5.5) tiene como objetivo modificar de forma sencilla los datos de entrada de la simulación. Consta de varias partes que se van a definir a continuación:

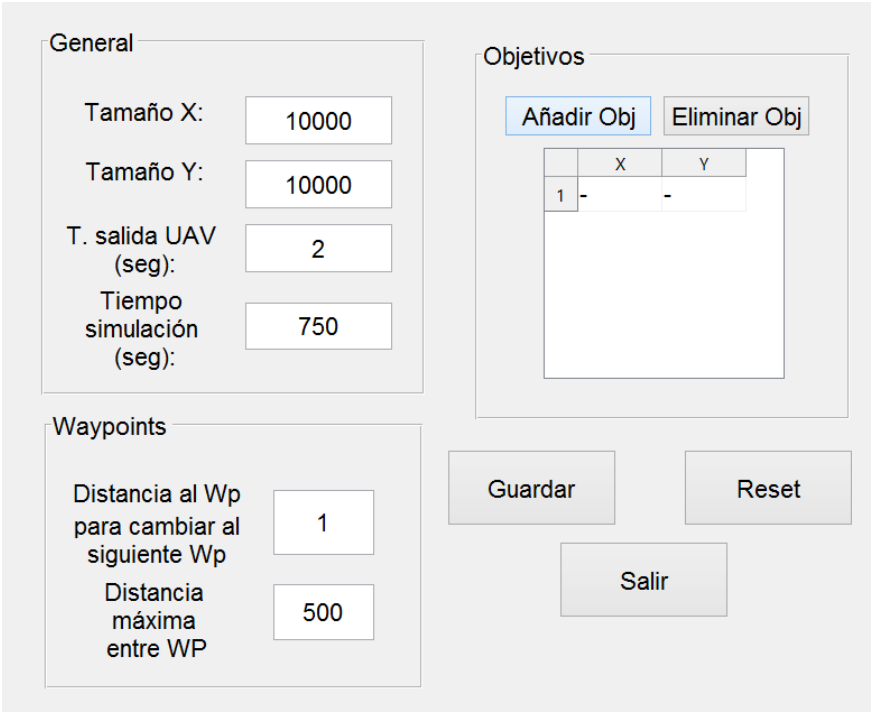

Figura 5.5 Interfaz de los datos de entrada.

- Panel *General*: consta de diferentes parámetros que se pueden modificar como el

tamaño horizontal del mapa de barrido, el tamaño vertical del mapa de barrido, el tiempo de salida del UAV líder que hay entre dos UAVs consecutivos y el tiempo de que va a durar la simulación.

- Panel *Waypoints*: contiene dos parámetros que se pueden modificar, la distancia que tiene que haber entre el UAV y el waypoint para dirigirse hacia el siguiente waypoint y la distancia máxima que hay entre waypoints en la zona de barrido.
- Panel *Objetivos*: desde este panel se modifican los objetivos que van a aparecer en la zona de barrido. para añadir un objetivo pulsar el botón añadir y el botón eliminar para eliminar el último objetivo de la lista. Para modificar la posición de cada objetivo, basta con modificar en la tabla el número manualmente. El mapa se encuentra centrado en el origen de coordenadas, dicho esto, se debe garantizar que al colocar un objetivo tiene que estar dentro del área de barrido.
- Botón *Guardar*: guarda los valores que aparecen actualmente en la pantalla en un archivo .mat para la ejecución de la simulación.
- Botón *Reset*: borra los valores que aparecen en pantalla y los modifica por los que hay guardados en el archivo .mat.
- Botón *Salir*: sale de esta interfaz sin guardar ningún dato.

Al igual que en el programa anterior, el botón *Guardar* y el botón *Reset* requiere de confirmación por parte del usuario, mientras que el botón *Salir* no requiere de dicha orden.

5.3.2 Interfaz de los UAVs

Esta es la última interfaz de este programa (figura 5.6) y sirve para modificar los datos relacionados con los UAVs. Dispone de ciertas características que se explican a continuación:

- En el hueco de número de UAVs se pone el número de UAVs, exceptuando al líder y al hunter, que habrá en la simulación. Exige un número mínimo de cuatro UAVs puesto que hay cuatro conjuntos, pero a partir de ahí se puede introducir cualquier número de UAVs.
- El botón *Ok* actualiza en la tabla el valor puesto en el número de UAVs, es decir, coloca en la tabla tantos UAVs como los que se quieran, pero todos con la misma altura deseada, velocidad deseada, apertura del sensor y radio mínimo de giro.
- En la tabla se pueden modificar los datos de cada UAV que van a estar presentes en la simulación. El líder no requiere ni apertura del sensor, ni radio mínimo porque no va a barrer el terreno y va a estar realizando un giro coordinado constante. El hunter no requiere apertura del sensor porque tampoco va a barrer el área, sino identificar los objetivos.

  El resto de UAVs sí requieren altura deseada, velocidad deseada, apertura del sensor de la cámara y radio mínimo de giro. Para saber qué UAV va a pertenecer a cada conjunto, basta con seguir la siguiente regla:

  – Se calcula cuantos UAVs hay en cada conjunto. De haber UAVs de pico, se añaden estos UAVs a cada conjunto de uno en uno empezando por el conjunto número uno.

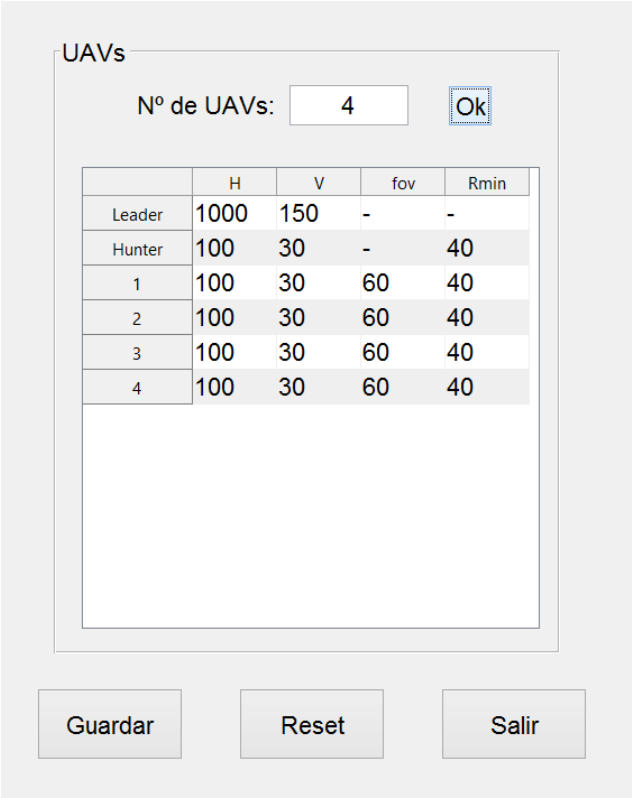

Figura 5.6 Interfaz de los UAVs.

  - Los *x* primeros UAVs corresponden con el primer conjunto, siendo *x* el número de UAVs que tiene el primer conjunto, siendo el número uno el de la parte superior del conjunto y así hacia abajo.
  - Así sucesivamente para cada conjunto.
- Los botones *Guardar*, *Reset* y *Salir* tienen las mismas funciones que los de la interfaz anterior.

5.4 Resultados del programa

Ya se tiene definido completamente el programa 2 del trabajo, pero no se ha visto el programa en ejecución y los resultados pertinentes. Para ello, se ha decidido realizar la misma simulación varias veces modificando los objetivos y ver qué resultados ofrece el programa en cuanto al tiempo que tardan en encontrarse todos los objetivos. Se han ejecutado las simulaciones en estas características:

- El mapa tiene 10x10 km.
- El tiempo de salida de cada UAV del UAV líder es de 2 segundos.
- El UAV líder vuela a 1.000 metros de altura y a una velocidad lineal de 150 metros por segundo.
- El UAV hunter no tiene importancia para estas simulaciones porque solo se va a tener en cuenta cuanto tiempo tardan los UAVs de barrido en encontrar todos los objetivos del mapa.

- Los UAVs de barrido vuelan todos a 100 metros de altura, a una velocidad de 30 m/s, con una apertura del sensor de la cámara de 60° y con un radio mínimo de giro de 40 metros. Al volar a la misma altura se van a producir colisiones entre los UAVs, pero éstas no se tendrán en cuenta. Por lo que éstas simulaciones no serían válidas en la realidad pero sí sirven para tener un rango de tiempos válidos.

Se van a suponer cuatro casos distintos de objetivos en el mapa, todos estos casos diferentes están expuestos en la figura 5.7. A continuación se va a describir cómo se han colocado los objetivos de cada caso:

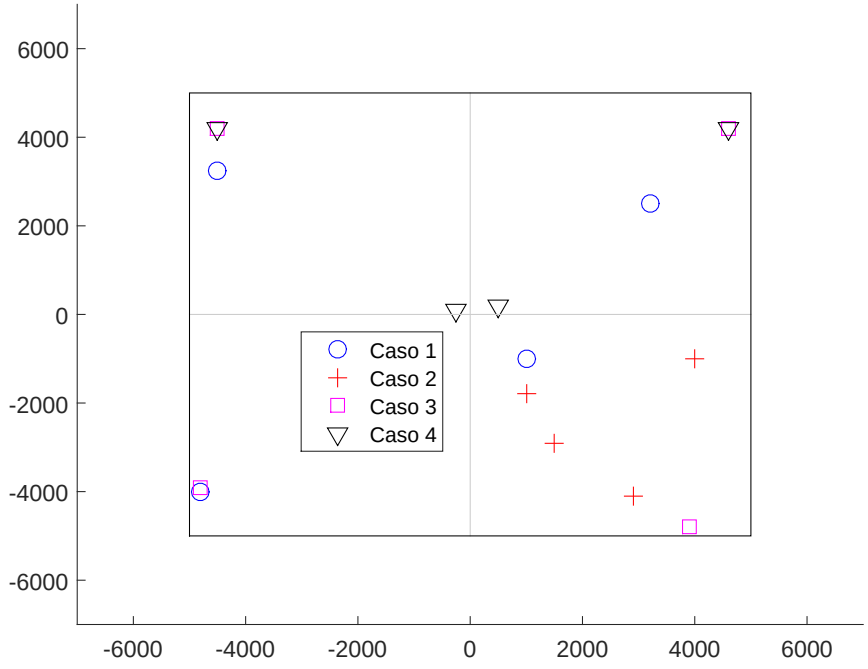

Figura 5.7 Posición de los objetivos para las cuatro simulaciones.

- Caso uno: se caracteriza por ser aleatorio, pero con la condición de tener cada uno de los cuatro objetivos en un cuadrante diferente.
- Caso dos: también es aleatorio, pero con todos los objetivos en un mismo cuadrante.
- Caso tres: todos los objetivos de este caso están cerca de las esquinas.
- Caso cuatro: se caracteriza por tener dos objetivos cerca de las esquinas y otros dos objetivos cerca del centro del mapa.

En las figuras 5.8, 5.9, 5.10, 5.11 y 5.12 se aprecian los resultados obtenidos para cada simulación. En éstas gráficas está representado el tiempo que tardan en encontrar todos los objetivos frente el número de UAVs usados. En principio, habría que decir que este tiempo debe disminuir cuantos más UAVs se usen porque cada uno de ellos tendría que barrer un área menor y entre todos, tardarían menos tiempo en barrer la misma área. Se aprecia en las figuras que es cierto.

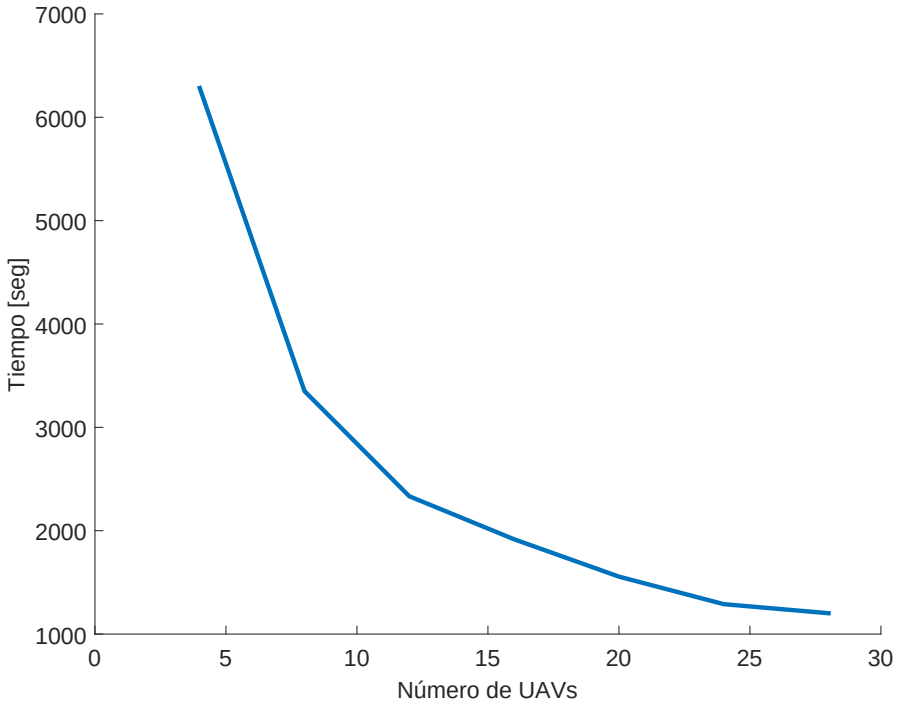

Figura 5.8 Resultados del caso uno.

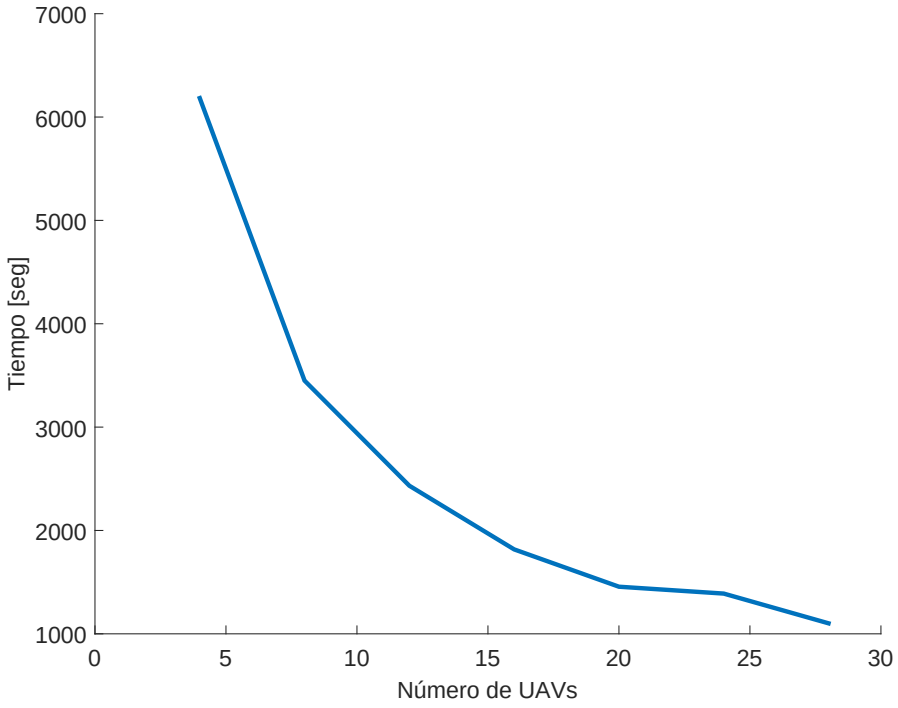

Figura 5.9 Resultados del caso dos.

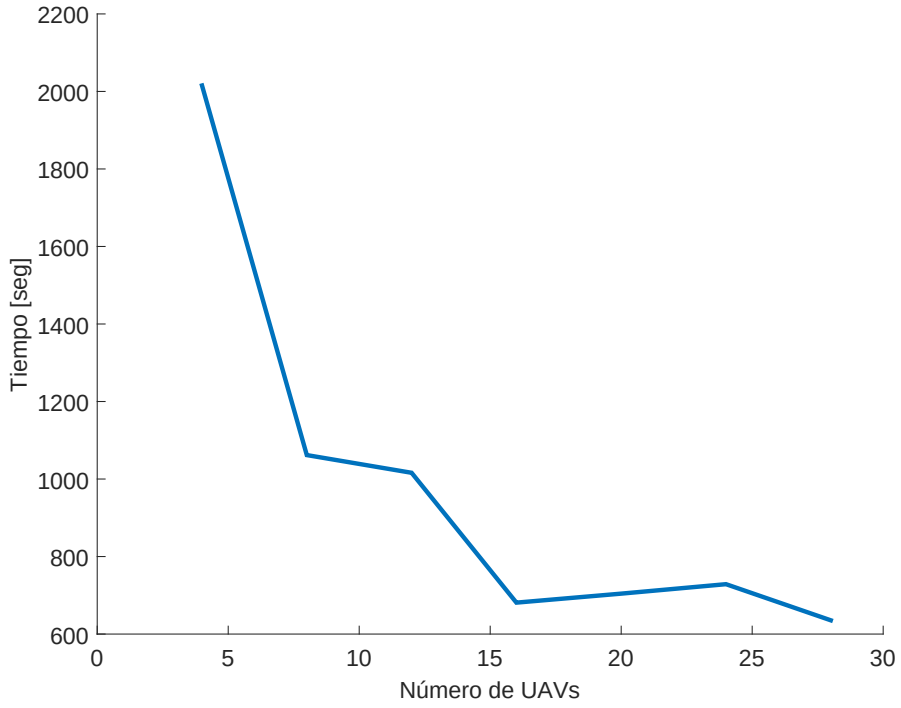

Figura 5.10 Resultados del caso tres.

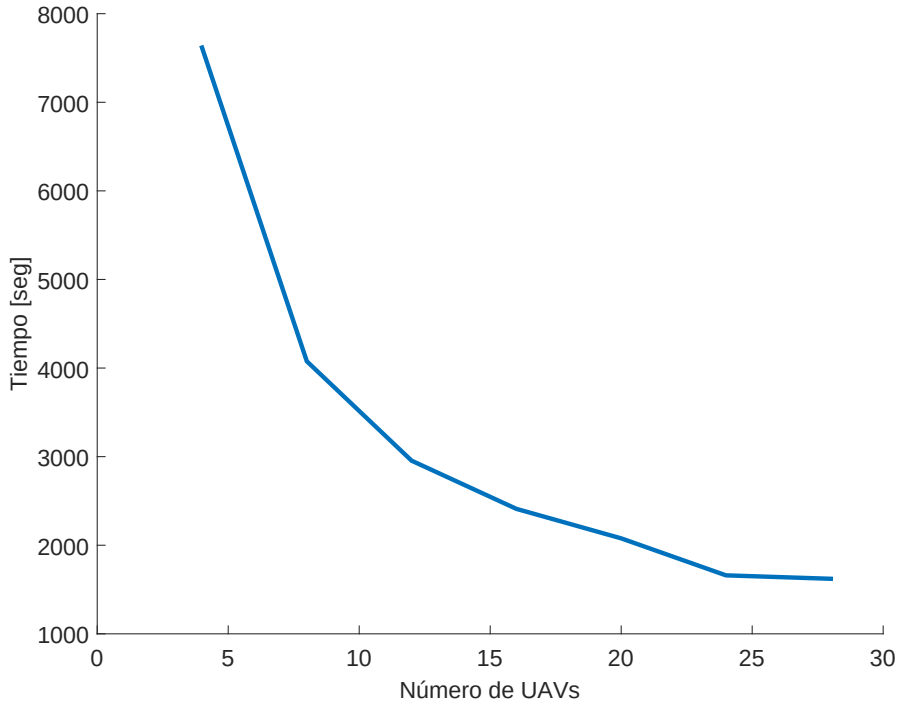

Figura 5.11 Resultados del caso cuatro.

Ahora bien, mirando las figuras independientemente se observa que todas ellas siguen más o menos el mismo patrón, exceptuando el caso número tres, que tiene más perturbaciones y no se ve claro que siga la misma guía que los otros casos. Esto se produce porque los objetivos en este caso están muy cercanos a las esquinas y hace que la solución dependa poco del número de UAVs usados, porque como los conjuntos empiezan a barrer desde las esquinas y se van acercando al centro, los objetivos de las esquinas se van a encontrar muy pronto tanto si hay muchos UAVs como pocos. De aquí se deduce que este caso será de los cuatro el que menos tiempo emplee en encontrar todos los objetivos como demuestra la figura 5.12.

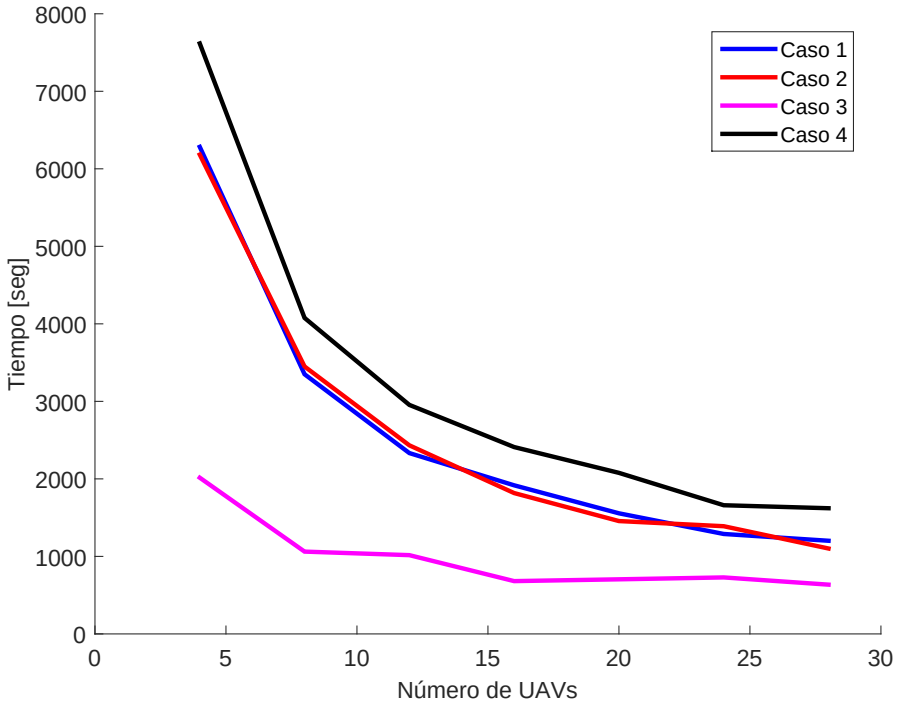

Figura 5.12 Resultados de todos los casos en una misma gráfica.

Viendo todos los resultados obtenidos, se pueden sacar las siguientes conclusiones.

- El caso cuatro tarda más que los otros casos porque tiene objetivos que se encuentran cercanos al centro y al empezar a barrerse por las esquinas, éstos objetivos son los últimos en ser encontrados.
- Por el contrario, el caso tres, al tener los objetivos cercanos a las esquinas, es el caso que menos tarda de los cuatro.
- Para evitar ésto se puede decir que el área se barre como en el programa anterior (capítulo 5), entonces el centro sí se encontraría antes, pero si se pusieran todos los objetivos en un sector como en el caso dos y se barriera como en el programa del capítulo anterior, éste caso sería claramente el perjudicado porque tardarían mucho hasta que llegaran a éstos objetivos. Por lo que se puede decir que ésto es otro método para barrer áreas y que dependiendo de donde se encuentren los objetivos será, o no será el más indicado.

6 Barrido de áreas de forma realista

Este programa es similar al programa uno (capítulo 4), pero se ha intentado conseguir una simulación lo más realista posible. Para explicarlo, se va a seguir un planteamiento similar al que se ha tenido en el programa uno.

El programa se basa en el barrido de un área introducida por el usuario por parte de una serie de UAVs que también son introducidos por el usuario, pero en este caso, los UAVs salen de una pista de despegue y cuando acaban de realizar su labor, vuelven y aterrizan de forma ordenada en la misma pista de donde han despegado.

A diferencia del programa uno, se ha optado por quitar la opción de desactivar UAVs puesto que se ha considerado que la probabilidad de que un UAV durante el barrido tenga un error fatal es mínimo, pero se ha mantenido la idea de crear interfaces gráficas que permitan el cambio sencillo de todos los parámetros del modelo para facilitar la labor al usuario.

6.1 Modelo

Siguiendo el mismo razonamiento que en el capítulo 4, se va primero a explicar las características que tiene el modelo del programa realizado en este capítulo, empezando por las hipótesis que se han realizado para su ejecución:

- El mapa donde se desarrolla, incluida la pista de despegue, se supone plano.
- La velocidad lineal en los giros de los UAVs se supone constante.
- Cada UAV sigue sus propios waypoints, por lo que no tiene en cuenta a los otros UAVs, salvo cuando va a aterrizar, en cuyo caso sí que los tiene en cuenta.

A continuación se van a desarrollar las características de los UAVs, cómo se consigue que cualquier área sea barrida por un número concreto de UAVs y cómo va a ser el movimiento de dichos UAVs a la hora de barrer el área.

6.1.1 Características de los UAVs

Los UAVs vienen definidos por las siguientes características:

- Posición horizontal del UAV.
- Posición vertical del UAV.

- Altura del UAV.
- Altura objetivo del UAV.
- Velocidad del UAV.
- Velocidad objetivo del UAV.
- Ángulo de trayectoria del UAV, medido en radianes. Siempre medido respecto el eje horizontal.
- Radio mínimo de giro del UAV.
- Apertura del sensor que tiene la cámara que lleva el UAV.
- Hay tres parámetros adicionales que indican cosas diferentes y que son:
  - Variable que indica si el UAV está activo o no. Vale el valor uno si está activo y el valor cero si no lo está. La variable se llama *act*.
  - Variable que indica si el UAV está tocando tierra o no lo está. Tiene el valor uno si está tocando tierra o cero si no lo está. La variable se llama *tierra*.
  - Variable que indica en qué fase de vuelo está la aeronave. Tiene el valor uno si está en vuelo, el valor cero si está en la fase de despegue y el valor dos si está en fase de aterrizaje. La variable se llama *vuelo*.

La diferencia entre altura y altura objetivo, es que la altura es la que tiene actualmente el UAV, mientras que la altura objetivo es la deseada por el UAV. Igual pasa con la velocidad y la velocidad objetivo.

Con estos parámetros se tiene perfectamente definido un UAV y lo que está haciendo en cada momento de la simulación.

6.1.2 Partición del área

Una vez que se tienen bien definidas las propiedades de los UAVs, es necesario estudiar como se va a partir el área a barrer en tantas partes como UAVs se vayan a desplegar. Para ello, se van a seguir los siguientes pasos:

- Se calcula el vértice del área más cercano al final de la pista de despegue o inicio de la pista de aterrizaje. Se ha hecho así, para que el tiempo que tardan los UAVs desde que pasan de la fase de despegue a la fase de vuelo hasta que llegan al área para empezar a barrerla sea mínimo.
- Una vez seleccionado el vértice por el que van a empezar a barrer los UAVs, se une dicho vértice con los otros vértices del área.
- Si hay tantas subáreas como UAVs ya estaría hecha la división, siendo cada una de las subáreas un triángulo, si no fuera así, podrían suceder dos variaciones:
  - Que hubiera más UAVs que subáreas, en cuyo caso se toma la subárea más grande y se divide en dos áreas más pequeñas. La recta divisoria se toma desde el vértice por el que empiezan a barrer hasta la mitad del segmento que unen los otros dos vértices. Se repite este proceso hasta que el número de UAVs coincida con el número de subáreas.

– Que hubiera menos UAVs que subáreas. En este caso, se toma el área más pequeña y se une al área adyacente a ésta más pequeña, formando un área que ya no será un triángulo. Se repite este proceso hasta que el número de UAVs coincide con el número de subáreas.

Destacar que el área a barrer tiene que estar formada **obligatoriamente** por arcos convexos.

Para aclarar como funciona esta parte de división de áreas, se ha decidido por poner un ejemplo práctico con un área aleatoria que se encuentra en la figura 6.1, la cual cuenta con tres áreas bien definidas. En la figura 6.2, se aprecia que el número de UAVs va disminuyendo a dos en la parte izquierda de la figura y a uno en la parte derecha. Para pasar de tres a dos UAVs, se ha tomado el área más pequeña y se ha unido con el área adyacente más pequeña, quedando como área final un triángulo y un cuadrilátero que está formado por los dos triángulos fusionados. Para el paso de dos a un único UAV, se han fusionado los dos únicos triángulos que hay en el área. Para este último caso es lógico que si sólo hay un único UAV, sea éste el que barra todo el área inicial.

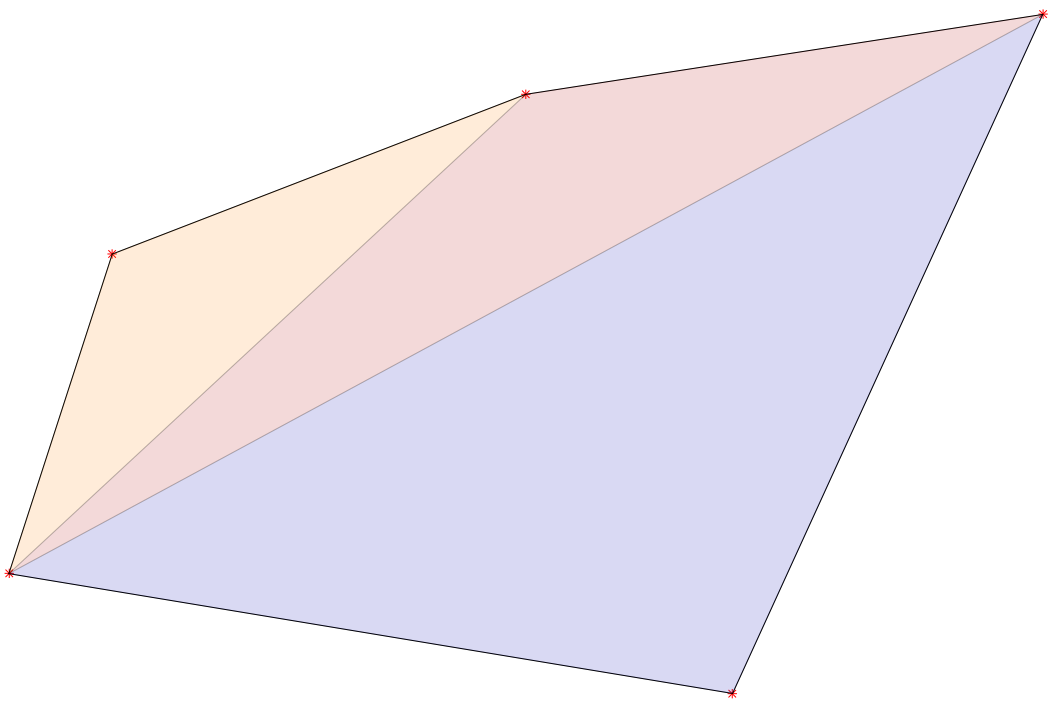

Figura 6.1 Área inicial.

Si hubiera más UAVs que subáreas, pasaría lo que hay descrito en la figura 6.3, en la cual la parte izquierda corresponde a cuatro UAVs y la parte derecha a cinco UAVs. Para el paso del área inicial a cuatro UAVs, se ha tomado el área mayor y se ha dividido por la mitad del segmento que une los dos vértices que no son el vértice principal, formando dos triángulos más pequeños que el ya existente. Para el paso de cuatro a cinco se ha hecho

exactamente los mismo. Y así se seguiría haciendo hasta que el número de UAVs coincida con el número de subáreas.

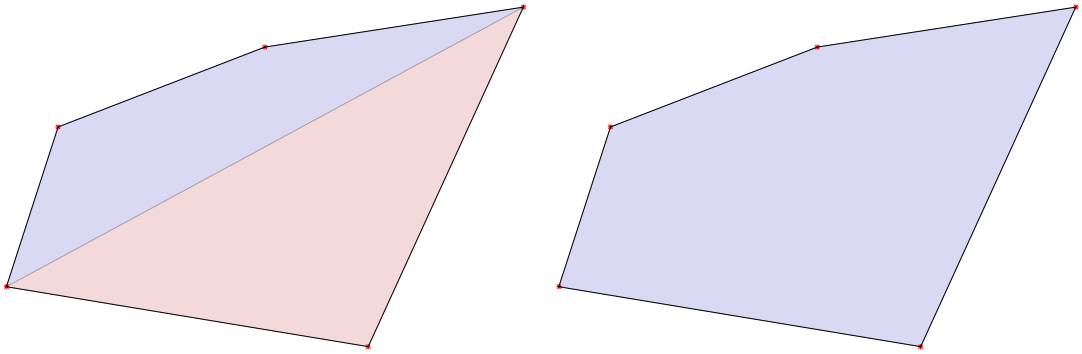

Figura 6.2 Áreas con dos y un UAV respectivamente.

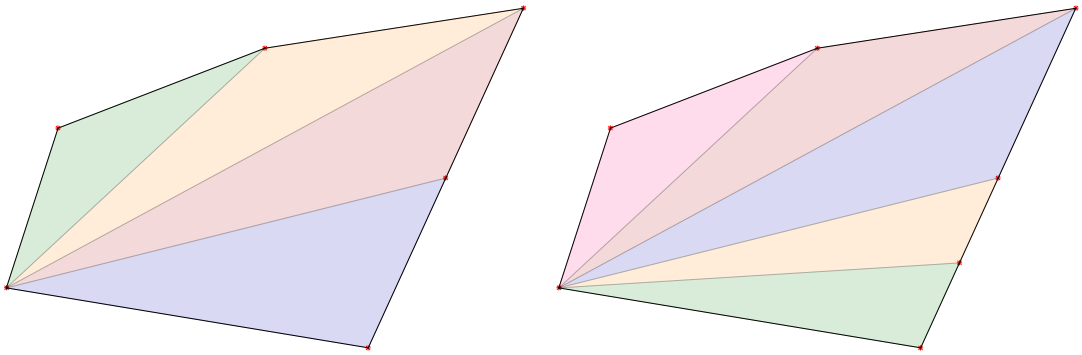

Figura 6.3 Áreas con cuatro y cinco UAV respectivamente.

En las figuras mostradas se aprecia que cada subáreas tiene un color que, lógicamente, se corresponde con cada subárea que tiene que barrer un UAV diferente, pero como se sabe que UAV tiene que hacer cada área. Para ello, se ha optado por considerar las capacidades de cada UAV, y el que tenga mejores capacidades que barra el subárea de mayor tamaño. Por tanto, se ha elaborado un sistema de puntos que se le dan a cada UAV y el que mayor puntuación obtenga será el que mejores capacidades tengo y por tanto el que tiene que barrer el área más grande.

Para obtener los puntos de cada UAV se han tenido en cuenta tres factores importantes que tienen que ver con el barrido:

- Velocidad: cuanto mayor sea la velocidad, mayor área puede barrer un UAV en menor tiempo.
- Altura: a mayor altura respecto del suelo, mayor es el área que ve en un mismo instante de tiempo.
- Apertura del sensor: cuanto mayor sea la apertura, más terreno ve en un mismo instante de tiempo.

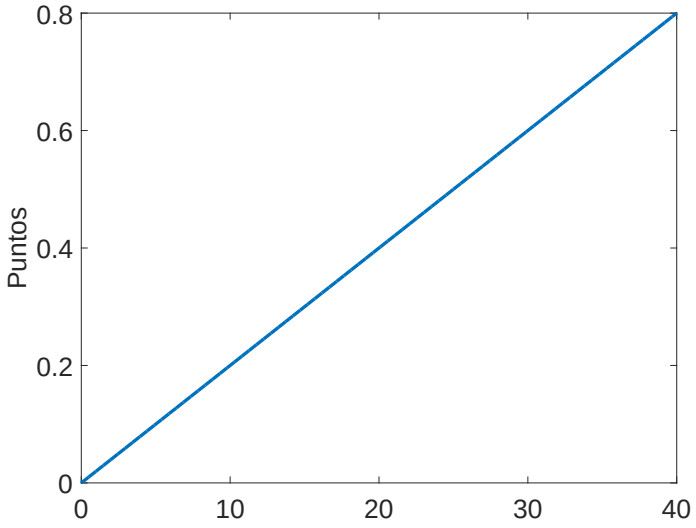

Figura 6.4 Puntos dados por la velocidad.

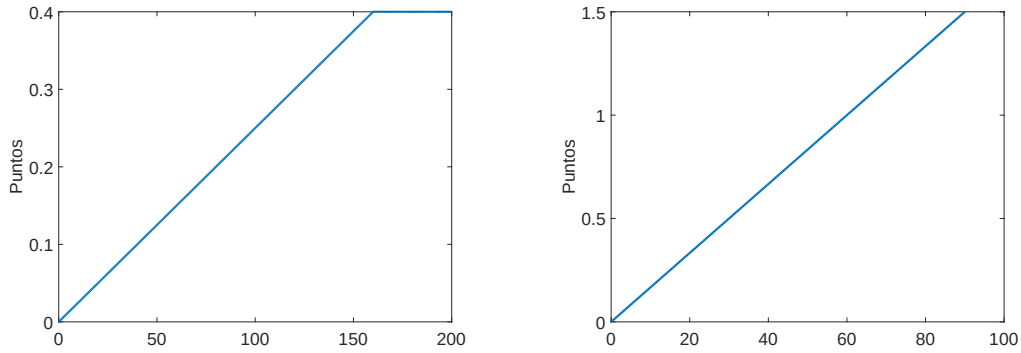

Figura 6.5 Puntos dados por la altura y la apertura del sensor.

De las figuras 6.4 y 6.5 se aprecia que las tres gráficas son lineales con su respectiva magnitud, pero la altura tiene una saturación superior en 150 metros de altura.

De las tres magnitudes que evalúan las capacidades de los UAVs, se ve que la apertura del sensor es la que más influye, puesto que da un valor de 1,5 puntos para un valor de la apertura del sensor de 90°, después la velocidad y por último, la altura a la que se encuentre el UAV.

6.1.3 Poner los waypoints a cada subárea

Para dar los waypoints a cada UAV se sigue el algoritmo 5. Los waypoints se dan en forma de zigzag para barrer todo el área con el número mínimo de pasadas, de acuerdo a la figura 6.6.

Comparando las figuras 6.7 y 6.8 con el algoritmo 5, se ve perfectamente cada uno de los pasos que se van dando hasta llegar a dar los waypoints. Los que más se aprecia es que al dar los waypoints, se alinean respecto el eje vertical para que sea más fácil el giro a los UAVs, esto se aprecia en la parte izquierda de la figura 6.8.

Algoritmo 5 Dar los waypoints a cada UAV

1: Se calcula el vértice más próximo a la zona donde ha alcanzado la altura y velocidad deseada.
2: Se calcula porque lado debe empezar a barrer, siempre por el lado mayor de los dos posibles.
3: Se desplaza el subárea al origen de coordenadas.
4: Se gira el subárea hasta que la recta por donde se va a empezar a barrer quede alineada con el eje x.
5: **si** El polígono está orientado hacia abajo **entonces**
6: Se aplica simetría respecto al eje x en el polígono.
7: **fin si**
8: Se calcula la pendiente y la ordenada en el origen de cada recta del polígono.
9: Se calcula el número de pasadas que va a tener que dar el UAV. El UAV se va a mover por rectas paralelas al eje x.
10: Se calculan los cortes de cada pasada con los lados del polígono, cada recta del UAV va a tener dos cortes con el polígono, uno por la derecha y otro por la izquierda.
11: Se alinean los cortes que tiene cada recta del UAV con el polígono, con el eje y, es decir, se alinean con respecto a una recta vertical el corte de la primera recta por la derecha con la segunda, de la segunda recta por la izquierda con la tercera, de la tercera recta por la derecha con la cuarta y así sucesivamente[a].
12: Se crea la matriz de waypoints.
13: **si** Se aplicó antes el cambio de simetría **entonces**
14: Se deshace el cambio de simetría respecto del eje x.
15: **fin si**
16: Se gira la matriz de waypoints en el sentido inverso al que se giro anteriormente.
17: Se desplaza la matriz de waypoints al contrario de lo que se le realizó antes al polígono.

[a] Esto provoca que los giros que tiene que hacer el UAV son suaves y que barra completamente el área porque entra a cada región del subárea de forma perpendicular que es lo óptimo.

En algunos casos, se le da también al UAV un waypoint de acercamiento a la zona de barrido. Dicho waypoint se sitúa a 200 metros del primer waypoint de la zona de barrido y alineado con la primera hilera de waypoint.

6.1.4 Movimiento de los UAV

El movimiento de los UAVs es diferente según la fase de vuelo. A continuación se explica detalladamente el movimiento en cada una de las tres fases:

- Despegue: durante esta fase de vuelo, los UAVs despegan en línea recta hasta que se alcanza la altura y velocidad deseada, que es cuando pasan a la fase de vuelo. La salida de cada UAV está espaciada un tiempo que se puede modificar en la interfaz. Las ecuaciones que se siguen durante este proceso se corresponden con la sección 2.7.
- Vuelo: en esta fase de vuelo, los UAV siguen los waypoints que tienen asignado. Cuando han barrido todos los waypoints de su área, se dirige a tres waypoints que se llaman *waypoints de aproximación* a la pista de aterrizaje. Cuando pasa por el último waypoint comprueba que no haya ningún UAV con la maniobra de aterrizaje empezada a menos de 300 metros, si es así, comienza la fase de aterrizaje. De no

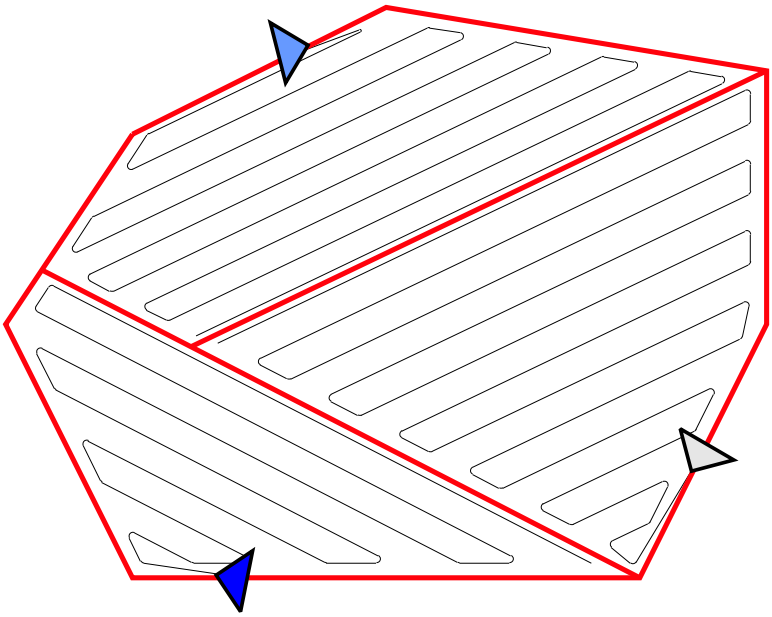

Figura 6.6 Vuelo sobre el área en zigzag.

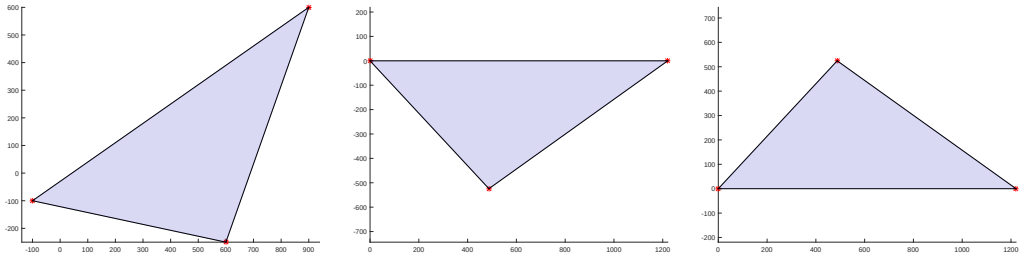

Figura 6.7 Proceso para dar los waypoints I.

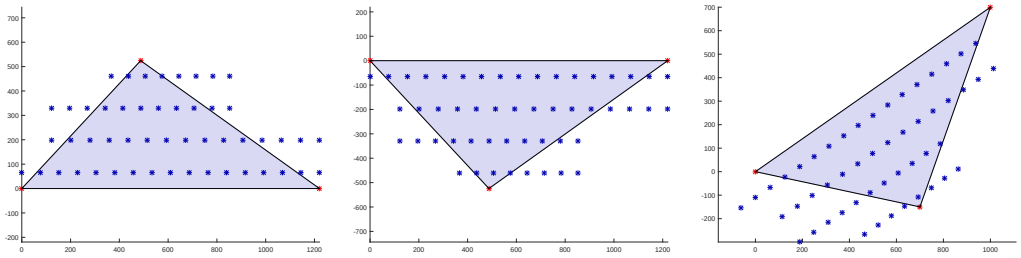

Figura 6.8 Proceso para dar los waypoints II.

serlo, el UAV realiza un giro de 360° sobre si mismo de radio igual a tres veces el radio mínimo de giro del UAV, si cuando vuelve al mismo punto ya no hay ningún UAV con la maniobra de aterrizaje empezada a menos de 300 metros, entonces ya tiene permiso para realizar la maniobra de aterrizaje.

- Aterrizaje: aquí, el UAV realiza el aterrizaje en la pista. Todo esta fase de vuelo se realiza en línea recta y una vez superados los waypoints de aproximación a la pista

de aterrizaje. Las ecuaciones que se siguen durante este proceso se corresponden con la sección 2.8.

En la figura 6.9 se aprecia el mismo área que en las figuras de la sección *partición de área* (sección 6.1.2), la cual la tiene que barrer un único UAV. Puede observarse que los waypoints están dados según se ha explicado en la sección 6.1.3, es decir, el vértice más próximo al de la pista de despegue es el inferior izquierdo y a partir de ahí, se empieza a barrer en zigzag hacia arriba. Cuando acaba de barrer todo el área se dirige a los waypoints de aproximación a la pista de aterrizaje. En la figura 6.9 la pista de aterrizaje se encuentra en el lado izquierdo de la misma alineada con los tres waypoints que se aprecian en la figura.

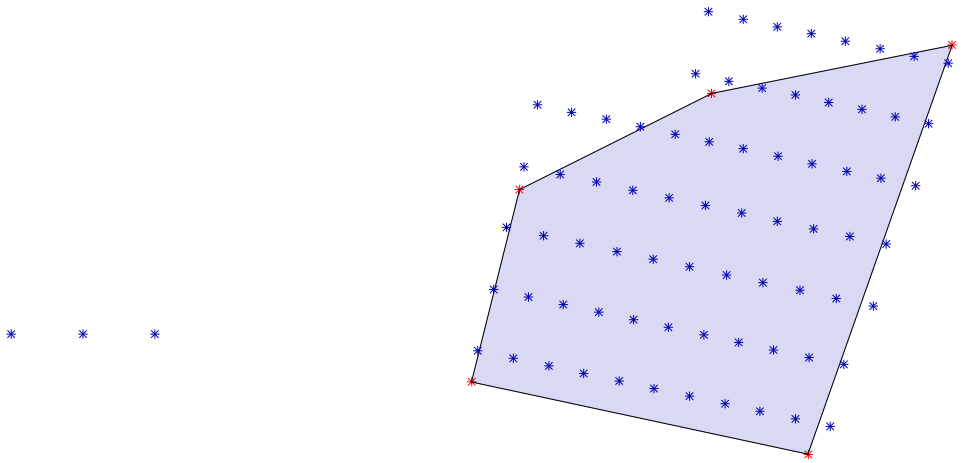

Figura 6.9 Waypoints para un solo UAV.

6.2 Modelado en MATLAB®

Ya está explicado qué hacen y cómo se hacen cada uno de los pasos del programa a realizar, queda por explicar como se ha modelado todo esto en MATLAB® . En esta sección no se van a explicar las interfaces gráficas que se explicarán posteriormente, sino que se va a explicar el código MATLAB® utilizado. Lo primero es explicar como ha influido el capítulo de programación a objetos en esta simulación (capítulo 3).

Para realizar correctamente esta simulación se han creado tres clases que se van a detallar a continuación:

6.2.1 Clase *UAV*

Esta clase es la que define cada uno de los UAVs, es similar a la de los capítulos anteriores, pero tiene sensibles diferencias. A continuación, se definen cada una de las variables que definen a la clase.

- x: posición horizontal del objeto.
- y: posición vertical del objeto.

- h: altura del objeto.
- H_{obj}: altura objetivo del objeto. La diferencia con la anterior es que ésta es la deseada por el objeto.
- v: velocidad del objeto.
- V_{obj}: velocidad objetivo del objeto. Similar al caso de las alturas.
- θ: ángulo de trayectoria del objeto. Se mide en grados y tiene un valor entre 0 y 2π.
- *Rmin*: radio mínimo de giro del objeto.
- fov: apertura del sensor que lleva la cámara del objeto.
- *act*: variable que indica si el objeto está activo, tomando el valor uno cuando si está activo y cero cuando no lo está.
- *tierra*: variable que indica si el objeto está en tierra, tomando el valor uno cuando si lo está y el valor cero cuando no lo está.
- *vuelo*: variable que indica en qué fase de vuelo está el objeto, siendo el valor uno si está en vuelo, el valor cero si está despegando y el valor dos si está aterrizando.

Esta clase también tiene diferentes métodos, son idénticos a los del capítulo 5, que se definen a continuación:

- *Alt_new*: es un controlador muy simple en altura, para que la altura del objeto tienda a la altura objetivo del mismo.
- *Vel_new*: es un controlador similar al de la altura pero en velocidad.
- *Datos*: sirve para coger cualquier dato del objeto.
- *Guardar*: sirve para guardar datos en el objeto.
- *display*: método que sirve para que muestre correctamente los datos del objeto por pantalla.
- θ*_new*: es un controlador simple para que el objeto se dirija al UAV establecido.

Los métodos de coger y guardar los datos y el control en el ángulo de trayectoria son iguales a los del capítulo 4 (códigos 4.2, 4.3 y 4.6 respectivamente). El método del *display* es ligeramente diferente al código 4.1 del capítulo 4 porque en este caso hay más variables que en el otro capítulo.

6.2.2 Clase *Mapa*

Esta clase nueva define el mapa que se está barriendo, el cual se define mediante los vértices del polígono a barrer, la localización de la base y la orientación de la pista. Las variables de esta clase son:

- *Pol*: son los vértices que definen el polígono.
- *Base*: es el punto desde el cual empiezan a despegar los UAVs y en el cual la velocidad al aterrizar es nula.
- *Pista*: es la orientación que tiene la pista. Se define en grados, siendo cero grados el este y 90° el norte.

- *Vertice*: es el vértice del polígono que está más cerca al punto donde los UAVs han pasado de la fase de vuelo a la fase de aterrizaje.
- *Areas*: indica los vértices de todos los triángulos que se definen en el polígono dado.
- *Area*: es el área de cada triángulo anterior.

Las entradas para crear un objeto de esta clase son el polígono, la base y la orientación de la pista. Esta clase tiene varios métodos que se definen a continuación:

- *Datos_mapa*: sirve para coger los datos del mapa en el objeto.
- *Guardar_mapa*: sirve para guardar los datos en el objeto de la clase *Mapa*.
- *Area_pol*: método que calcula cada área del polígono del objeto.

6.2.3 Clase *Are*

Esta es la última clase creada para este programa, la cual está pensada como ayuda a cada UAV. Tiene dos variables, el polígono que delimita qué zona tiene que barrer y el vértice que indica por donde va a empezar a barrerlo. Por último, esta clase tiene un único método que se llama *Datos_are* y que sirve para coger los datos que se deseen de un objeto de esta clase.

6.3 Interfaces gráficas (GUIDE)

La interfaz gráfica de este programa es bastante compleja, pero es que tiene que ser capaz de modificar de forma sencilla cada variable que utiliza el programa. Se divide en una interfaz principal y tres interfaces secundarias, cada una de ellas se encargará de parámetros diferentes del programa.

La interfaz principal (figura 6.10) es similar a la del capítulo 4, por lo que todo lo que se le aplicaba a está interfaz se aplica también a ésta (sección 4.3.1), es decir, para ejecutar la simulación es necesario pulsar el botón *Ejecutar*, pero para pintar la simulación hay que pulsar el botón *Pintar*. Ésto permite que una misma simulación se pueda pintar infinitas veces.

La única diferencia entre ambas interfaces radica en que la de este capítulo, en el botón *Menú*, tiene a su vez cuatro botones y no dos botones como la del capítulo 4. Uno de los botones es el de salir de la interfaz y los otros tres botones sirven para abrir las interfaces secundarias.

Se va a proceder a detallar cada una de las interfaces secundarias, pero antes hay que decir que todas las interfaces tienen tres botones llamados, *Guardar*, *Reset* y *Salir*, que sirven para:

- *Guardar*: Guarda los datos de la interfaz en un archivo .mat que será el que utilice el programa principal.
- *Reset*: Carga los datos del archivo .mat, borrando los datos actuales de la interfaz.
- *Salir*: sale de la interfaz actual sin guardar ningún dato.

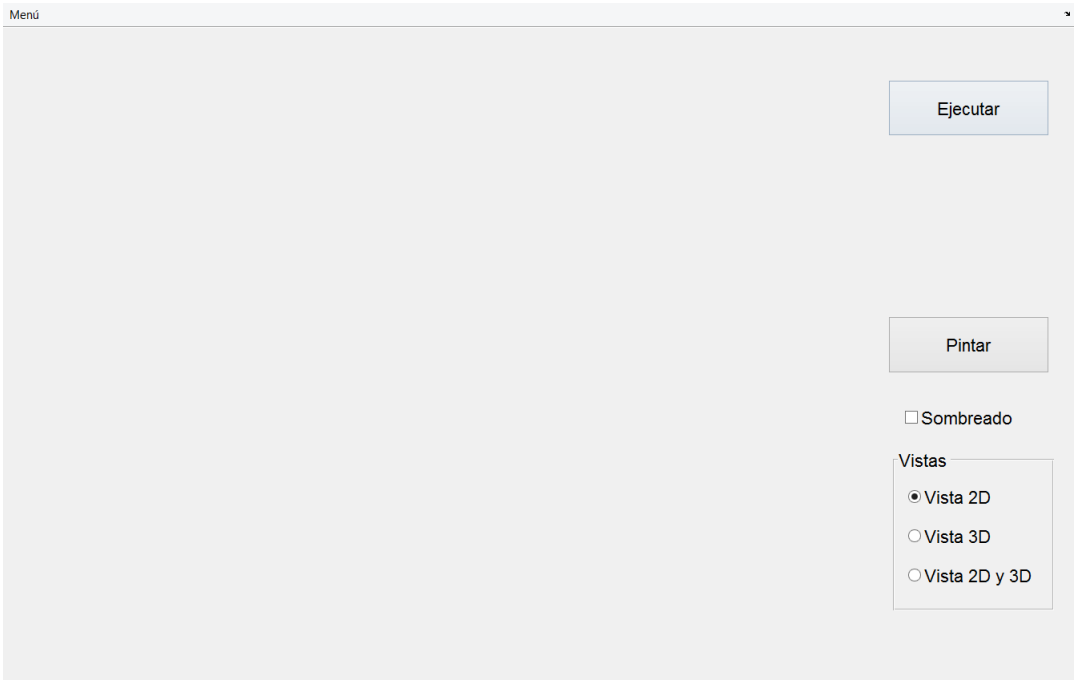

Figura 6.10 Interfaz principal.

6.3.1 Interfaz de los datos de entrada

Esta interfaz, figura 6.11, es la encargada de definir todos los parámetros generales del programa, se divide en tres paneles:

- Panel de las constantes: sirve para definir constantes generales del programa. Por ejemplo, la constante de la gravitación universal (g), el coeficiente de dilatación adiabática (γ), constante de los gases ideales (R), el coeficiente de rozamiento de la pista de aterrizaje y despegue (μ) y el tiempo de salida al despegue que hay entre dos UAVs consecutivos.
- Panel de la base: controla los parámetros relacionados con la base, como por ejemplo, la posición horizontal y vertical de la misma, la altitud a la que se encuentra la base y la orientación de la pista.
- Panel de los UAVs: controla parámetros aerodinámicos y estructurales de los UAVs. Por ejemplo, los coeficientes aerodinámicos de sustentación y resistencia (C_{l_0}, C_{d_0}, C_{l_α} y C_{d_α}), el peso máximo al despegue ($MTOW$), la superficie aerodinámica de los UAVs (S), los ángulos de ascenso y descenso a la hora de despegar y aterrizar (γ_s y γ_d respectivamente) y la velocidad de crucero y de entrada en pérdida de los UAVs (V_c y V_s respectivamente).

6.3.2 Interfaz de los UAVs

Es la encargada de modificar los parámetros relacionados con las características de los UAVs. La interfaz, figura 6.12, es muy simple y solo tiene dos botones, uno para añadir UAVs y otro para eliminar los UAVs. Cuando se añade un UAV, se añade una nueva fila con los mismos datos que la anterior y al eliminar un UAV se elimina la última fila de ellos.

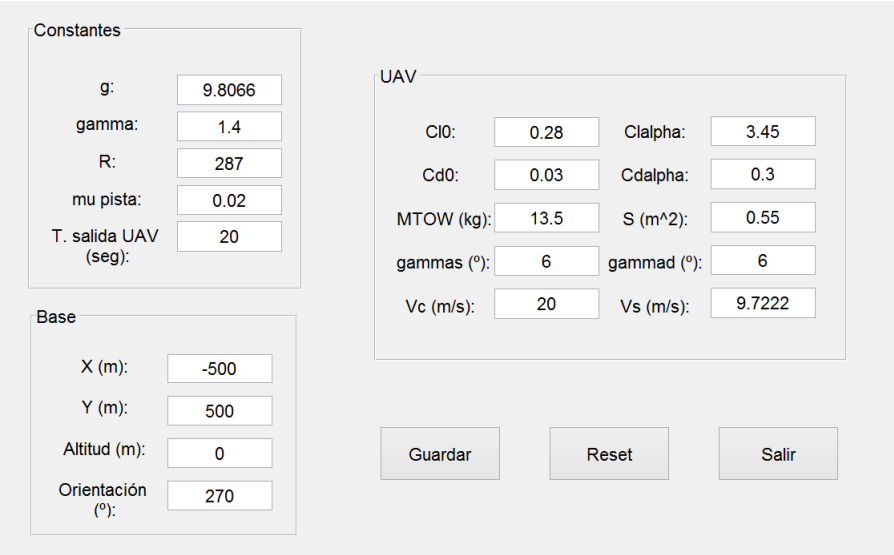

Figura 6.11 Interfaz de los datos de entrada.

Los datos que se pueden tocar de los UAVs son la altura y velocidad deseada, la apertura del sensor de la cámara y el radio mínimo de giro. El orden de los UAVs para salir viene definido por los puntos que tenga cada UAV (sección 6.1.2), porque se entiende que el que más puntos tenga es el que más área tiene que barrer y por tanto, el que antes debería salir.

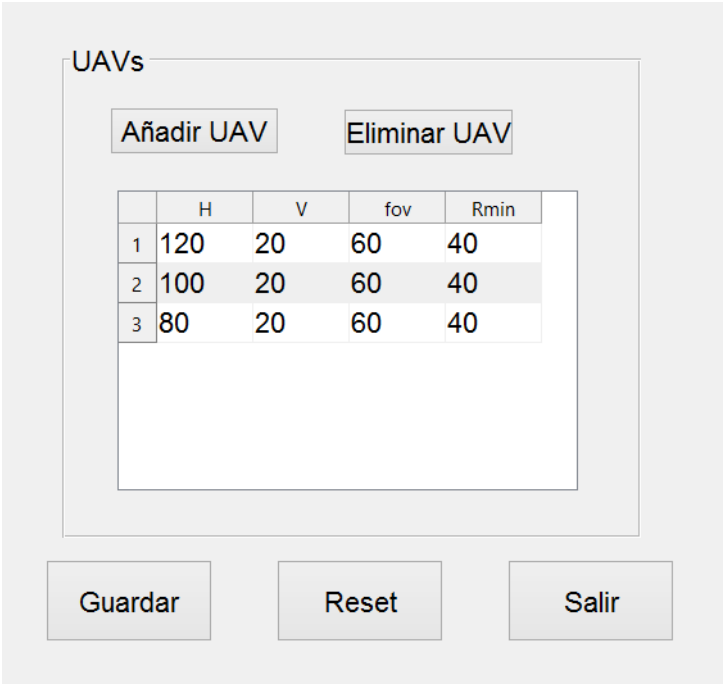

Figura 6.12 Interfaz de los datos de los UAVs.

6.3.3 Interfaz del área

Esta es la última interfaz secundaria, figura 6.13, y sirve para modificar el área que tienen que barrer los UAVs. En la interfaz se aprecian dos paneles y una gráfica:

- Panel del área: permite crear el área. Tiene dos botones, uno para añadir vértices y el otro para eliminarlos. Al crear vértices del área, se crean vértices en el origen de coordenadas, y al eliminarlos se elimina el vértice de la última fila.

  Para cambiar los vértices de posición, basta con cambiarlos en la tabla y cada cambio hará modificar el dibujo que hay en la derecha de la interfaz, permitiendo ver en tiempo real el área que se está creando.

- Panel de los waypoints: sirve para modificar dos parámetros, la distancia, en metros, a la que tiene que estar el UAV del waypoint para cambiar al siguiente waypoint y la distancia máxima, en metros, que tiene que haber entre dos waypoints consecutivos en una misma fila.

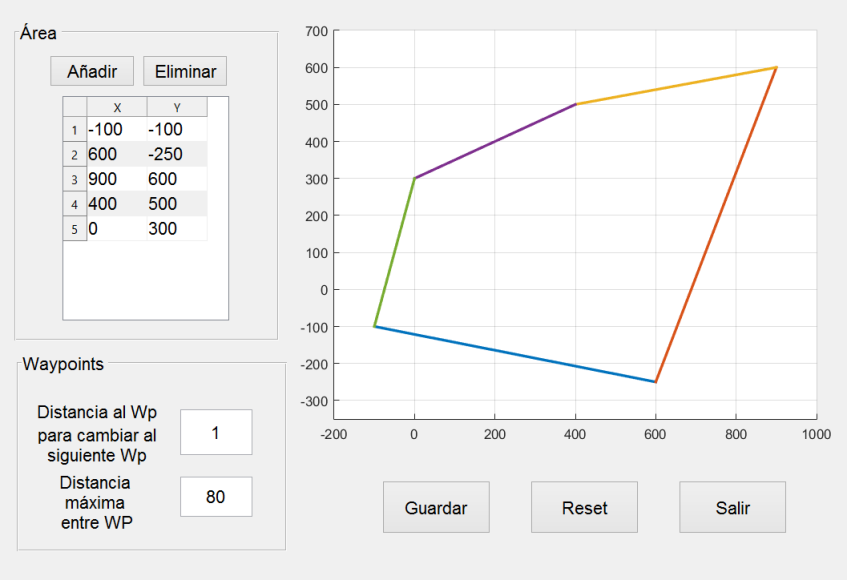

Figura 6.13 Interfaz de los datos del área.

6.4 Resultados del programa

Por último se va a ver que resultados arroja este programa y si son fiables o no. Para ello, se van a ejecutar distintas simulaciones y se va a comprobar que son coherentes con lo que debería pasar en la realidad.

Se van a dividir estas simulaciones en tres casos diferentes, variando el área de barrido, el número de UAVs que realizan el barrido y variando la apertura del sensor de la cámara de cada UAV. Éstos tres casos son los mismos que se realizaron en el programa uno (capítulo 4).

Primero, se van a describir las características generales de todas las simulaciones y en cada simulación se describirá cada variación hecha respecto a las características generales que aquí se presentan:

- Todos los UAVs de la simulación vuelan a 100 metros de altura, a 20 metros por segundo, con un campo de visión de la cámara de 60 grados y con un radio mínimo de giro de 40 metros. Al volar todos a la misma altura va a haber colisiones entre los UAVs, pero esto es una simulación y se ha optado por tomar esta decisión. Ésto no

es real, pero bastaría con cambiar la altura de los UAVs para evitar colisiones o bien, ponerles un controlador eficiente para evitar las colisiones entre ellos.

- Las características aerodinámicas y estructurales usadas para los UAVs son:
  - C_{l_0}: 0,28
  - C_{d_0}: 0,03
  - $MTOW$: 13,5 kg.
  - γ_s: 6°
  - V_c: 20 m/s.
  - C_{l_α}: 3,45
  - C_{d_α}: 0,3
  - S: 0,55 m^2.
  - γ_d: 6°
  - V_s: 9,72 m/s.
- Las constantes generales usadas en el programa son:
  - g: 9,8066
  - γ: 1,4
  - R: 287
  - μ_{pista}: 0,02
- El tiempo de salida al despegar entre cada UAV es de 20 segundos.
- La base de aterrizaje se encuentra en las coordenadas [-2,-2] kilómetros, a una altitud de nivel del mar y una orientación de la pista de 45°, es decir, dirección noreste.
- La distancia a la que tiene que estar el UAV del waypoint para cambiar al siguiente waypoint es de un metro.
- La distancia máxima entre waypoints de una misma hilera es de 80 metros.
- El área a barrer viene definida por las coordenadas de cada uno de los vértices que componen el área. Ésas coordenadas se muestran en la tabla 6.1. La figura 6.14 se corresponde con el área que se tiene que barrer.

Tabla 6.1 Coordenadas de los vértices del área.

	Coordenada x	Coordenada y
Vértice 1	0	0
Vértice 2	600	-250
Vértice 3	1200	50
Vértice 4	1300	450
Vértice 5	1000	725
Vértice 6	300	700
Vértice 7	-150	400

- Para el caso en los que se varíe el número de UAVs y el campo de visión de la cámara que llevan montada los UAVs se van a suponer dos casos distintos, uno con el área anteriormente descrita y otro con un área superior, para ello se ha multiplicado por tres la coordenada x e y de cada vértice del polígono. La tabla 6.2 se corresponde con los valores de las coordenadas de los vértices del polígono.

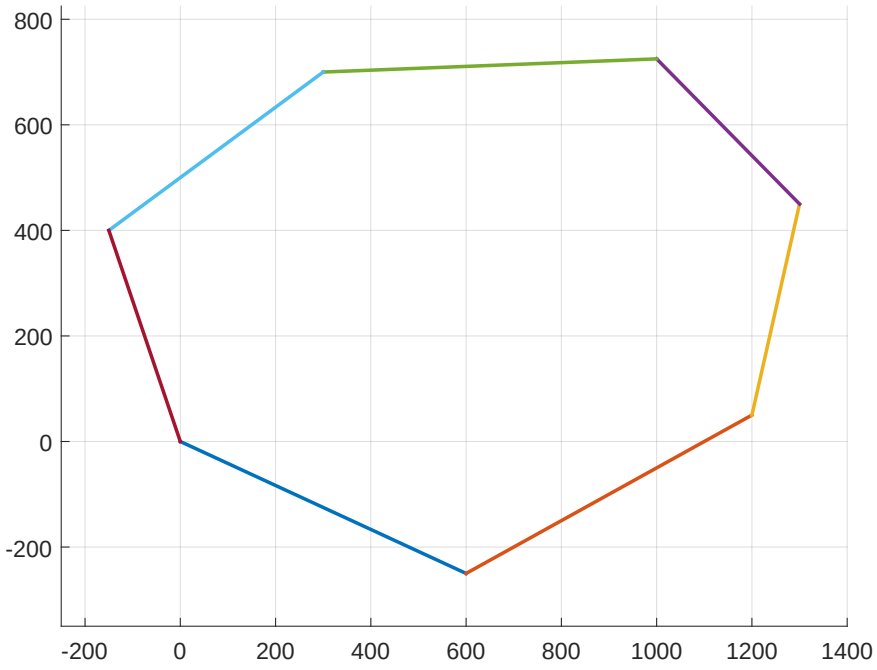

Figura 6.14 Área a barrer.

Tabla 6.2 Coordenadas de los vértices del área en el caso dos.

	Coordenada x	Coordenada y
Vértice 1	0	0
Vértice 2	1800	-750
Vértice 3	3600	150
Vértice 4	3900	1350
Vértice 5	3000	2175
Vértice 6	900	2100
Vértice 7	-450	1200

Ya están vistas todas las características de las simulaciones realizadas. Se va a empezar explicando las simulaciones cuando se varía el área de barrido.

En este programa al tiempo de barrido se le llama al tiempo que se necesita para despegar los UAVs, barrer el área y volver a aterrizar todos los UAVs de la simulación.

6.4.1 Área de barrido

En este caso se toma como área principal el área que define el polígono de la tabla 6.1, a partir de éste área se va a ir aumentando multiplicando las coordenadas de los vértices del polígono por un número. Ésto no implica que el área aumente en dicho número, de hecho, ésto no va a ser así.

Se utilizan tres UAVs con las características descritas anteriormente.

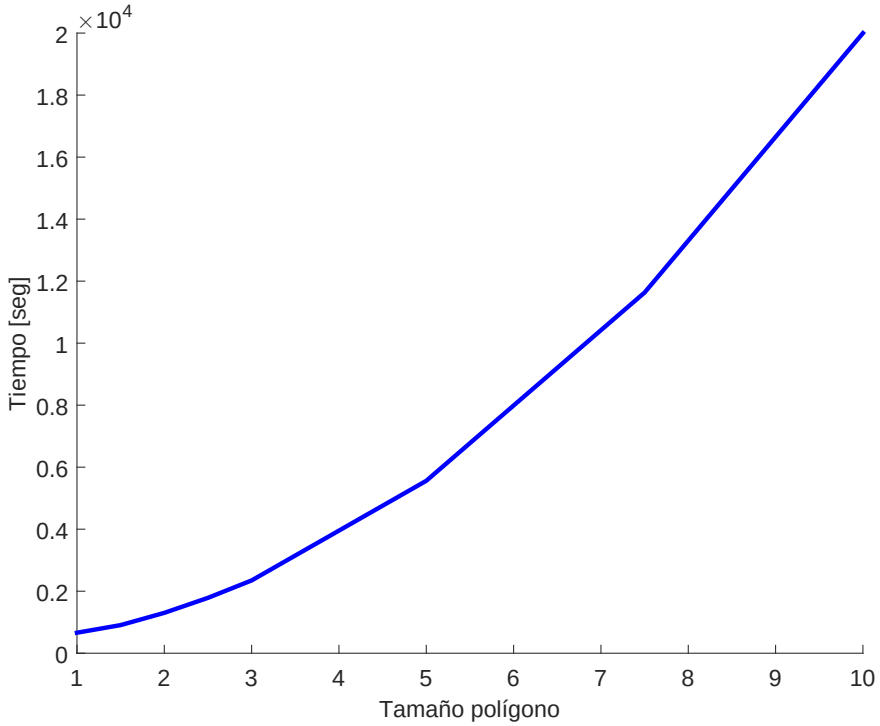

Figura 6.15 Tamaño del área frente al tiempo de simulación.

La figura 6.15 es la correspondiente a esta simulación y en ella se aprecia que conforme se aumenta el área del polígono el tiempo de barrido aumenta lo cual es lógico. Se ve que a diferencia del programa uno (figura 4.11), en la cual, el tiempo de barrido crecía de forma lineal, en este caso el tiempo de barrido crece de forma exponencial. Ésto se debe a que en la figura 4.11 del programa uno se pinta el área respecto del tiempo de barrido, mientras que aquí en el eje x no está pintada el área sino el crecimiento del polígono que no es lineal con el área sino potencial[1].

6.4.2 Número de UAVs

En esta segunda simulación se han hecho dos casos diferentes, en los cuales varía el área utilizada en cada caso. En esta simulación se ha ido aumentando el número de UAVs desde un UAV hasta diez UAVs, dando lugar a dos gráficas según el tipo de área que se tienen que barrer. La figura 6.16 que se refiere al caso del área más pequeña y la figura 6.17 que se refiere al área más grande.

De la figura 6.16 se aprecia que conforme se aumenta el número de UAVs se disminuye notablemente el tiempo de barrido, pero a partir de tres UAVs ya no se disminuye el tiempo de barrido, es más, el tiempo de la simulación aumenta poco a poco. Esto es debido a que el área es demasiado pequeña para tantos UAVs, es decir, al haber una sola pista de aterrizaje y despegue, los UAVs tienen que despegar y aterrizar de uno en uno, lo que provoca que si

[1] La explicación de esta afirmación es simple. Por ejemplo, un cuadrado de longitud uno tiene área unidad, si a éste cuadrado se le multiplica a cada vértice del polígono que lo define por un número, por ejemplo tres, se obtiene un cuadrado de longitud tres, pero el área de este cuadrado es nueve y no tres. Lo que demuestra que el área crece de forma potencial.

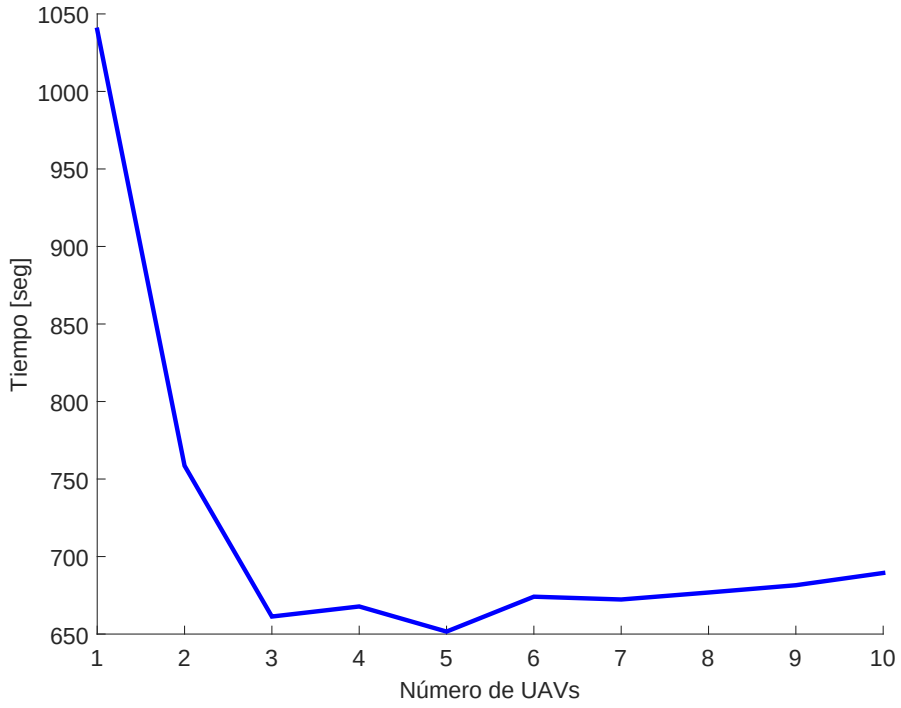

Figura 6.16 Número de UAVs frente al tiempo de simulación en el caso uno.

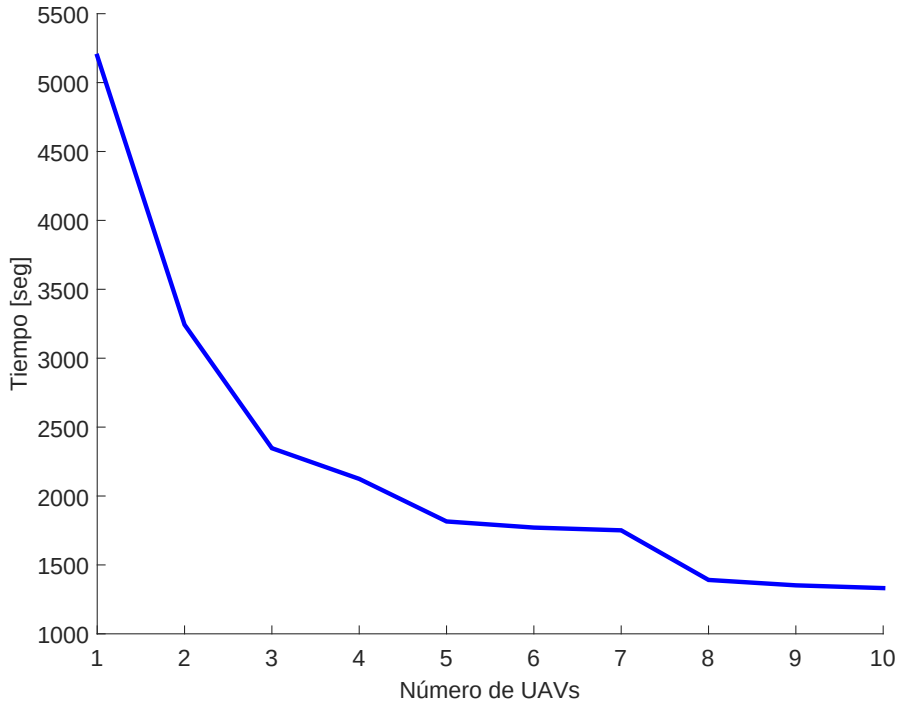

Figura 6.17 Número de UAVs frente al tiempo de simulación en el caso dos.

hay muchos UAVs puede haber mucho tráfico a la hora de aterrizar, por lo que tienen que perder tiempo dando vueltas mientras aterrizan el resto de los UAVs.

En el segundo caso, figura 6.17, ya no se tiene el problema de haber demasiado tráfico a la hora de aterrizar porque el área que se tiene que barrer es lo suficientemente grande como para que no se pisen los UAVs entre ellos. Esto provoca que el tiempo de barrido pueda ir descendiendo según el número de UAVs que se necesite.

De estas figuras obtenidas se pueden obtener muchas más conclusiones en relación al tipo de uso que se le quiera dar al programa. A continuación se presentan algunas de ellas:

- Obtenidas éstas gráficas se puede saber cuantos UAVs hay que usar en la realidad para realizar una misión determinada, por ejemplo:
  - Para un uso de búsqueda y rescate lo importante es que el tiempo sea el menor posible, por lo que interesaría usar un mínimo de tres UAVs en el primer caso y ocho UAVs para el segundo caso.
  - En una misión de búsqueda de incendio, no hay tanta prisa para barrer el área, por lo que no sería necesario el uso de tantos UAVs, permitiendo el uso de tan sólo dos UAVs para el primer caso y tres UAVs en el segundo caso.
  - Si fuera una misión de ocio o simplemente de barrido de cultivos no sería nada importante el tiempo de barrido y sí importaría el coste de usar los UAVs, intentando reducir los costes al mínimo, por lo que en el primer caso con un UAV sería suficiente y en el segundo caso también con un UAV sería suficiente.
- Se puede obtener, sabiendo la autonomía del UAV simulado, si es posible realizar una misión determinada, por ejemplo, en el segundo caso, no es posible barrer el área entera si sólo se tiene un UAV y la autonomía del modelo de UAV es inferior a 5.000 segundos.

6.4.3 Apertura del sensor

Al igual que en el caso del barrido del área, se han utilizado tres UAVs para barrer el área, tanto para el caso uno como para el dos. En esta simulación se ha ido variando el campo de visión de la cámara desde 50° hasta 120°. Menos de 50° no es posible simular porque el UAV introducido no tiene la capacidad necesaria para realizar un giro tan cerrado al final de cada hilera de waypoints a una altura de 100 metros. Si se disminuyera la altura sí se podría disminuir más el campo de visión.

En la figura 6.18 se aprecia que el tiempo de barrido disminuye cuanto mayor es el campo de visión de la cámara de los UAVs, pero al igual que en la simulación anterior, en el caso uno que se corresponde al área pequeña, a partir de un número de UAVs no disminuye el tiempo de barrido. Ésto se debe al igual que el caso anterior, a que los UAVs llegan al mismo tiempo a la zona de aterrizaje y pierden ahí el tiempo ganado durante el barrido del área.

En la figura 6.19 se observa que al igual que para el caso uno, el tiempo de barrido disminuye conforme se aumenta el campo de visión de la cámara. En este caso, al ser el área lo suficientemente grande como para que no se pisen los UAVs entre ellos en la zona de aterrizaje, no se ve que el tiempo se estanque en ningún campo de visión y se aprecia que disminuye de forma lineal. Por lo tanto, coincide con la figura 4.13 del programa uno, en la cual también disminuía de forma lineal.

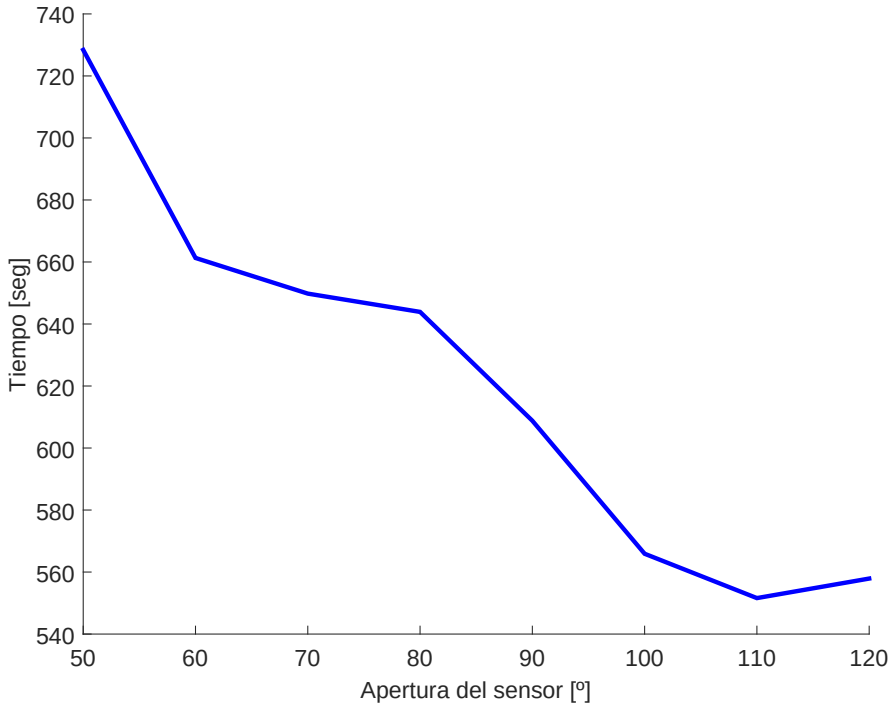

Figura 6.18 Apertura del sensor frente al tiempo de simulación en el caso uno.

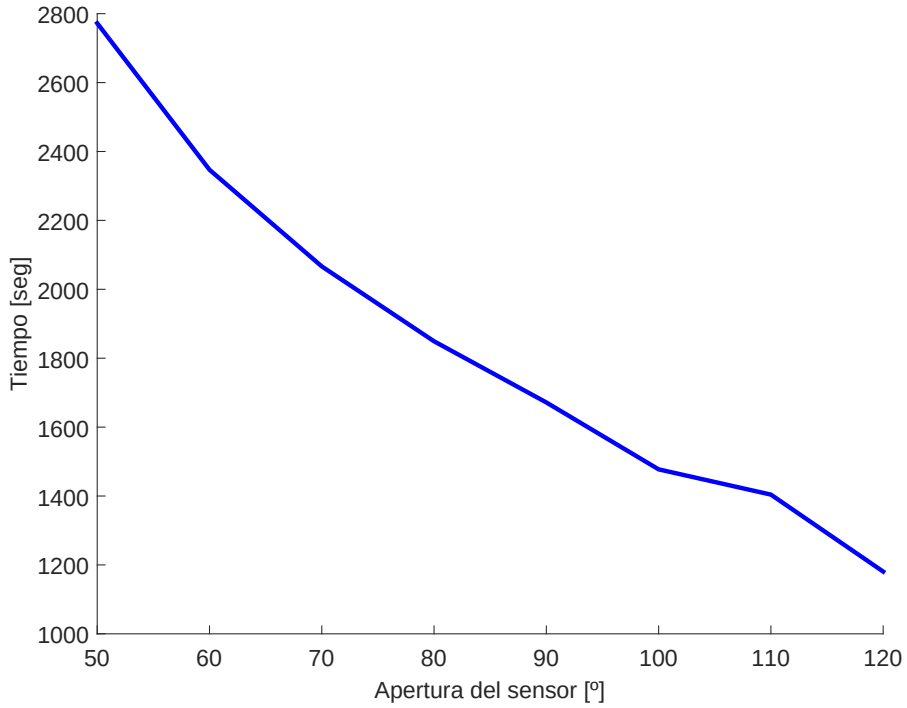

Figura 6.19 Apertura del sensor frente al tiempo de simulación en el caso dos.

Al igual que para el caso anterior de variación de UAVs, se pueden obtener distintas conclusiones en relación a la misión que se vaya a realizar. Hay que saber que un elevado campo de visión permite ver mucho terreno, pero a una mala calidad de las imágenes, sin embargo, un campo de visión pequeño, permite fotografías de un alto nivel de detalle.

- Para una misión de fotogrametría, se necesita una alta definición de las imágenes, por lo que se necesita un bajo campo de visión de la cámara, lo que lleva asociado un mayor tiempo de barrido para un mismo número de UAVs en uso.
- Para una misión de búsqueda y rescate, se necesita una también una alta calidad de las imágenes, pero dependiendo del objeto a buscar, es decir, no es lo mismo buscar a una persona que una casa en un campo. Se suele necesitar menos calidad que para la fotogrametría pero aún así se necesita mucha, por lo que también se necesita un campo de visión pequeño.
- En una misión de búsqueda de incendios, no importa demasiado la calidad de las imágenes porque el fuego se suele propagar rápidamente ocupando un espacio grande en un corto período de tiempo. Es por eso, que no se requiere un campo de visión reducido y se puede utilizar un mayor campo de visión reduciendo el tiempo de barrido.
- En una misión de ocio o de información sobre los cultivos, no es necesario un alto nivel de detalle, por lo que el campo de visión puede ser muy grande llevando consigo pequeños tiempos de barrido respecto a las otras misiones.

En todas éstas misiones hay que consultar los datos con cuidado porque si el tiempo de barrido es superior a la autonomía de los UAVs no se podrá realizar la misión y habrá que, o bien aumentar el número de UAVs, o bien, aumentar el campo de visión de la cámara de los UAVs.

7 Conclusiones y trabajos futuros

Este trabajo ha desarrollado en tres programas diferentes, distintas situaciones para ver el comportamiento de los UAVs. Lo que ha permitido llegar a las siguientes afirmaciones:

- Conforme se aumenta el número de UAVs disponibles para barrer un área disminuye el tiempo de barrido hasta un valor mínimo. Dicho valor aparece en áreas pequeñas y porque al aumentar el número de UAVs disponibles no mejora el tiempo de barrido.
- Cuanto mayor es el área a barrer para un mismo número de UAVs disponibles, aumento el tiempo de barrido.
- Al aumentar el campo de visión (FOV) disminuye el tiempo de barrido, pero disminuye también la calidad de las imágenes de las cámaras, y viceversa.
- En el programa tres, hay un estudio sobre qué campo de visión debe tener la cámara para barrer un área según el tipo de misión que se esté llevando a cabo, diferenciando entre una misión de fotogrametría, de búsqueda y rescate, búsqueda de incendios y una misión de observación de cultivos o ocio. Este estudio revela que la fotogrametría requiere un mayor tiempo de barrido debido a su bajo valor del campo de visión para obtener unas imágenes de muy alta calidad. La observación de cultivos y búsqueda de incendios no requiere una alta calidad de las imágenes porque el incendio se expande rápidamente y es visible fácilmente, al igual que los cultivos que tampoco requieren un alto nivel de detalle lo que favorece que ante estas misiones, el tiempo de barrido sea mínimo. Por último, la misión de búsqueda y rescate requiere un bajo valor del campo de visión para obtener las imágenes de calidad suficiente para poder diferenciar los objetivos de la misión.
- En el programa dos se sacan conclusiones similares a los otros dos programas, pero en este caso el tiempo de barrido necesario hasta encontrar los objetivos puestos varía mucho en función de donde se coloquen los objetivos, porque los conjuntos de UAVs empiezan a barrer por un lado concreto del mapa (las esquinas), por lo que si se ponen los objetivos cerca de las esquinas el tiempo de barrido será inferior que si se colocan cercanos al centro.

Una vez desarrollado los programas de este trabajo, se pueden aplicar para futuros estudios diferentes consecuencias que se han obtenido de este trabajo.

- En este trabajo, se han realizado tres programas diferentes para tres situaciones diferentes, pero se podrían estudiar los tres programas como si fueran uno solo. De tal forma que exceptuando el despegue y aterrizaje de UAVs, se pueden combinar los tres programas en uno solo y ver qué nuevos resultados se obtienen del mismo.
- Se podría modificar la altura de los UAVs en vuelo durante la simulación en alguna zona que sea de un mayor interés, con el fin de bajar la altura en zonas más interesantes para tomar mejores datos y en zonas que no sean relevantes, subir la altura para tomar más datos en menos tiempo.
- Se podría intentar unir estos esfuerzos aéreos en barrer un área con esfuerzos terrestres para mejorar la información y la rapidez del barrido.

Apéndice A
Aerosonde UAV

Para la realización de este trabajo se han utilizado los datos del UAV *Aerosonde* que se han encontrado en [3, pág. 276]. Dichos datos se encuentran en la tabla A.1.

Figura A.1 UAV Aerosonde.

Aparte de los datos aerodinámicos, las especificaciones del UAV son:

- Longitud: 1,7 m
- Altura: 0,6 m
- Peso en vacío: 10 kg
- Planta motora: Motor Enya R120 modificado. 1,74 hp (1280 W).
- Máxima velocidad: 72 km/h
- Alcance: 150 km
- Techo de vuelo: 4.500 m

Tabla A.1 Coeficientes aerodinámicos del UAV Aerosonde.

Parámetro	Valor	Longitudinal		Lateral	
		Coef.	Valor	Coef.	Valor
m	13,5 kg	C_{L_0}	0,28	C_{Y_0}	0
J_x	0,8244 kg· m^2	C_{D_0}	0,03	C_{l_0}	0
J_y	1,135 kg· m^2	C_{m_0}	-0,02338	C_{n_0}	0
J_z	1,759 kg· m^2	C_{L_α}	3,45	C_{Y_β}	-0,98
J_{xz}	0,1204 kg· m^2	C_{D_α}	0,30	C_{l_β}	-0,12
S	0,55 m^2	C_{m_α}	-0,38	C_{n_β}	0,25
b	2,8956 m	C_{L_q}	0	C_{Y_p}	0
c	0,18994 m	C_{D_q}	0	C_{l_p}	-0,26
S_{prop}	0,2027 m^2	C_{m_q}	-3,6	C_{n_p}	0,022
ρ	1,2682 kg/m^3	$C_{L_{\delta_e}}$	-0,36	C_{Y_r}	0
k_{motor}	80	$C_{D_{\delta_e}}$	0	C_{l_r}	0,14
k_{T_p}	0	$C_{m_{\delta_e}}$	-0,5	C_{n_r}	-0,35
k_Ω	0	C_{prop}	1,0	$C_{Y_{\delta_a}}$	0
e	0,9	M	50	$C_{l_{\delta_a}}$	0,08
		α_0	0,4712	$C_{n_{\delta_a}}$	0,06
		ε	0,1592	$C_{Y_{\delta_r}}$	-0,17
		C_{D_p}	0,0437	$C_{l_{\delta_r}}$	0,105
				$C_{n_{\delta_r}}$	-0,032

Figura A.2 UAV Aerosonde.

Apéndice B
Cámara gimbal

Un gimbal es una plataforma motorizada y controlada mediante una placa con sensores varios, generalmente acelerómetros y compás magnético que se encargan mediante el uso de algoritmos de control y PIDs de mantener un objeto, normalmente una cámara estabilizada, de modo que independientemente del movimiento que realice el portador de la misma, ésta quede estable permitiendo tomar buenas capturas.

Figura B.1 UAV Aerosonde con gimbal I.

Los gimbals más comunes son los de dos ejes, aunque también se ven pero son menos usados los de tres ejes. En la figura B.1 se aprecia este sistema montado en el UAV *Aerosonde*.

Con dos ejes tendremos bloqueados la mayoría de movimientos de "roll" y "pitch" pudiendo incluso mediante la asignación de un canal a la emisora configurar fácilmente estos para establecer nuevas posiciones fijas para la cámara, quedando de este modo perfecta para cambiar de plano.

Si este aparato se le coloca a un UAV, permite que la cámara siempre mire hacia el terreno sin importar la actitud del mismo. Lo que hace que el controlador no se preocupe

de tener siempre el objetivo de la cámara mirando hacia el suelo y se centre en dirigir a la aeronave.

Figura B.2 UAV Aerosonde con gimbal II.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Índice de Algoritmos

Bibliografía

[1] *Historia de los Drones I*, *http:// mundrone.blogspot.com.es*.

[2] *La Historia de los Drones*, *http:// eldrone.es*, Mayo 2016.

[3] Randal W. Beard and Timothy W. McLain, *Small Unmanned Aircraft: Theory and Practice*, Princeton University Press, 2012.

[4] Alberto Lobit Cerrato, *Control en Vuelo de Formación Cerrada de UAVs de Ala Fija*, Universidad de Sevilla, 2014.

[5] Sergio Esteban, Antonio Franco, and Alfonso Valenzuela, *Modelos Propulsivos Genéricos*, 2013.

[6] Geroge Evers, *Particle Swarm Optimization Research Toolbox Documentation*.

[7] Natalie R. Frantz, *Swarm Intelligence for Autonomous UAV Control*, Naval Postgraduate School, Monterey; California, Junio 2005.

[8] Guillermo Heredía, Fernando Caballero, Iván Maza, Luis Merino, Antidio Viguria, and Aníbal Ollero, *Multi-Unmanned Aerial Vehicle (UAV) Cooperative Fault Detection Employing Differential Global Positioning (DGPS), Inertial and Vision Sensors*, Septiembre 2009.

[9] Luis R. Izquierdo, *Introducción a la Programación Orientada a Objetos*.

[10] Robert J. Bamberger Jr., David P. Watson, David H. Scheidt, and Kevin L. Moore, *Flight Demonstrations of Unmanned Aerial Vehicle Swarming Concepts*, 2006.

[11] Simon Lacroiz, Rachid Alami, Thomas Lemaire, Gautier Hattenberger, and Jérémi Gancet, *Decision Making in Multi-UAVs Systems: Architecture and Algorithms*.

[12] Ivan Maza, Aníbal Ollero, Enrique Casado, and David Scarlatti, *Classification of Multi-UAV Architectures*.

[13] Laurence R. Newcome, *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles*, American Institute of Aeronautics and Astronautics (AIAA), 2004.

[14] Aníbal Ollero, *Introducción a la Robótica*, Apuntes de clase: Robótica Espacial, 2016.

[15] Damián Rivas Rivas, *Mecánica del Vuelo I*, Universidad de Sevilla, 2012.

[16] Ivan Maza y Aníbal Ollero, *Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms*, Enero 2007.