Trabajo SMA*

Jorge Juan González y José Javier Calvo Moratilla

Universitat Politècnica de València

Abstract. Estudio académico de "A fault-tolerant self-organizing flocking approach for UAV aerial survey". Propuesta implementación en Netlogo.

Keywords: Multiagentes · Flocking · UAV.

1 Introducción

En el presente trabajo se describe el estudio académico titulado *"A fault-tolerant self-organizing flocking approach for UAV aerial survey"*, en el que se describe una técnica de "flocking" para el control autónomo de drones *UAV* multirotor, cuya principal misión es la vigilancia de un área concreta de terreno.

La organización de los drones se realiza mediante el uso de algoritmos, donde se tienen en cuenta varios aspectos: las distancias mutuas entre unidades motorizadas para poder cubrir la mayor superficie de terreno posible, la elección de un líder que marca el posicionamiento del grupo para agrupar inteligentemente los drones y la tolerancia a fallos para sustituir las unidades que dejan de funcionar, todo en tiempo real.

Los drones realizan fotografías en el terreno desde el aire, auto-organizándose, tomando la naturaleza como referencia, emulando el movimiento de las bandadas de pájaros sobrevolando y agrupándolos en el cielo.

La tarea de vigilancia actualmente es realizada por aviones o helicópteros, algo que implica una inversión económica bastante significativa, por tiempo y capital humano invertido, sumado a otros inconvenientes derivados de la tecnología, como la limitación de las baterías o el peso de los *UAV*, dichos retos también se abordan en el estudio, con la motivación de encontrar una solución viable y efectiva.

El estudio académico también pretende simular mediante software el comportamiento de los drones en su vuelo grupal, para poder seleccionar los parámetros de los algoritmos y así ajustarlos para poder estudiar el comportamiento de los drones.

* Supported by Universitat Politècnica de València.

Después de haber introducido brevemente el tema en cuestión se enumeran las diferentes partes del trabajo:

En primer lugar se enumeran todos los trabajos relacionados con el estudio académico para entrar en contexto con la materia.

En segundo lugar se plantean los objetivos a lograr y se describe el entorno sobre el que van a funcionar los *UAV* durante la simulación, así como sus restricciones.

En tercer lugar se trata la implementación del artículo propuesta por los autores del mismo.

En cuarto lugar se describen los experimentos llevados a cabo en el artículo y sus resultados.

En quinto y último lugar, una vez realizado el análisis de todos los puntos importantes se procede a implementar una aproximación mediante el software *NetLogo*, software utilizado en los laboratorios de la asignatura de Sistemas Multiagente, simulando el comportamiento organizativo de los drones (UAV).

2 Trabajos relacionados

Conocer los trabajos previos, actuales y posteriores relacionados con el artículo nos ayuda a entender en mayor profundidad el contexto del artículo que describimos, por ello en los siguientes puntos se recogen los artículos relacionados que más han influido en el presente trabajo y el estado de la cuestión.

2.1 Trabajos previos

Dentro de los trabajos previos se entiende que cada dron corresponde a un agente inteligente y que tiene el poder de decidir que hacer para poder cumplir sus objetivos, en dicho sentido el primer trabajo trata sobre la formación y mantenimiento de las bandadas, las comunicaciones entre entidades y los algoritmos para la cobertura de superficies aéreas, situados en el ámbito de los sistema multiagente de Van der Hoek y Wooldridge en el año 2008 [2]. Los vehículos aéreos no tripulados entran dentro del marco de agentes inteligentes autónomos, que tienen la capacidad de comunicarse entre ellos.

Para poder ajustar los agentes de manera correcta se necesitan hacer pruebas, por ello la herramienta base para realizar la simulación corresponde con la aplicación Netlogo. Contar con herramientas para simulación permite observar el comportamiento de los agentes poniendo en práctica las estrategias definidas.

En dicho sentido el segundo marco de referencia para el artículo es la propuesta de simulación de un enjambre de vehículos aéreos no tripulados mediante el uso del software utilizado por la asignatura de NetLogo por Wilensky y Rand en el año 2015 [3], dónde se definen diferentes reglas que controlan el comportamiento del conjunto del enjambre.

Una vez claro el comportamiento que deben de seguir los drones no tripulados se deben de crear algoritmos que permitan a los vehículos aéreos planificar las misiones que deben de realizar y qué decisiones tomar cuando el entorno cambia. Por ello como tercer artículo de referencia se observa el trabajo realizado por Yi Wei [4] en el que se propone una planificación dinámica de misiones para UAVs a través de un marco de control híbrido, centralizado y distribuido para la programación de misiones concretas, con alto poder de adaptación a un entorno natural variable.

Teniendo controlado el comportamiento de los drones desde una posición de referencia ahora se precisa de alguna estrategia para que las unidades no tripuladas puedan posicionarse en lugar más óptimo, para la maximización del objetivo grupal, por ello en el cuarto artículo de referencia se observa el trabajo de Sekin, Ahmet ada del año 2016 [5] en el que se plantea un algoritmo de posicionamiento distribuido de los UAVs, calculados mediante las imágenes que se intercambian los agentes.

Para poder fotografiar una superficie de terreno concreta, los drones precisan de organización para maximizar su objetivo, en dicho caso en el artículo de Fazli del año 2013 [6] se realiza una aproximación a la cobertura de áreas de una superficie, utilizando parámetros como el tiempo, la distancia, el número de visitas para cada punto de interés, con una minimización del tiempo de la misión total de los agentes.

Por último el planteamiento de la superficie como un campo de celdas se obtiene del enfoque de Latombe en el año 1991 [7] y de Rekleitis [8] en el 2008 dónde se propone la descomposición del trabajo global en tareas para asignar a cada agente, planteando el terreno como una una división en celdas similares en tamaño y forma.

2.2 Estado del arte

Habiendo hecho un análisis exhaustivo de los trabajos previos, se identifican los artículos más importantes publicados posteriormente al artículo descrito en el presente trabajo, dónde se identifican nuevas aproximaciones relacionadas con el campo del aprendizaje automático, y otras técnicas que han revolucionado el mundo de la ciencia y de la empresa.

Uno de los campos más importantes actuales es el aprendizaje automático, concretamente en el paradigma del aprendizaje por refuerzo, donde los agentes tienen la habilidad de aprender sin la programación explícita del programador, por ello es una avance muy significativo respecto al presente trabajo, ya que el comportamiento de los vehículos lo determinan unas reglas, previamente definidas por el ser humano.

El primer ejemplo se engloba dentro del campo del aprendizaje automático, concretamente en el campo del aprendizaje por refuerzo llamado "Trajectory Design and Power Control for Multi-UAV Assisted Wireless Networks: A Machine Learning Approach" [9] de Xiao Liu, Yuanwei Liu, Yue Chen y Lajos Hanzo, en el que se propone que los UAV observen su entorno y aprendan de los errores por sí mismos realizando predicciones de cómo va a variar dicho entorno, pudiendo adaptarse a él sin la necesidad de la interacción implícita del ser humano.

Siguiendo en el mismo marco teórico se describe el paper "Flocking Control of UAV Swarms with Deep Reinforcement Learning Approach" [11] de Peng Yan, Chengchao Bai, Hongxing Sheng y Jileng Guo, dónde se propone una aproximación basada en el aprendizaje por refuerzo profundo que soluciona el problema de utilizar un Partially Observable Markov Decision Process (POMDP), cuando existen restricciones de los UAvs que hacen referencia a la comunicación entre unidades autónomas y la percepción de las mismas. Las unidades autónomas pueden modificar su comportamiento en base a un entrenamiento del UAVs local y la posibilidad de poder compartir dicho conocimiento al resto. Dicha aproximación permite que los drones puedan evitar obstáculos ya existentes o nuevos en la naturaleza.

Por último, a parte de los avances realizados dentro del campo del aprendizaje automático también se observan avances en el campo de *Flocking*, en dicho sentido el segundo "Using Lazy Agents to Improve the Flocking Efficiency of Multiple UAVs" [10] de Yeongho Song, Myeonggeun Gu, Joonwon Choi, Hyondong Oh, Seunghan Lim, Hyo-Sang Shin y Antonios Tsourdos, propone crear una bandada de UAVs utilizando el modelo Cucker-Smale (C-S) aumentado [12], que alinea a los vehículos aéreos no tripulados de manera autónoma, manteniendo la distancia de los agentes de manera distribuida, cubriendo el mayor número de espacio posible.

3 Planteamiento

El objetivo del sistema será monitorizar un área en el menor tiempo posible minimizando sobre-cobertura, siendo el área una superficie plana situada en exteriores.

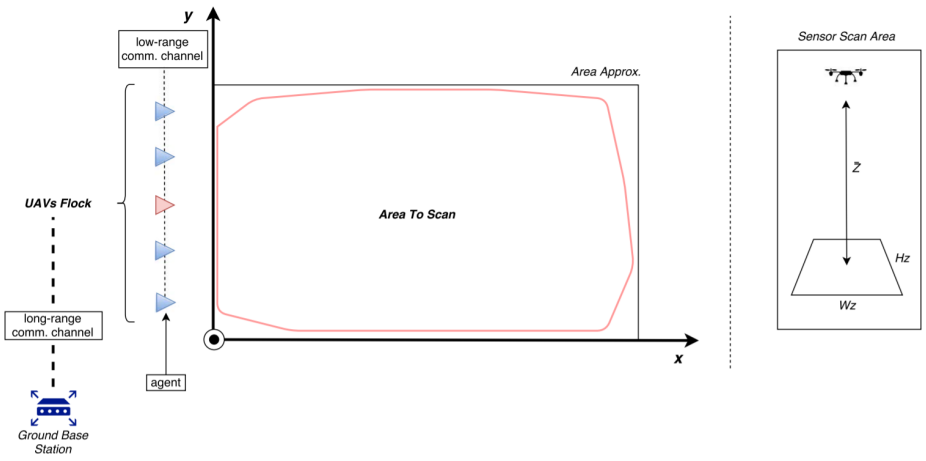
Este objetivo se deberá cumplir en un entorno sometido a las restricciones propias

de un entorno real como son la posibilidad de que los UAVs puedan fallar con una cierta probabilidad o debido colisiones con otros UAVs, la latencia de una transmisión o la frecuencia con la que se pierden paquetes durante la misma.

4 Implementación

El sistema descrito en el artículo [1] se divide en tres entidades o miembros fundamentales: área a monitorizar, estación base y agentes.
En primer lugar, el área a monitorizar puede ser irregular pero en ese caso se toma el rectángulo de área mínima que pueda contenerla.
Por otro lado, la estación base permanecerá en el suelo y recibirá periódicamente, por parte del agente líder, el mapa de regiones monitorizadas o fotografiadas hasta el momento a modo de backup por si los drones fallasen.
En cuanto a los agentes, se simulan como drones cuadricópteros de despegue y aterrizaje vertical (VTOL) con un ID único y equipados con una cámara con la que tomar las fotos del suelo y 2 sistemas de comunicación: uno de corto alcance y bajo consumo para comunicación frecuente entre agentes, y otro de largo alcance y alto consumo para comunicación puntual con la estación base.
En la simulación, los drones usan geoposicionamiento por GPS por lo que el sistema de referencia de un agente será sus coordenadas geográficas (latitud y longitud) y su heading o rotación. Los drones se moverán a la misma altura respecto al suelo.
La siguiente figura muestra la arquitectura y el escenario planteado por los autores.

Fig. 1. Arquitectura y escenario.



Para poder cumplir el objetivo de este trabajo, los agentes, operando de forma autónoma, deberán cooperar para:

- Asegurar que la estación base recibe lecturas de toda el área.
- Minimizar el tiempo de la misión.
- Evitar/minimizar la capturas solapadas (sobre-cobertura).

Cada agente tendrá dos bases de datos que irá actualizando con la información que recibe por mensajes de otros drones:

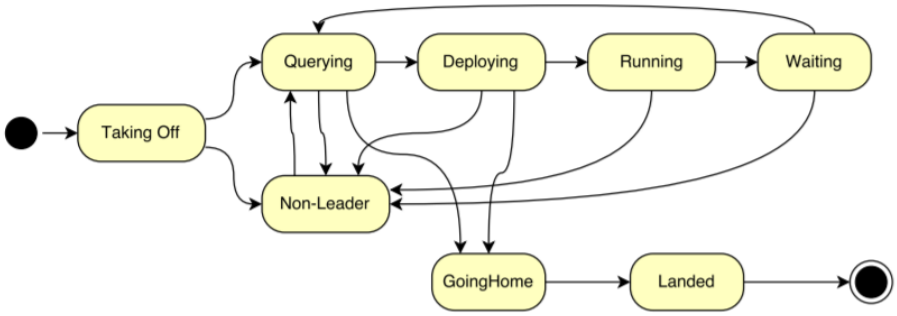
- Base de Datos de Agentes (ADB): Registro del estado (posición, orientación, antigüedad de los datos, etc.) de cada agente del sistema.
- Base de Datos de Partes del Área (APD): Registro las regiones del área que han sido capturadas por al menos un agente.

Durante la simulación, cada agente ejecutará en bucle las siguientes tareas:

1. Gossiping: Informará de su estado a los agentes de su entorno y retransmitirá los mensajes que haya recibido de otros agentes si estos están actualizados.
2. Actualiza ADB: Con la información recibida en el paso anterior actualizará su ADB donde almacena internamente el estado del resto de agentes.
3. Flocking: Conociendo el estado del resto de agentes, calcula el siguiente movimiento que debe realizar haciendo uso de su algoritmo de flocking.
4. Actualiza APD: Toma una imagen del terreno (si esa región no aparece dentro de la APD), recibe los mapas de regiones capturadas de otros agentes, actualiza el mapa interno de regiones totales capturadas y si hay alguna zona capturada que desconozcan los vecinos se transmite la nueva región capturada a los vecinos.
5. Actualiza ruta (sólo líder): Conociendo las regiones que aún no han sido capturadas, el agente líder determina la dirección óptima hacia la que avanzar.

Los autores también explican la máquina de estados finitos que dirige el comportamiento de un agente. La siguiente figura muestra los distintos estados a los que puede transicionar un agente. El estado “Querying” indica que se está esperando a recibir mensajes de todos los agentes para decidir quién debe ser el líder. El estado “Deploying” significa que el agente es líder y está esperando a que el resto de agentes se terminen de posicionar en la línea de formación. El estado “Running” se refiere al estado en el que el líder está en vuelo monitorizando el área junto al resto de agentes. El estado “Waiting” indica que el líder ha terminado una pasada de la captura y está esperando que el resto de agentes terminen.

Fig. 2. Máquina de estados de un UAV.



4.1 Flocking

Cada agente, una vez actualizado su conocimiento sobre el estado del resto de agentes (ADB), calculará la dirección hacia la que avanzar de forma que se cumplan lo mejor posible una serie de reglas. Por lo general, un algoritmo de flocking aplica las siguientes reglas:

1. R1-Separación: Los agentes deben mantenerse separados al menos a una determinada distancia mínima para evitar colisiones.
2. R2-Alineamiento: Los agentes deben tener la misma la dirección que sus vecinos.
3. R3-Cohesión: Los agentes deben permanecer agrupados y cerca los unos de los otros.

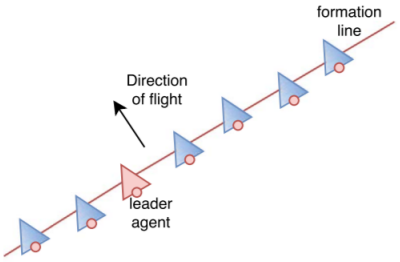
En este trabajo, además de las reglas expuestas anteriormente también se establecen restricciones:

1. Minimizar sobre-cobertura longitudinal: Si un drone volase detrás de otro, estaría tomando las mismas capturas que el primer drone, malgastando tiempo y recursos. Para evitar capturar muchas veces una misma zona del terreno los autores de este trabajo proponen que los agentes mantengan una formación lineal a ambos lados del líder.
2. Minimizar sobre-cobertura horizontal: Si los drones vuelan en formación lineal pero demasiado cercanos unos de otros, además de aumentar el riesgo de colisión, hace más probable que las imágenes tomadas se solapen entre ellas. Los autores proponen ajustar los parámetros de la regla R1 para que los drones vuelven a la distancia adecuada para que este solapamiento sea mínimo.
3. Evitar huecos: Esta restricción se puede satisfacer con la regla R3, la cual evita precisamente que los drones vuelen dispersos.

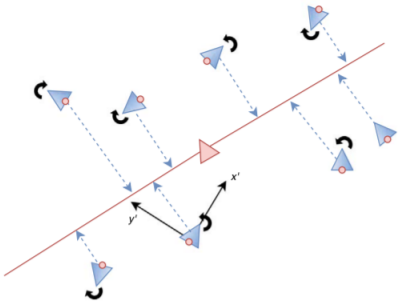
En el algoritmo propuesto por los autores, se elige como agente líder aquel que tenga el menor ID. Si un drone líder falla, dejará de comunicar su estado. Después, cuando la entrada de un ADB asociada al drone caído expire, el agente

borrará al drone caído de su ADB por lo que el nuevo líder será aquel con menor ID de entre los restantes. De forma similar, si un drone que no es líder falla, el resto de drones acabarán por borrar su entrada en la ADB y dejarán de tenerlo en cuenta en sus cálculos de modo que de forma emergente la flota se reorganizará para cubrir el hueco dejado por el agente caído.
Para mantener la formación en línea, los agentes se dirigen a la proyección de su posición sobre la línea que cruza al líder de lado a lado como se muestra en la siguiente figura.

Fig. 3. Alineamiento de los UAV.



(a) Flock Shape



(b) Flocking Behavior

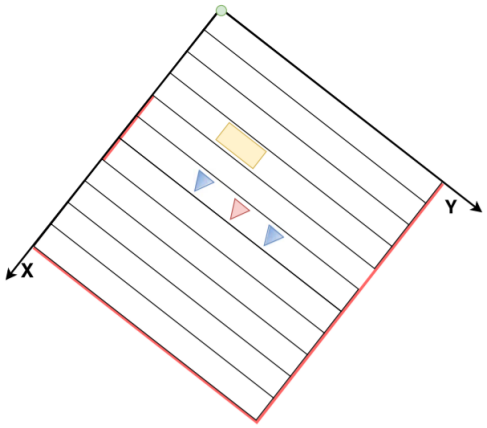
El algoritmo de flocking descrito en este trabajo se puede configurar modificando parámetros como por ejemplo:

- Intensidad con la que los agentes se mantienen separados (R1).
- Intensidad con la que un agente debe seguir la orientación de vuelo (R2).
- Intensidad con la que los agentes deben volar juntos (R3).
- Distancia que deben tratar de mantener los agentes durante la formación.
- Intensidad con la que una agente debe permanecer en la línea de formación.
- Rotación máxima permitida para corregir la dirección de vuelo.
- Velocidad máxima de un agente.

4.2 Composición de las capturas

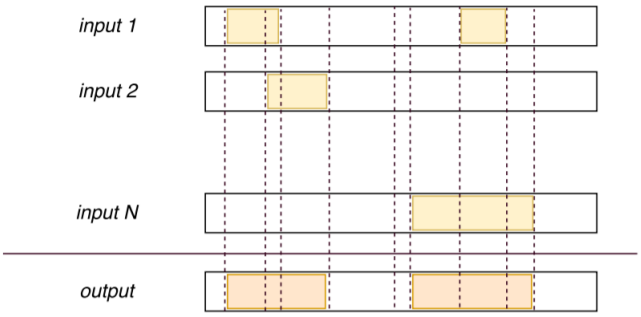
El área a monitorizar se divide en bandas horizontales que a su vez se dividen en regiones capturadas o sin capturar. Una región se define como la sección de una de estas bandas, delimitada por los dos extremos. La siguiente figura representa con un cuadrado una posible región del área. Una región puede tener extensión variable (eje Y de la figura) pero siempre tendrá la misma anchura (eje X de la figura) que la banda a la que pertenece.

Fig. 4. Bandas de un área y región de una banda.



Como hemos expuesto anteriormente, cada agente transmite la nueva región que ha monitorizado al resto de agentes hasta que esta información llega al líder el cual, periódicamente, combina todas las nuevas regiones monitorizadas por el resto de agentes para construir el nuevo mapa de regiones totales monitorizadas o capturadas para después transmitirlo de vuelta al resto de agentes para que actualicen sus APDs.
Aunque se busca minimizar el solapamiento entre regiones monitorizadas o capturadas, este solapamiento se produce y el líder, encargado de construir el nuevo mapa, es por tanto el encargado de combinar regiones solapadas.
El algoritmo propuesto por los autores une estas regiones solapadas quedándose sólo con los dos extremos que caen dentro de una única región, descartando el resto, y uniendo estos extremos para formar la región combinada. La siguiente figura ejemplifica esta operación.

Fig. 5. Unión de regiones de una banda.



Si el líder alcanza la última banda y han quedado huecos en el mapa de regiones capturadas, dará la vuelta y repetirá el proceso en dirección opuesta. Si no, todo el mapa habrá sido escaneado y los diferentes agentes podrán volver a sus correspondientes bases y aterrizar. De esta forma se garantiza que toda el área quedará escaneada.

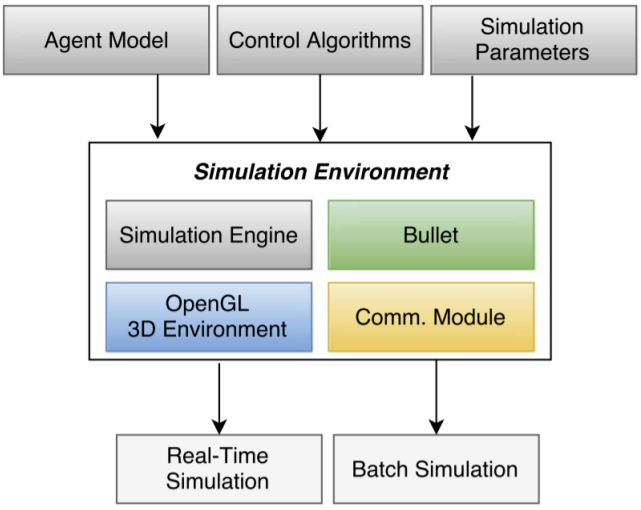
4.3 Planificación de ruta

El agente líder es el responsable de determinar la ruta que debe seguir y el resto de agentes volarán en formación a su alrededor. En la implementación propuesta por los autores, el agente líder calcula 2 posibles direcciones a tomar a partir de los bordes externos de varias regiones de la próxima banda horizontal que aún no han sido cubiertas. Después, elige aquella que esté más próxima a la flota de drones. Los autores proponen utilizar un suavizado para evitar cambios bruscos de dirección que puedan desestabilizar la flota.

4.4 Simulador

Los autores de este trabajo han creado un simulador 3D sobre el que probar los algoritmos planteados. Este simulador implementa físicas que los autores utilizan para detectar colisiones y mover los drones de forma realista usando fuerzas.
Para renderizar el escenario 3D han usado la librería OpenGL y para la simulación física han utilizado la librería Bullet. La comunicación entre agentes se realiza a través de un bróker de mensajes donde la latencia y la probabilidad de fallo en los mensajes son parámetros que se pueden controlar desde la interfaz del simulador. La siguiente figura muestra la arquitectura del simulador implementado.

Fig. 6. Arquitectura del simulador.



5 Evaluación experimental

Para evaluar que los algoritmos planteados permiten cumplir los objetivos propuestos, los autores han elegido comparar, para un área de 600 x 600 metros, cómo afecta el número de drones que participan en la misión.
Los experimentos descritos en este trabajo buscan evaluar los efectos de distintos parámetros de la simulación en el desempeño de la misión como por ejemplo:

- E1 - Efecto de la probabilidad de fallo de un solo agente y el número de agentes sobre el tiempo total de la misión y energía consumida.
- E2 - Efecto de la probabilidad de fallo en la sobre-cobertura.
- E3 - Proporción que pasan los agentes en los distintos estados.
- E4 - Efecto de la reducción de la distancia de separación mínima en el número de colisiones.

En el experimento E1 determinan que el número de agentes reduce el tiempo y la energía total requeridos para completar la misión hasta alcanzar un valor mínimo a partir del cual añadir agentes al sistema no mejora los resultados. Las siguientes gráficas muestran los resultados obtenidos en este experimento.

Fig. 7. Energía total consumida por número de agentes.

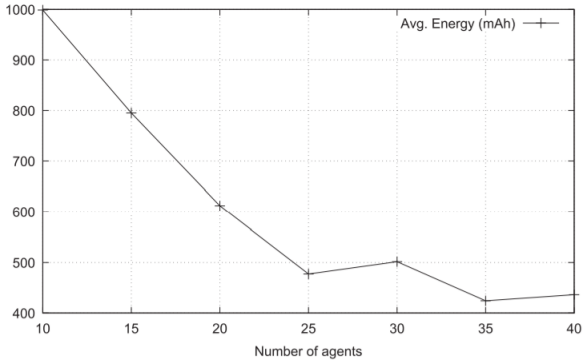
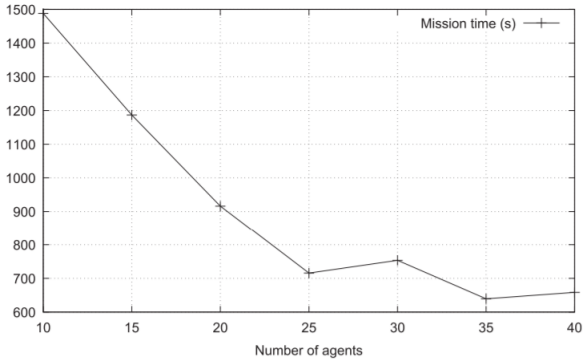
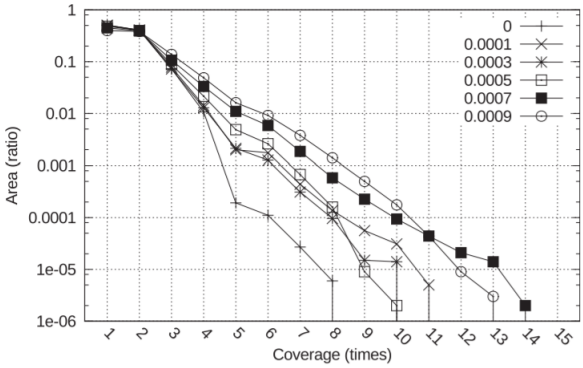


Fig. 8. Tiempo total de misión por número de agentes.



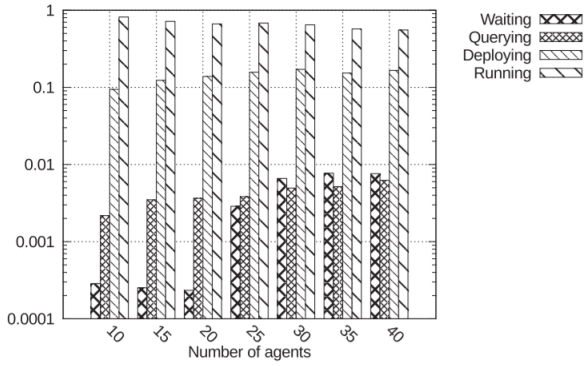
En el experimento E2 se busca evaluar la eficiencia de la estrategia usada para recuperar las áreas no capturadas debido a agentes fallidos. Los resultados obtenidos para este experimento determinan que si la probabilidad de fallo es 0, el 50% del área se capturará una única vez. Sin embargo, aumentar ligeramente esta probabilidad hasta el 0.09% no afecta de forma notable en la proporción del área que es cubierta más de una vez. La siguiente figura muestra los resultados obtenidos para este experimento.

Fig. 9. Sobre-cubrimiento en función de la probabilidad de fallo.



El número de agentes influye en el tiempo que el líder debe pasar en ciertos estados esperando a que el resto de agentes se posicionen o le alcancen (estados "Deploying" y "Waiting"). También influye en el tiempo requerido para que todos los agentes reciban mensajes del resto de agentes para así elegir un líder (estado "Querying"). En el experimento E3 se busca medir cuánto tiempo pasan los agentes en cada estado en función del número de agentes participantes, siendo el estado "Running" el que se busca maximizar, considerando el resto de estados como estados de servicio o improductivos. Los resultados de este experimento se muestran en la siguiente figura. La proporción de tiempo que el líder pasa en el estado "Running" es, en proporción, muy superior al tiempo empleado en el resto de estados.

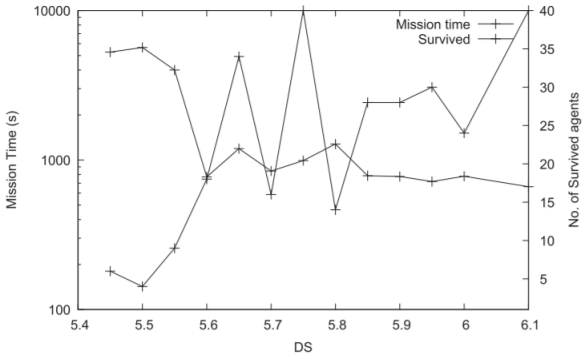
Fig. 10. Proporción de tiempo empleado en cada estado.



En el experimento E4 los autores miden cómo afecta la distancia de separación mínima al tiempo total de la misión y al número de agentes que colisionan.

Para el rango de distancia mínima utilizado, el número de colisiones varía mucho. Sin embargo, los autores consideran que el impacto de esta variación de la distancia mínima en el tiempo total de la misión no es muy grande. La siguiente figura muestra los resultados obtenidos en este experimento.

Fig. 11. Efecto de la distancia mínima entre UAV.



6 Adaptación a NetLogo

En nuestra propuesta de implementación en NetLogo buscaremos emular el algoritmo de flocking que se describe en el artículo. Además trataremos de visualizar la densidad con la que cada punto del área es monitorizada o capturada por los agentes.
El algoritmo implementado determina en cada tick cuánto debe avanzar un agente y en qué dirección. A continuación se muestra el pseudocódigo de nuestra adaptación del algoritmo.

$leader \leftarrow min_{ID}(turtles)$
if $t \neq leader$ **then**
  $flockmates \leftarrow findFlockmates(radius)$
  if $num_{flockmates} > 0$ **then**
    $nearest \leftarrow findNearestNeighbor(flockmates)$
    $\vec{R1} \leftarrow R1(nearest, minDistance)$
    $\vec{R2} \leftarrow R2(avg(heading_{flockmate}))$
    $\vec{R3} \leftarrow R3(avg(position_{flockmate}))$
    $\vec{R4} \leftarrow R4(leader)$
    $V \leftarrow [\vec{R1}, \vec{R2}, \vec{R3}, \vec{R4}]$
    $W \leftarrow [w_{R1}, w_{R2}, w_{R3}, w_{R4}]$
    $\vec{goal} \leftarrow weightedAvg(V, W)$
    $goal_{coordinates} \leftarrow position_{myself} + \vec{goal}$

$r \leftarrow towards(position_{myself}, goal_{coordinates})$
$r \leftarrow clampRotation(r, maxTurn)$
$rotate(r)$
$d \leftarrow minSpeed + (distanceWeight \times \|\vec{goal}\|)$
$moveForward(d)$
if $tick\%ticksToScan == 0$ **then**
$takePic()$
end if
end if
else
$rotate(degrees)$
$moveForward(distance)$
end if

En cada tick, se elige como líder al agente con menor Id.
Si el agente no es el líder, busca agentes vecinos dentro de un radio y guarda el más cercano. Después se calcula el vector que al sumarse a la posición del agente resultaría en la posición global a la que dirigirse para cumplir cada una de las reglas ($\vec{R1}$, $\vec{R2}$, $\vec{R3}$ y $\vec{R4}$). Desde la interfaz de NetLogo se puede iniciar que la magnitud de estos vectores se mantenga normalizada. A cada uno de estos vectores se les asocia un peso y se calcula su media ponderada. La dirección de este vector medio indica la dirección a la que debe avanzar el agente y su magnitud indica con qué intensidad debería hacer el movimiento o cuán lejos está de satisfacer las reglas. El agente avanza siempre en la dirección a la que apunta por lo que se calculan los grados que el agente debe rotar para que su dirección coincida con la dirección objetivo. Para evitar comportamientos erráticos, antes de aplicar esta rotación, el número de grados se limita a un valor máximo. Una vez el agente apunta a la dirección correcta el agente avanza una distancia que se calcula como una velocidad mínima (en distancia/tick) más la magnitud ponderada del vector objetivo. Por último, si es el momento de tomar una captura del terreno, el agente lo hará.
Si el agente es líder, se moverá de forma independiente con una rotación y velocidad que se pueden controlar desde la interfaz de NetLogo.

Los vectores objetivo de cada regla se calculan de la siguiente manera:

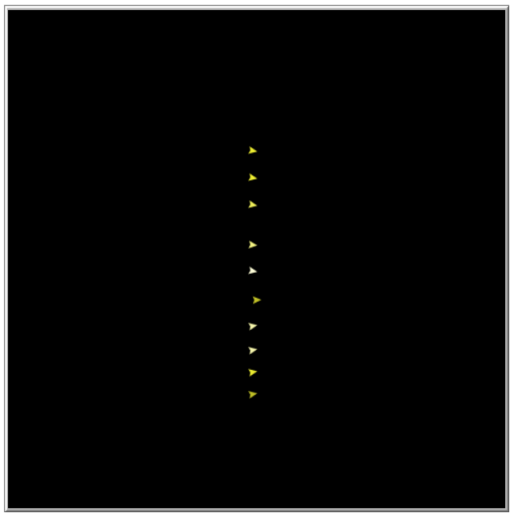
- R1: Para calcular $\vec{R1}$ se obtiene primero el vector que va del agente más cercano hasta nuestra posición ($\vec{v}$). Si $\|\vec{v}\|$ es menor que la distancia mínima, entonces $\vec{R1}$ tendrá la dirección de $\vec{v}$ y su magnitud, si no se usan magnitudes normalizadas, será la diferencia entre la distancia mínima y $\|\vec{v}\|$. Si $\|\vec{v}\|$ es mayor o igual a la distancia mínima, no se deberá hacer ninguna acción y $\vec{R1}$ será $(0, 0)$.
- R2: $\vec{R2}$ tiene la dirección hacia la que el agente deberá apuntar para que su rotación sea la misma que la rotación media de sus vecinos. Si no se usan magnitudes normalizadas, la magnitud de $\vec{R2}$ será la relación entre el número de grados a rotar y un valor constante.

- R3: $\vec{R3}$ tiene la dirección que va del agente hasta la posición media de los agentes vecinos. Si no se usan vectores normalizados, su magnitud será la distancia del agente hasta esa posición media.

- R4: $\vec{R4}$ tiene la dirección que va del agente hasta la proyección de la posición del agente sobre la línea infinita que cruza al agente líder. Si no se usan vectores normalizados, su magnitud será la distancia del agente hasta esa proyección.

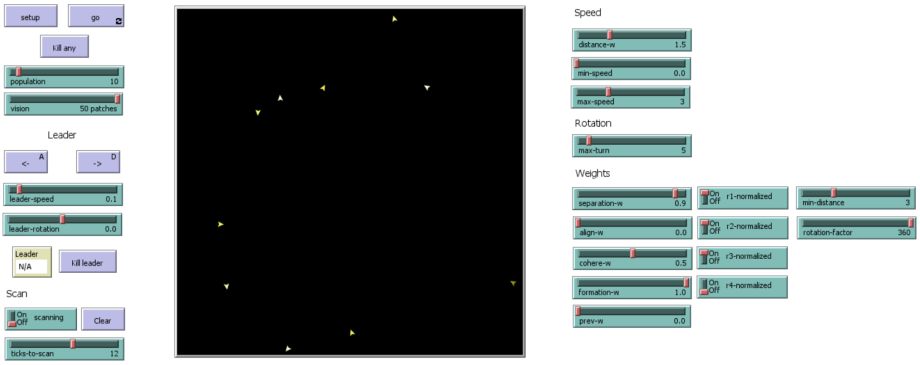
Como se puede ver en la siguiente figura, con la calibración adecuada, los agentes siguen una formación en línea alrededor del líder de forma similar a la formación descrita por los autores del artículo.

Fig. 12. Agentes volando en formación.



La siguiente captura muestra la interfaz de NetLogo de nuestra adaptación del algoritmo de flocking descrito en el artículo.

Fig. 13. Interfaz de NetLogo de nuestra adaptación.



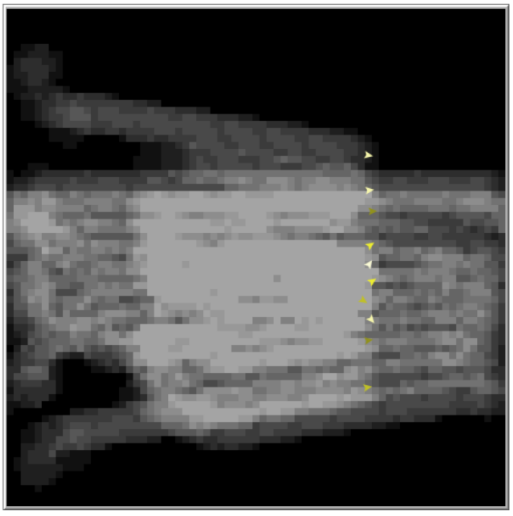
Desde esta interfaz se pueden modificar los siguientes parámetros principales:

- Número de agentes en la simulación.
- Rango de visión de cada agente.
- Rotación máxima de un agente.
- Velocidad mínima y máxima de los agentes.
- Dirección y velocidad del agente líder.
- Influencia de $\|\vec{goal}\|$ en la velocidad.
- Distancia mínima (regla R1).
- Interruptor para utilizar vectores normalizados.

Además, desde la interfaz de NetLogo es posible eliminar un agente cualquiera o eliminar el que en ese momento es el agente líder para comprobar que el algoritmo es resistente a fallos y observar cómo la flota de agentes se reorganiza para cubrir el hueco dejado por un agente eliminado o cómo se sigue a un nuevo líder si fuera necesario.

También es posible activar/desactivar y visualizar la monitorización de los agentes sobre el terreno marcándose las celdas del terreno con un color cada vez más claro en función de las veces que ha sido muestreado por algún agente. La siguiente figura muestra un terreno siendo escaneado por los agentes.

Fig. 14. Agentes monitorizando el área.



7 Conclusiones

En este trabajo hemos realizado un estudio académico del artículo *"A fault-tolerant self-organizing flocking approach for UAV aerial survey"* [1] en el que se presenta, simula y evalúa un sistema multi-agente compuesto de UAVs que vuelan en línea monitorizando un área determinada. Hemos analizado el contexto de este artículo describiendo los trabajos relacionados más importantes que se han dado antes y después de esta publicación, llegando a tratar el estado del arte en la actualidad en lo referente a sistemas multi-agente para la monitorización y captura de terrenos.
También hemos analizado el contenido del artículo describiendo la arquitectura del sistema y sus elementos, qué tareas realizan los agentes, cómo se comunican, cómo vuelan en formación, cómo se componen las regiones capturadas, cómo el líder planifica la ruta a seguir y cómo está implementado el simulador que utilizan los autores en los experimentos.
Hemos descrito los experimentos realizados más relevantes y los resultados obtenidos en los mismos.
Por último, hemos descrito la adaptación que hemos hecho en NetLogo del algoritmo de flocking que proponen los autores llegando a resultados satisfactorios que coinciden con el comportamiento descrito en el artículo original.

References

1. Benedetti, M.D. D'Urso, F. Fortino, G. Messina, F. Pappalardo, G. Santoro, C.: A fault-tolerant self-organizing flocking approach for UAV aerial survey. J. Netw. Comput. Appl. 96, 14-–30 (2017) https://doi.org/10.1016/j.jnca.2017.08.004

2. Van der Hoek, Wiebe, Wooldridge, Michael, 2008: Multi-agents systems. Found. Artif. Intell. 3, 887-928.

3. Wilensky, Uri, Rand, William, 2015. An Introduction to Agent-based Modeling: Modeling Natural, Social and Engineered Complex Systems with Netlogo. MIT Press.

4. Wei, Yi, Blake, Brian M., Madey, Gregory R., 2013. An operation-time simulation framework for uav swarm configuration and mission planning. Procedia Comput. Sci, 18, 1949-1958.

5. Sekin, Ahmet ada, Karpuz, Ceyhun, zek, Ahmet, 2016. Feature matching based positioning algorithm for swarm robotics. Comput. Electr. Eng.

6. Fazli, Pooyan, Davoodi, Alireza, Mackworth, Alan K., 2013. Multi-robot repeated area coverage. Auton. Robots 34 (4), 251-276.

7. Latombe, Jean-Claude, 1991a. Approzimate Cell Decomposition. Robot Motion Planning, Springer, 248-294.

8. Rekleitis, Ioannis, New, Ai Peng, Rankin, Edwatd Samuel, Choset, Howie, 2008. Efficient boustrophedon multi-robot coverage: an algorithmc approach Ann, Math. Artif. Intell. 52 (24), 109-142.

9. Xiao Liu, Yuanwei Liu, Yue Chen y Lajos Hanzo, 2019. Trajectory Design and Power Control for Multi-UAV Assisted Wireless Networks: A Machine Learning Approach"

10. Yeongho Song, Myeonggeun Gu, Joonwon Choi, Hyondong Oh, Seunghan Lim, Hyo-Sang Shin y Antonios Tsourdos, 2021. Using Lazy Agents to Improve the Flocking Efficiency of Multiple UAVs. https://doi.org/https://doi.org/10.1007/s10846-021-01492-1

11. Peng Yan, Chengchao Bai, Hongxing Sheng y Jileng Guo, 2020. Flocking Control of UAV Swarms with Deep Reinforcement Learning Approach.

12. Felipe Cucker and Steve Smale, 2007. Emergent Behavior in Flocks.