



Modelling Pedestrian Behaviour Using Swarm Techniques

Yago Ávila Moré , Basil Mohammed Al-Hadithi , and Victor Cadix Martín 

Abstract—Modelling pedestrians and groups of people is a highly multidisciplinary technique, given the significant interest it attracts from various branches of science and engineering. This results in many different methodologies that may arise from diverse objectives. The model developed in this work is an agent-based model, in which pedestrian behaviour is defined by a set of forces. Each force models an aspect of pedestrian gait, with the objective of creating a virtual environment to train and test control systems for collaborative robots or autonomous vehicles. To meet the modelling requirements, the system employs various algorithms, such as "flocking" which simulates the coordination and formation of groups, "pathfinding" which enables agents to discover optimal routes within a given space, and algorithms specialized in avoiding walls and dynamic obstacles. These components collaborate to accurately depict how crowds move and react in different environments and situations. Thanks to the modularity of this approach, which facilitates the adjustment and expansion of the components, the developed system can be integrated into various applications, such as simulating non-playable characters (NPCs) in video games or modelling the evacuation of a building.

Link to graphical and video abstracts, and to code: <https://latam.ieeer9.org/index.php/transactions/article/view/8927>

Index Terms—Swarm algorithms, Crowd simulation, Pathfinding, Pedestrian modeling, NPCs, Obstacle avoidance, Crowd dynamics, Flocking, Environment modeling, Autonomous agents, Path planning.

I. INTRODUCCIÓN

En los últimos años, ha habido un gran crecimiento en el desarrollo e investigación de robots y vehículos autónomos. En específico, en aquellos orientados a trabajar en entornos con personas, como la automatización de coches, robots de reparto autónomos o vehículos de transporte de personas con movilidad reducida. Esto ha generado la necesidad de disponer de entornos virtuales fidedignos, donde poder llevar a cabo pruebas de manera segura y poder evaluar y perfeccionar los sistemas de control de estos robots y su interacción con grupos de personas. Como respuesta a dicha necesidad, en este trabajo se pretende modelar el comportamiento de peatones y grupos de personas mediante técnicas de enjambre.

Dentro del ámbito del modelado de grupos de personas, destaca la gran diversidad de los estudios publicados. Este aspecto multidisciplinar se debe a que existe un gran interés

por parte de muchas ramas de la ciencia y la ingeniería, como la física aplicada [1], la robótica [2], las comunicaciones inalámbricas [3], la ingeniería civil [4], la antropología [5], etc. Esto hace que existan una gran cantidad de metodologías diferentes, que pueden provenir de objetivos diferentes, lo que hace difícil poder establecer criterios sólidos para agrupar estos modelos.

Un criterio clásico utilizado en la literatura es la diferenciación entre modelos macroscópicos, microscópicos y mesoscópicos. En los modelos macroscópicos, el estado del sistema viene definido por magnitudes como la densidad y el momento lineal. Los modelos más representativos son los desarrollados por Roger L. Hughes [4], [6], en los que aprovecha los modelos propios de la dinámica de fluidos para caracterizar el comportamiento de las multitudes. En los modelos microscópicos se define individualmente el estado de cada agente. El estado general del sistema queda definido por cada agente, con sus características propias de posición y velocidad. Uno de los modelos más importantes es el modelo de Fuerza Social diseñado por Dirk Helbing y Peter Molnar [7], en el que se hacía una división de estas fuerzas en fuerzas de auto-impulso y fuerzas de interacción social. Por último, en los modelos mesoscópicos se mantiene la identificación individual de cada agente, sin embargo el sistema global viene representado mediante una distribución de probabilidad adecuada sobre el estado microscópico de los agentes. Uno de los trabajos más representativos es el de Nicola Bellomo et al. [8], en el que se basan en la teoría cinética para modelar multitudes.

El modelo desarrollado es un modelo microscópico que se basa en los algoritmos de *Flocking* (Enjambre) de Craig Reynold's [9]. Se han elegido estos algoritmos gracias a su capacidad modelar comportamientos complejos y dinámicos de grupos a partir de un conjunto de reglas individuales simples. Siendo capaces de simular interacciones locales que dan lugar a comportamientos colectivos sin necesidad de una coordinación centralizada, lo que representa con precisión cómo interactúan los peatones en entornos reales. A parte de estos, se diseñan otros algoritmos que modelen las diferentes características del movimiento de los peatones. Como son la navegación, compuesta por la búsqueda de caminos y la evasión de obstáculos estáticos, o la evasión de obstáculos dinámicos.

Si bien, el trabajo de Reynold's no busca modelar el comportamiento de personas, sí que ha servido de base para el desarrollo de otros trabajos. Como un sistema basado en los algoritmos de flocking para evaluar la circulación dentro de un edificio [10], u otro que utiliza estos algoritmos para

Y. Á. Moré, B. M. Al-Hadithi, and V. C. Martín are with School of Industrial Design and Engineering, Universidad Politécnica de Madrid, Madrid, Spain (e-mails: yago.avila.more@alumnos.upm.es, basil.alhadithi@upm.es, and victor.cadix.martin@alumnos.upm.es).

la animación de grupos de personas [11]. También se han utilizado estos algoritmos para crear sistemas multiagente de vehículos aéreos no tripulados (UAVs) [12].

En el apartado 2 se dará una explicación del modelo de agente que se ha utilizado. En el apartado 3 se verá el desarrollo de los diferentes algoritmos que se han utilizado para modelar los comportamientos. En el apartado 4 se expondrán y discutirán los resultados obtenidos. Por último, en el apartado 5 se darán unas conclusiones y posibles trabajos futuros.

II. AGENTE AUTÓNOMO

Un agente es una entidad que tiene la capacidad de percibir su entorno a través de sensores o algún medio de observación, y de influir en ese entorno mediante actuadores, pudiendo realizar tareas o acciones específicas sin la intervención o dirección externa continua. [13]

El comportamiento de un agente, en cuanto al movimiento se refiere, se puede dividir en tres niveles: acción, dirección y locomoción, como puede verse en la Fig. 1. La capa de acción es la que opera a más alto nivel, se encarga de definir a nivel estratégico los objetivos a alcanzar por los agentes, así como el orden o prioridad de estos. Se puede ver como los decisiones conscientes que toma una persona. La capa de dirección se encarga de definir la ruta y la evasión de obstáculos para alcanzar los objetivos propuestos en la capa de acción. Por último, la capa de locomoción es la encarnación física del agente y se encarga de la física del movimiento, colisiones, etc.

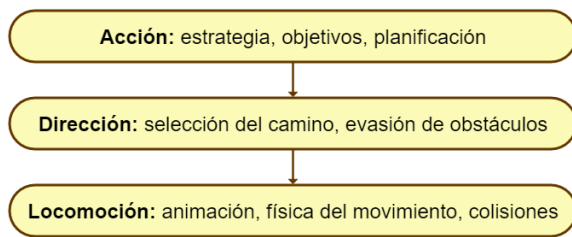


Fig. 1. Capas de un agente autónomo.

El desarrollo de este trabajo se enfoca en la capa dirección. En cuanto a la capa de locomoción, el modelo de agente usado se basa en una aproximación de masa puntual, caracterizada por una física muy simple y computacionalmente barata. Gracias a que las capas son independientes entre sí, este modelo puede ser cambiado por uno más realista si la aplicación lo requiere.

III. ALGORITMOS DEL COMPORTAMIENTO

El sistema de dirección desarrollado se basa en la combinación de algoritmos de comportamientos básicos, de forma que se puedan conseguir comportamientos más complejos y realistas. La principal ventaja de este diseño, reside en el comportamiento atómico de estos algoritmos, de manera que sean independientes entre ellos. Esto proporciona una gran modularidad, ya que permite agregar o quitar comportamientos en función de la situación que se quiera representar.

El funcionamiento del algoritmo a nivel global, consiste en que cada agente calcula las diferentes componentes de aceleración en función de su entorno y de los otros agentes que están en su zona de percepción. Posteriormente se suman estas componentes, ponderadas por un peso, y se calcula la aceleración final que percibe el agente.

A. Flocking

El primer algoritmo del que vamos a hablar es el *flocking*, que es el encargado de generar los comportamientos de grupo. Se compone de tres “subalgoritmos”: cohesión, separación y alineamiento. Estos operan con lo que se ha decidido llamar “*flock local*”.

El *flock local* es el conjunto de agentes, que forman parte de su grupo y que están dentro de su rango de percepción, en los que se basa para definir sus movimientos. Para calcular este conjunto, se genera una geometría alrededor del agente que delimite cuáles se consideran dentro de su grupo y cuáles no. El comportamiento del agente variará en función de como se defina dicho área. Sin embargo se ha comprobado que el comportamiento es menos sensible a variaciones en dicho área respecto a las ganancias de repulsión, cohesión y alineación [14]. Por esto para éste trabajo, por comodidad, eficiencia y simpleza, se ha decidido usar un área circular alrededor del agente.

La componente de cohesión se encarga de modelar el comportamiento por el cual los agentes tienden a permanecer cerca unos de otros. Esta componente su calcula de forma que sea proporcional a la distancia entre el agente y el “centro de masas” del *flock local*. Este centro, se calcula haciendo la media de las posiciones de los agentes. En la Fig. 2, se puede ver al agente para el que se está haciendo el cálculo (círculo verde), a los agentes vecinos (círculo azul) y los centros teniendo en cuenta o no al propio agente (amarillo y rojo, respectivamente). Añadir o no al propio agente para el cálculo del centro, hace que el vector “agente-centro” no cambie su dirección, pero sí su módulo, que decrecerá. Por lo que, para este trabajo, se ha decidido no incluirlo en el cálculo ya que proporciona un comportamiento más cohesivo.

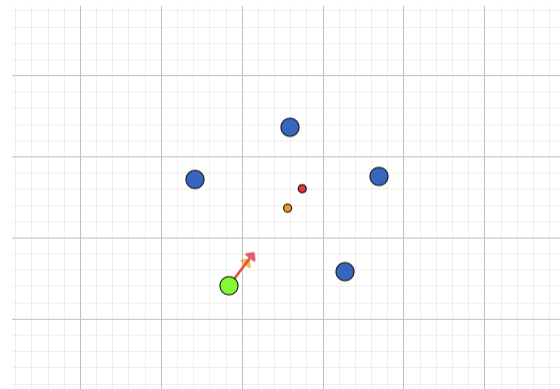


Fig. 2. Ejemplo de cálculo de centro de *flock local* y valores de la componente de cohesión.

La alineación busca modelar la tendencia de las personas de compartir un movimiento común, tanto en la dirección como

en la velocidad. Este comportamiento contribuye a la formación de patrones de movimiento uniformes y da coherencia al movimiento del grupo. Esta componente se calcula de forma que sea proporcional a la diferencia entre el vector velocidad actual y el vector velocidad media del *flock* local. De nuevo, tenemos la opción de incluir la velocidad del agente o no, en la Fig. 3 se puede ver la diferencia entre el vector naranja, que incluye la dirección del agente, y el rojo, que no. Al incluirla, el agente ‘auto-valida’ su dirección, haciendo que sea menos propenso a cambiarla, lo que puede aportar mayor estabilidad y cohesión al grupo. Por otro lado, al excluirla del cálculo, basa el cálculo únicamente en la información de sus vecinos, lo que puede resultar en una adaptación más rápida a los cambios de dirección del grupo, pero también una mayor sensibilidad a perturbaciones. En función de las pruebas realizadas se ha decidido optar por incluir la velocidad del agente en el cálculo.

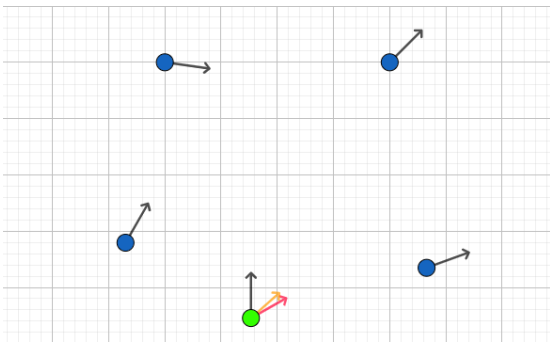


Fig. 3. Ejemplo de cálculo de velocidad deseada.

Por último, la separación se encarga de hacer que los agentes del *flock* no colisionen entre sí, de forma que en la medida de lo posible se respete el espacio personal de cada uno. Para calcular esta componente, se calcula la separación individual del agente con los diferentes vecinos, y se hace una media para calcular la componente final. En la Fig. 4 se puede ver un ejemplo gráfico del cálculo. Estas componentes individuales se calculan de forma que sean inversamente proporcional a la distancia. Dado que la función $1/x$ crece asintóticamente al infinito en 0, se debe prestar especial atención a la hora de implementarlo, evitando que tome demasiado peso en el cómputo global, haciendo que se generen comportamientos erráticos.

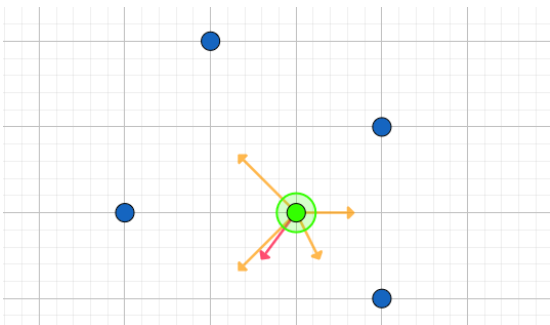


Fig. 4. Ejemplo gráfico de cálculo de la componente de separación. Los vectores naranjas son las diferentes componentes de separación y el vector rojo es la componente final.

B. Sistema de Pathfinding (búsqueda de caminos)

Los sistemas de pathfinding generalmente utilizados se basan en buscar una ruta del punto A al punto B, como por ejemplo el algoritmo A* que puede verse en la Fig. 5, donde las casillas verde y roja son el origen y el destino; las casillas azules, el camino encontrado; las verde claro, las que se han explorado; y las amarillas, las que quedaron en la lista abierta al encontrar el destino.

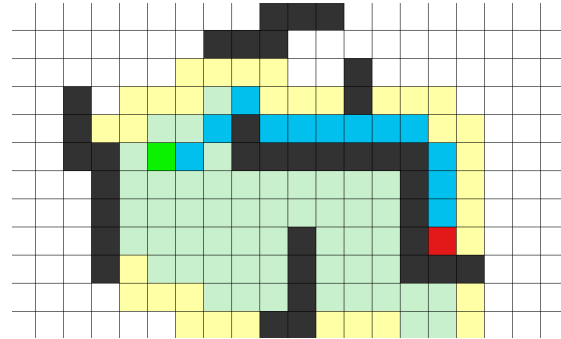


Fig. 5. Ejemplo de camino generado por el algoritmo A*.

Sin embargo, al trabajar con grupos de agentes, algunos pueden salirse del camino debido a las interacciones entre ellos o con obstáculos. En esta situación, se puede manejar su vuelta al camino o recalcularlo para los agentes extraviados. Para tratar de evitar esto, se han probado diferentes soluciones como hacer que las casillas adyacentes redirigieran al camino original, pero generaba un comportamiento muy poco natural. O aprovechar las relaciones de parentesco creadas durante el proceso de búsqueda, que terminaría redirigiendo a los agentes a una casilla del camino principal. El problema de este método, el número de casillas exploradas no es constante y el camino de vuelta generado puede no ser óptimo.

Finalmente, se ha optado por el uso de *flow fields* (campos de flujo) o *direction maps* (mapa de direcciones) [15], [16], una técnica de *pathfinding* que generan un campo vectorial en el espacio de navegación, donde a cada casilla se le asigna una dirección, de modo que siguiéndolas se llega desde cualquier punto al objetivo, en la menor distancia posible. En este caso cada agente sigue la dirección de la casilla sobre la que se encuentra actualmente, a modo de velocidad deseada, aplicando una aceleración que transforme su velocidad actual en esta.

Para asignar las direcciones se pueden utilizar diferentes técnicas, para este trabajo se ha decidido hacer una modificación del algoritmo A*. En los *flow field* no hay casilla de origen, solo hay destino, por lo que el algoritmo de búsqueda debe empezar por este, de forma que se expanda por toda la superficie. El algoritmo utiliza dos estructuras principales para la exploración del mapa. La primera, es el conjunto de casillas visitadas, denominado ‘conjunto cerrado’. Se utiliza para asegurar que no se vuelven a visitar nodos ya comprobados, y como es lógico, empieza estando vacío. La segunda, es el conjunto de casillas que se han descubierto, pero todavía no se han visitado, llamado ‘conjunto abierto’ o ‘frontera’. Cada vez que se visita una casilla, se añaden a esta lista las casillas vecinas que son transitables y no están en la lista cerrada.

Cuando se añade una nueva casilla a la lista abierta, se le asocia un coste y se establece una relación de paternidad con la casilla por la que ha sido añadida. Este coste es la suma del coste de la casilla “padre” y un coste extra de pasar del “padre” a la casilla que se añade al conjunto abierto. En este caso, al ser un *grid* cuadrado, este coste es de 10 para los casos en los que el movimiento sea horizontal o vertical, y 14 ($10 \cdot \sqrt{2}$) para el movimiento diagonal. Las casillas a su vez pueden tener un modificador que escale este coste, para representar zonas menos atractivas para los peatones.

Si al añadir una casilla a la lista abierta, esta ya pertenece al conjunto abierto, se debe verificar si el nuevo coste para llegar a ella es inferior al registrado previamente. De ser así, se actualiza el coste de la casilla y modifica su nodo padre al que ofrece el camino con menor coste. En caso de que el coste sea mayor o igual, no se hacen cambios en el coste ni en la paternidad de la casilla.

A la hora de visitar las casillas de la lista abierta, se hace en función de la variable de coste que se ha ido adjudicando. De modo que se visiten primero las casillas de menor coste, asegurando así, que todo *tile* que entra en el conjunto cerrado lo ha hecho con el menor coste posible. En la Fig. 6 se puede ver un ejemplo de un *flow field*. En él, se pueden encontrar las direcciones de de casa casilla para llegar al destino, la casilla roja.

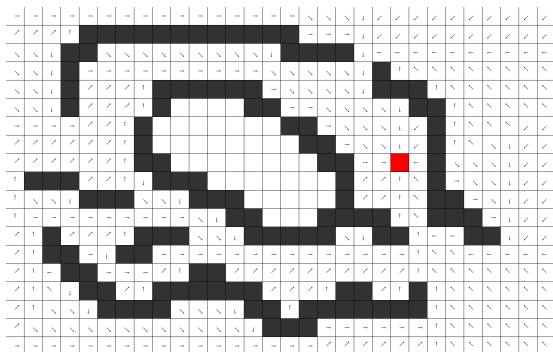


Fig. 6. Ejemplo de *flow field*.

Este sistema presenta tres problemas principales. El primero es el comportamiento en las esquinas, para mejorar esto lo que se ha hecho es impedir que la casilla que tiene como vecina la esquina de un muro no pueda tomar la dirección diagonal que le llevaría a pasar dicha esquina. El segundo problema, reside en que es el único sistema que trabaja a nivel global, y no con la información local de cada agente. Esto hace que sea vulnerable a cambios dinámicos en el entorno. Para solucionar esto, se hace que el sistema recalculé los *flow fields* de los grupos cuando se produce un cambio en la transitabilidad de las casillas. Y el tercero, es el comportamiento en las inmediaciones del destino. Como ya hemos dicho, cada casilla puede tomar 8 posibles direcciones, estas son suficientes para que el movimiento por el mapa, por lo general, sea fluido. Sin embargo, en zonas abiertas, es raro que los agentes se encuentren justo en una casilla que le permita ir directamente a su destino usando estas 8 direcciones.

Para solucionar esto y que el movimiento al destino se haga de forma directa, se implementa el concepto de *Line of Sight*

[17]. Esto consiste en asignar una propiedad a las casillas en función de si desde ellas hay línea de visión directa con el destino, de forma que si un agente se encuentra en una de ellas ignore el *flow field*, y su componente de dirección sea directamente un vector que apunte al objetivo. Para calcular que zonas tienen línea de visión con el destino y cuales no, se utiliza una técnica parecida al *raycasting*. Consistente en lanzar “rayos” desde la casilla objetivo a las del perímetro del mapa. Estos “rayos” van aplicando la propiedad de *Line of Sight* a las casillas con las que se van cruzando, hasta que se tope con un muro u obstáculo. Para el trazado de esos “rayos” se utiliza el algoritmo de Bresenham, muy eficiente para la representación de líneas rectas en cuadrículas discretas, como las matrices de leds de los dispositivos gráficos o, en este caso, un *grid* de casillas. En la Fig. 7 se puede ver un ejemplo de un *flow field* completo.

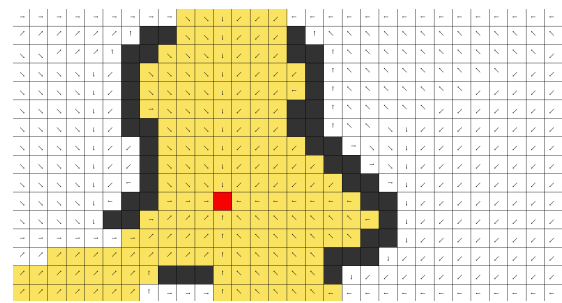


Fig. 7. Ejemplo de *flow field* incluyendo el sistema de línea de visión.

C. Sistema de Evasión de Obstáculos Dinámicos

Este algoritmo se encarga tanto de la evasión de obstáculos dinámicos así como la evasión de otros *flocks* o grupos. En la simulación de agentes en videojuegos y animación, suelen utilizar sistemas basados en el obstáculo de velocidad, como RVO [18], o ClearPath [19]. Sin embargo, en este trabajo, se ha decidido aprovechar el comportamiento de separación del *flocking* para modelar la evasión de obstáculos dinámicos.

Una implementación simple de este algoritmo es suficientemente buena para una primera aproximación. No obstante, produce comportamientos muy bruscos en la evasión. Para corregirlos, el objetivo es representar la capacidad de anticipación de las personas. Esto se consigue mediante la proyección de la posición del agente y el obstáculo en el futuro. Entonces, en lugar de calcular la componente de separación, utilizando las posiciones actuales, se usa la posición futura de ambos, como puede verse en la Fig. 8. De esta forma, si se prevé una futura colisión, se empezará a modificar el movimiento de forma preventiva, haciendo que se produzca un comportamiento más suave.

Es importante seleccionar una correcta distancia en la que se proyecta la posición, dado que esto se hace teniendo en cuenta la velocidad actual, por lo que, si esta cambia de manera repentina, se puede tener una discontinuidad en el trazo de las futuras proyecciones. Esto puede provocar situaciones en las que un agente esté a punto de colisionar y, sin embargo, mediante la comparación de las posiciones futuras no detecta colisión. Si bien, al trabajar en aceleraciones los cambios de

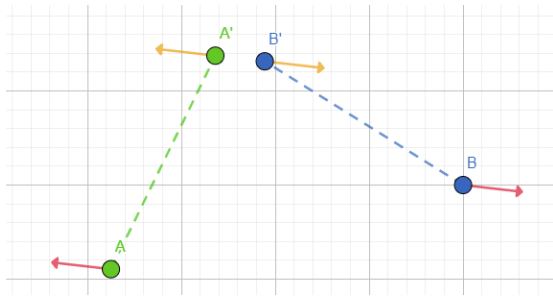


Fig. 8. Cálculo de la componente de evasión en función de la posición futura.

velocidad son continuos, sigue siendo un modelo digital que funciona por intervalos de tiempo.

Como se ha dicho, esto solo se produce en situaciones de cambio de dirección bruscos. Donde, con un correcto ajuste de los parámetros, como el tiempo de proyección o la distancia de detección, se soluciona en gran medida este problema. También se puede modificar la velocidad de giro máxima, pero eso es un parámetro de la capa de locomoción y no consideramos correcto que este se modifique para solucionar un problema de la capa de dirección. En las pruebas realizadas, se ha visto que este comportamiento se da en situaciones como, al coincidir dos grupos pasando una esquina o al reasignar el destino de un grupo habiendo otro cerca. Situaciones donde de manera natural también pueden verse comportamientos bruscos o erráticos.

El caso de la evasión entre dos agentes, es un caso especial ya que existe una reciprocidad en la evasión. Ambos van a tomar acciones para evitar la colisión, por lo que la fuerza que se ejerce se divide entre los dos. En el caso de los obstáculos en sí, se considera que estos son elementos sin control que no ejercerán ninguna acción para evitar la colisión, por lo que toda la acción de esquivar la ejercerá completamente el agente.

La última modificación que se ha hecho también diferencia entre agente y obstáculo. Según se ha definido, la fuerza resultante es contraria al vector de distancia, esto hace que para encuentros frontales la mayor parte de la componente sea de desaceleración, haciendo que sea fácil que el agente se llegue a parar. Esto puede ser deseable en el caso de obstáculos, pero cuando dos grupos de personas se encuentran, lo que nos dice la experiencia es que, si el espacio lo permite, la componente principal del movimiento sea lateral. Para representar esto se descompone la fuerza en una componente lateral y otra de desaceleración, limitando esta última y potenciando la componente lateral.

Este método de evasión de obstáculos es capaz de representar situaciones comunes, como los breves momentos de duda al encontrarse de repente a una persona de frente, o la evasión entre dos grupos de personas que se entremezclan, como se puede ver en la Fig. 9.

D. Evasión de Muros

Por último, queda hablar de la evasión de muros, si bien esto es ya una de las funciones del *pathfinding*, cuando se introducen otros comportamientos, como el *flocking* o la evasión de

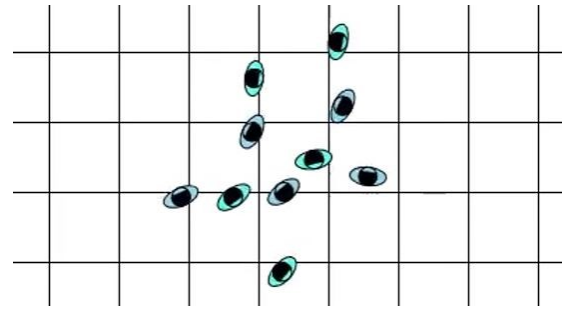


Fig. 9. Evasión de los agentes atravesándose entre los grupos.

obstáculos dinámicos, la aceleración final no siempre mantiene a los agentes en las casillas transitables. Para solucionar este problema, se ha adoptado una técnica utilizada en el desarrollo de videojuegos, el uso de *feelers*, o antenas, para percibir el entorno, un principio similar a los sensores de proximidad en robótica y vehículos autónomos.

La implementación de esta técnica implica añadir protuberancias a los agentes, que permiten medir la proximidad de un muro u obstáculo. Dichas protuberancias no son visibles para el usuario y, generalmente, se sitúan en un arco delante del agente. Tras las pruebas realizadas se ha visto que el número óptimo de antenas es cuatro: dos aproximadamente a 30° a cada lado del eje de dirección y dos perpendiculares, aunque estas últimas más cortas. La longitud de las antenas perpendiculares se ajusta de forma que el agente camine con una distancia de seguridad de la pared. Cuando estas antenas entran en contacto con un muro u obstáculo, hacen que se genere una fuerza en el agente que lo separe del muro, en función de la profundidad que ha penetrado la antena

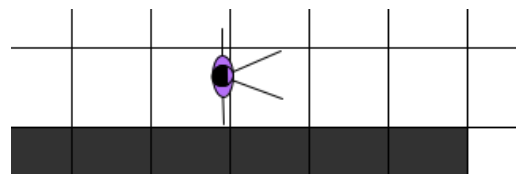


Fig. 10. Agente con los *feelers* visibles.

IV. RESULTADOS

El sistema diseñado se adapta muy bien a situaciones diferentes, tanto en el número de agentes que es capaz de manejar como en las situaciones que en la que es capaz de generar un resultado satisfactorio. Gracias a que los comportamientos no están prediseñados y son reactivos al entorno, encontramos respuestas muy diferentes para situaciones prácticamente iguales, lo cual representa muy bien la naturaleza caótica del movimiento de peatones.

A continuación se muestra un conjunto de imágenes de una de las pruebas realizadas. Se han seleccionado 6 fotogramas, que se pueden ver en la Fig. 11, que permiten ver como actúan todas las componentes entre sí. El vídeo completo de la prueba está disponible en el siguiente enlace¹.

¹<https://youtu.be/I20BYTusXPA>

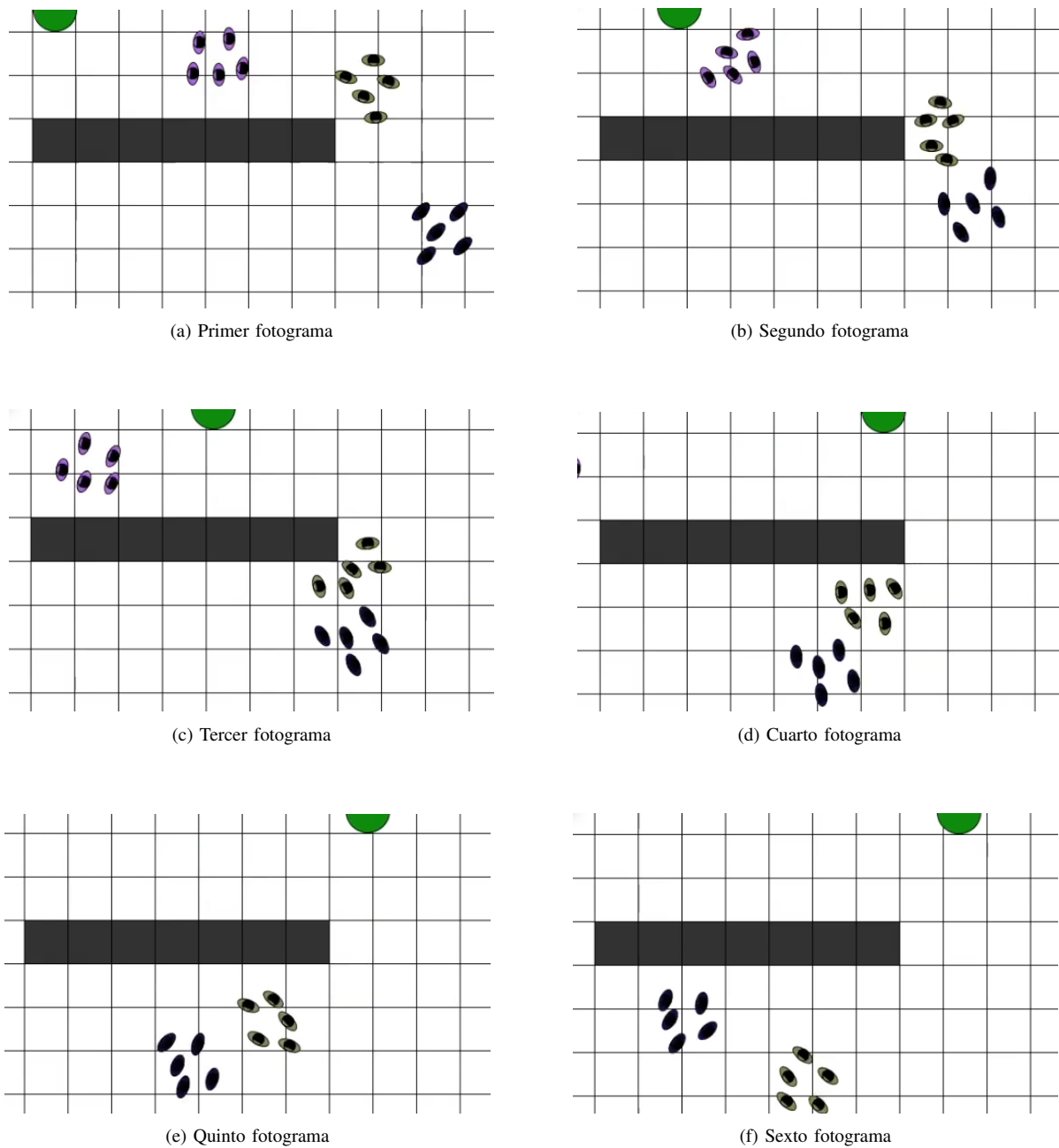


Fig. 11. Conjunto de fotogramas de una prueba del funcionamiento del algoritmo.

En este conjunto de imágenes se pueden ver dos grupos de peatones interactuando en las inmediaciones de una esquina, así como otro grupo de peatones que se encuentra un obstáculo en movimiento en la cercanía de un muro. Ambas situaciones son muy complicadas ya que entran en juego todas las componentes entre sí.

En el primer caso, se puede ver dos grupos que llevan trayectorias que tienden a colisionar. Para referirnos a ellos llamaremos “grupo 1” al que en el primer fotograma se encuentra

en la parte superior, y “grupo 2” al que se encuentra en la parte inferior. Como se puede ver en el segundo fotograma el grupo 2 cambia su dirección dejando más distancia con el muro, esto se puede ver también en los fotogramas 3 y 4. Por su parte, el grupo 1 reduce su velocidad, compactándose ligeramente, y continúa moviéndose a muy baja velocidad hasta que el grupo 2 termina de pasar delante de él en el fotograma 5. En este mismo fotograma 5, se puede ver que una vez pasado el obstáculo del grupo 1, el grupo 2 vuelve a acercarse al muro,

recuperando su dirección original. También se puede ver entre el fotograma 5 y 6 como el grupo 1 incrementa su velocidad al no tener ningún obstáculo delante.

En el segundo caso la acción es más rápida. Se puede ver como un grupo de peatones se echa hacia un lado ante la presencia de un objeto móvil que se acerca a ellos. Esta escena nos puede recordar fácilmente a un grupo de peatones andando por una calle, teniendo que desviar la trayectoria hacia un lado cuando un coche pasa. Una vez pasado el obstáculo vuelven a andar con normalidad, manteniendo de nuevo las distancias interpersonales que tenían anteriormente.

V. CONCLUSIONES

La simulación de peatones es un campo extenso y en constante evolución, donde hay interés por parte de múltiples disciplinas académicas y profesionales. En este campo, se han desarrollado diversos modelos con objetivos y necesidades muy variadas, lo que lleva a innovar y estimular la creatividad en las soluciones aportadas.

En este trabajo se ha conseguido desarrollar un simulador que permitirá la puesta a prueba de robots o vehículos autónomos que vayan a operar en entornos compartidos con personas. A lo largo del proceso de desarrollo, se ha procurado que se mantenga un alto grado de modularidad, con la implementación e integración de comportamientos básicos independientes, dejando espacio para futuras inclusiones y mejoras. Como agentes líderes, a los cuales el resto de agentes sigan. O al revés, agentes de los cuales el resto huya. Esta modularidad es una de las mayores ventajas de este sistema en comparación con otros, ya que permite su aplicación en una amplia variedad de situaciones.

En cuanto a la escalabilidad del modelo, gracias a la independencia de los cálculos de las fuerzas entre los agentes, se pueden añadir técnicas de computación paralela. Esto es crucial para escenarios con alta densidad de peatones.

Como posibles futuros trabajos, se contempla el uso del algoritmo en entornos reales. Tanto para la predicción de la posición de peatones por parte de robots o vehículos autónomos, como para el control de enjambres de los mismos. En segundo lugar, se plantea cambiar el sistema de evasión de obstáculos dinámicos por uno más robusto, seguramente uno basado en obstáculos de velocidad. También se propone implementar una componente de comportamiento caótico que permita simular el comportamiento de peatones en situaciones de alto estrés como la evacuación de un edificio ante una situación de emergencia.

REFERENCES

- [1] J. Toner and Y. Tu, "Flocks, herds, and schools: A quantitative theory of flocking," *Physical review E*, vol. 58, no. 4, p. 4828, 1998. DOI: 10.1103/PhysRevE.58.4828.
- [2] A. Aroor, S. L. Esptein, and R. Korpan, "Mengers: A crowd simulation tool for autonomous robot navigation," in *2017 AAI Fall Symposium Series*, 2017. DOI:10.48550/arXiv.1801.08823.
- [3] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless communications and mobile computing*, vol. 2, no. 5, pp. 483–502, 2002. DOI:10.1002/wcm.72.
- [4] R. L. Hughes, "A continuum theory for the flow of pedestrians," *Transportation Research Part B: Methodological*, vol. 36, no. 6, pp. 507–535, 2002. DOI:10.1016/S0191-2615(01)00015-7.

- [5] R.-Y. Guo, H.-J. Huang, and S. Wong, "Collection, spillback, and dissipation in pedestrian evacuation: A network-based method," *Transportation Research Part B: Methodological*, vol. 45, no. 3, pp. 490–506, 2011. DOI:10.1016/j.trb.2010.09.009.
- [6] R. L. Hughes, "The flow of human crowds," *Annual review of fluid mechanics*, vol. 35, no. 1, pp. 169–182, 2003. DOI:10.1146/annurev.fluid.35.101101.161136.
- [7] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995. DOI:10.1103/PhysRevE.51.4282.
- [8] N. Bellomo, A. Bellouquid, and D. Knopoff, "From the microscale to collective crowd dynamics," *Multiscale Modeling & Simulation*, vol. 11, no. 3, pp. 943–963, 2013. DOI:10.1137/130904569.
- [9] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 25–34, 1987. DOI:10.1145/37402.37406.
- [10] C.-T. Li and S.-D. Lin, "Evaplanner: an evacuation planner with social-based flocking kinetics," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1568–1571, 2012. DOI:10.1145/2339530.2339782.
- [11] M. Dewi, M. Hariadi, and M. H. Purnomo, "Simulating the movement of the crowd in an environment using flocking," in *2011 2nd International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering*, pp. 186–191, IEEE, 2011. DOI:10.1109/ICICI-BME.2011.6108638.
- [12] Y. Feng, J. Dong, J. Wang, and H. Zhu, "Distributed flocking algorithm for multi-uav system based on behavior method and topological communication," *Journal of Bionic Engineering*, vol. 20, no. 2, pp. 782–796, 2023. DOI:10.1007/s42235-022-00287-w.
- [13] C. W. Reynolds *et al.*, "Steering behaviors for autonomous characters," in *Game developers conference*, vol. 1999, pp. 763–782, Citeseer, 1999. DOI: not available.
- [14] E. Soria, F. Schiano, and D. Floreano, "The influence of limited visual sensing on the reynolds flocking algorithm," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pp. 138–145, IEEE, 2019. DOI:10.1109/IRC.2019.00028.
- [15] M. Sabbagh, M. H. Tanveer, A. Thomas, J. Faile, and M. Salman, "Real time voronoi-like path planning using flow field and a," in *2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*, pp. 103–107, 2020. DOI:10.1109/HONET50430.2020.932283.
- [16] G. Pentheny, "Advanced techniques for robust, efficient crowds," in *Game AI Pro 360*, pp. 137–146, CRC Press, 2019. DOI:10.1201/9780429055096.
- [17] E. Emerson, "Crowd pathfinding and steering using flow field tiles," in *Game AI Pro 360: Guide to Movement and Pathfinding*, pp. 67–76, CRC Press, 2019. DOI:10.1201/9780429055096.
- [18] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*, pp. 1928–1935, Ieee, 2008. DOI:10.1109/ROBOT.2008.4543489.
- [19] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: highly parallel collision avoidance for multi-agent simulation," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 177–187, 2009. DOI:10.1145/1599470.1599494.



Yago Ávila Moré got the title of B. Sc. in Industrial Electronics and Automation Engineering in 2023 from the Universidad Politécnica de Madrid (UPM) (Spain). His interests are mainly focused on systems control, navigation, artificial intelligence and robotics. He is currently working at Perseo Techworks, a company dedicated to the development of autonomous unmanned vehicles.



Basil Mohammed Al-Hadithi got the title of B. Sc. in control and system engineering in 1983 and the M. Sc. in control and instrumentation engineering in 1988. He received a Ph.D. in process control and artificial intelligence in 2002 from Universidad Politécnica de Madrid (UPM) (Spain) with a thesis on analysis, design and stability of fuzzy slide-mode control systems. He is a full professor at UPM. His teaching activity covers control engineering and analogue electronics, being an author and co-author of seven textbooks and having supervised and co-supervised several B. Sc. final year projects and M. Sc. theses and 6 Ph.D. theses. He is a researcher at the Centre for Automation and Robotics UPM-CSIC. His interest is mainly focused on fuzzy control and slide mode control. He has several publications (JCR), book chapters and conference papers. Moreover, he has participated in several research projects and industrial contracts with companies. He is a board member and reviewer of several international scientific societies and International journals in modelling and designing control systems.



Víctor Cadix Martín got the title of B. Sc. in Industrial Electronics and Automation Engineering in 2019 and the M. Sc. In Automation and Robotics in 2021 from the Universidad Politécnica de Madrid (UPM) (Spain). His interests are mainly focused on systems control, navigation, legged robots and swarm robotics. He is currently working at Perseo Techworks.