UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES

MÁSTER EN ROBÓTICA Y AUTOMÁTICA

Inteligencia Artificial

Optimización de Controlador de Navegación mediante Algoritmos Genéticos

---

Autores:

Asil Arnous

Fernando Pérez Sancho

Raúl Chamorro

Grupo 2.

Madrid, 12 de enero de 2026

Resumen.

En este trabajo se aborda el desarrollo y la optimización de un sistema de control para un agente móvil autónomo en un entorno de simulación bidimensional. El objetivo principal es diseñar un controlador capaz de permitir al agente navegar de forma reactiva, evitando colisiones con obstáculos dinámicos y alcanzando objetivos distribuidos aleatoriamente en el entorno, utilizando únicamente información local y limitada obtenida a través de sensores de proximidad.

El sistema de control propuesto se basa en una Red Neuronal Artificial que traduce las lecturas sensoriales del agente en acciones de navegación, concretamente velocidad. A diferencia de los enfoques clásicos de entrenamiento supervisado, este trabajo emplea un Algoritmo Genético para optimizar los parámetros de la red neuronal, evitando así la necesidad de disponer de datos de conducción óptima previamente etiquetados. Los pesos y sesgos de la red neuronal se codifican como un vector de números reales, que actúa como genotipo dentro del proceso evolutivo.

La evaluación de cada individuo se realiza mediante su ejecución directa en la simulación, utilizando una función de score diseñada para recompensar comportamientos deseables, como el acercamiento a los objetivos y la evasión de obstáculos, y penalizar colisiones y situaciones de riesgo. A lo largo de múltiples generaciones, el proceso evolutivo permite la aparición de comportamientos de navegación eficientes y robustos.

Los resultados obtenidos muestran que el enfoque de neuroevolución empleado es capaz de generar controladores que generalizan su comportamiento a diferentes configuraciones del entorno, demostrando la viabilidad de los Algoritmos Genéticos como método de optimización para controladores basados en redes neuronales en entornos dinámicos.

Contenido

1. Introducción.

En el ámbito de la Inteligencia Artificial, el diseño de agentes autónomos capaces de interactuar con su entorno de manera eficiente y adaptativa constituye una de las áreas de investigación más relevantes. Estos agentes deben ser capaces de percibir su entorno, tomar decisiones y actuar en consecuencia, incluso en situaciones donde la información disponible es incompleta, ruidosa o limitada. En particular, el problema de la navegación autónoma en entornos dinámicos representa un desafío significativo, ya que requiere combinar percepción, control y adaptación en tiempo real.

Tradicionalmente, los sistemas de control para navegación autónoma se han abordado mediante técnicas clásicas de control o mediante el entrenamiento supervisado de modelos de aprendizaje automático, lo que implica la disponibilidad de datos de referencia que describan un comportamiento óptimo. Sin embargo, en muchos escenarios reales resulta complejo o inviable definir de antemano dichos datos, especialmente cuando el entorno es dinámico o impredecible. En este contexto, los enfoques basados en aprendizaje evolutivo surgen como una alternativa adecuada para el desarrollo de comportamientos inteligentes sin necesidad de supervisión explícita.

Este trabajo se centra en la optimización de un controlador de navegación para un agente móvil autónomo en un entorno de simulación bidimensional. El agente debe desplazarse evitando colisiones con obstáculos móviles y alcanzando objetivos distribuidos aleatoriamente, basándose exclusivamente en información local proporcionada por sensores de proximidad. El comportamiento del agente es de carácter reactivo, es decir, las decisiones de navegación se toman en función del estado actual del entorno sin disponer de un mapa global ni de planificación a largo plazo.

El controlador del agente se implementa mediante una Red Neuronal Artificial (RNA), encargada de transformar las entradas sensoriales en acciones de control continuo. En lugar de emplear algoritmos de entrenamiento tradicionales, los parámetros de la red neuronal son optimizados mediante un algoritmo genético, siguiendo un enfoque de neuroevolución. Este método permite explorar el espacio de soluciones de forma eficiente y descubrir configuraciones de control que maximizan el rendimiento del agente dentro de la simulación.

El principal objetivo de este proyecto es analizar la viabilidad de los Algoritmos Genéticos como herramienta de optimización de controladores neuronales en tareas de navegación autónoma. Asimismo, se pretende estudiar el comportamiento emergente que surge a partir de la interacción entre el agente, el entorno y la función de score definida, evaluando la capacidad del sistema para generar soluciones robustas y generalizables.

2. Planteamiento del problema.

El proyecto se desarrolla en un entorno de simulación en 2D. En este entorno, el vehículo se puede mover libremente, apareciendo obstáculos dinámicos que se desplazan desde los bordes hacia el interior. Estos obstáculos, de color rojo, tienen un tamaño similar al vehículo y velocidades variables, generando un entorno impredecible que desafía la capacidad de reacción del controlador.

El entorno está diseñado de manera que toda la información percibida por el vehículo provenga únicamente de sensores locales de proximidad, sin disponer de un mapa global ni información anticipada sobre la posición de los obstáculos o del objetivo. Esta limitación simula escenarios reales donde los agentes autónomos deben tomar decisiones basadas únicamente en datos parciales y en tiempo real.

El agente considerado se desplaza en el plano bidimensional, cuya dinámica se aproxima a un movimiento continuo con velocidad controlable. Para interactuar con su entorno, el agente está equipado con un conjunto de sensores de proximidad tipo láser (LIDAR) que permiten medir distancias a obstáculos en diferentes direcciones.

El objetivo principal del agente es navegar evitando colisiones y alcanzar un objetivo representado por un rectángulo dentro del entorno. Este rectángulo se genera de manera aleatoria tras cada interacción exitosa. El agente es, por tanto, un sistema reactivo que basa sus decisiones únicamente en la información sensorial inmediata, sin planificación a largo plazo.

Es importante resaltar que el vehículo solo puede acceder a la información que le proporciona el sensor de proximidad laser y a la posición relativa del objetivo. No contará con un mapa global, ni las ubicaciones de los obstáculos, lo que le obligará a tomar decisiones en tiempo real basadas en datos incompletos.

La navegación se considera reactiva, ya que cada acción se calcula a partir del estado actual del agente y de su entorno, sin almacenar memoria de posiciones pasadas ni realizar predicciones complejas. Esta característica enfatiza la capacidad de la red neuronal para generalizar patrones de evasión y aproximación al objetivo**.**

El desarrollo de un controlador para navegación autónoma en entornos dinámicos presenta varios desafíos clave:

1. Limitación de información**:** El agente solo cuenta con lecturas locales de sensores, lo que hace que deba tomar decisiones basadas en datos incompletos. Esto aumenta la complejidad del problema y exige un controlador capaz de generalizar patrones de evasión y aproximación al objetivo sin depender de un mapa global.
2. Evitar colisiones en tiempo real**:** Los obstáculos se generan de manera aleatoria y se mueven por el entorno, por lo que el agente debe calcular continuamente nuevas trayectorias y ajustar su velocidad y dirección para no colisionar. Esto requiere un comportamiento reactivo eficiente y robusto.
3. Optimización de múltiples objetivos**:** El agente debe equilibrar dos objetivos simultáneos: avanzar hacia el objetivo y evitar obstáculos. Estos objetivos pueden entrar en conflicto, por lo que la función de score debe reflejar adecuadamente la prioridad de cada uno.

4. Espacio de búsqueda complejo**:** La combinación de sensores, velocidad y objetivos dinámicos crea un espacio de soluciones muy amplio y no lineal. Esto hace que la optimización de los parámetros de la red neuronal sea un desafío, especialmente sin supervisión directa.

3. Marco teórico.

En Inteligencia Artificial, un controlador inteligente es un sistema que permite a un agente tomar decisiones sobre sus acciones en función del estado del entorno. Estos controladores van más allá de reglas predefinidas y pueden adaptarse a situaciones cambiantes, aprender de experiencias pasadas o evolucionar mediante procesos de optimización.

Los controladores inteligentes se aplican a robots móviles, vehículos autónomos, drones y simulaciones, y pueden basarse en diversas técnicas, desde lógica difusa hasta redes neuronales y algoritmos evolutivos. Su objetivo es que el agente pueda navegar de manera eficiente y segura, adaptándose a entornos complejos y dinámicos.

Las Redes Neuronales Artificiales (RNA) son modelos inspirados en el cerebro humano, capaces de aprender relaciones no lineales entre entradas y salidas. Están formadas por capas de neuronas conectadas entre sí mediante pesos y sesgos, que determinan la influencia de cada entrada sobre la salida.

Una RNA típica consta de tres tipos de capas:

1. Capa de entrada: recibe los datos del entorno. En nuestro proyecto, estas entradas corresponden a los sensores de proximidad y la información relativa al objetivo.
2. Capas ocultas: procesan la información mediante combinaciones lineales de las entradas y funciones de activación no lineales. Estas capas permiten que la red aprenda patrones complejos.
3. Capa de salida: proporciona las acciones de control del agente, en este caso la velocidad.

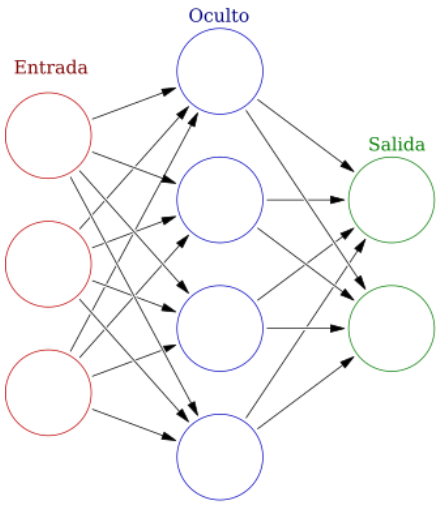

Figura 1. Modelo de Redes Neuronales Artificiales.

En el contexto de navegación autónoma, la RNA actúa como un mapa de sensores a acciones. Cada lectura de sensor se combina en la red para generar decisiones en tiempo real, permitiendo al agente adaptarse a obstáculos dinámicos y a objetivos cambiantes sin necesidad de planificación explícita.

Los Algoritmos Genéticos son métodos de optimización inspirados en la evolución natural. Permiten buscar soluciones en espacios complejos donde los métodos analíticos o de gradiente son difíciles de aplicar.

Los elementos principales de un Algoritmo Genético son:

1. Población: conjunto de soluciones candidatas.
2. Genotipo: representación de una solución como un vector de genes (en nuestro caso, los pesos y sesgos de la RNA).
3. Fitness: medida de calidad de cada individuo.
4. Selección: elección de individuos para reproducirse en base a su fitness.
5. Cruce y mutación: operadores que generan diversidad y permiten explorar nuevas soluciones.
6. Generaciones: iteraciones del proceso evolutivo, donde la población mejora progresivamente.

La combinación de Redes Neuronales y Algoritmos Genéticos, conocida como neuroevolución, permite:

- Optimizar controladores sin necesidad de datos de entrenamiento supervisado.
- Generar comportamientos emergentes adaptativos.
- Explorar un espacio de soluciones amplio y no lineal.

Este enfoque es especialmente adecuado para entornos dinámicos y parcialmente medibles, como el escenario propuesto en este proyecto.

La neuroevolución es un enfoque que combina Redes Neuronales Artificiales con Algoritmos Genéticos para optimizar automáticamente los parámetros de la red (pesos y sesgos) sin necesidad de entrenamiento supervisado.

En este paradigma, cada individuo de la población representa un controlador neuronal completo, codificado como un vector de números reales. Cada número corresponde a un peso o sesgo de la red, y la población evoluciona a lo largo de múltiples generaciones aplicando selección, cruce y mutación.

Este enfoque permite explorar soluciones que podrían ser difíciles de encontrar mediante técnicas tradicionales de aprendizaje supervisado, especialmente en problemas donde no se dispone de datos de referencia óptimos o el espacio de soluciones es altamente no lineal.

La principal ventaja de esta técnica es la robustez frente a entornos dinámicos. Al evaluar el fitness en escenarios simulados, la red aprende a adaptarse a obstáculos y objetivos cambiantes, lo cual es perfecto para este proyecto. Además, con esta técnica, el rendimiento mejora con el tiempo de simulación.

4. Arquitectura del sistema.

La simulación está implementada en Python utilizando la librería Pygame para la representación gráfica, permitiendo visualizar en tiempo real el comportamiento de los agentes, obstáculos y objetivos.

El sistema consta de cuatro componentes principales: vehículo, obstáculo, controlador y población. Todo ello reunido en una rama principal (**main.py**).

El vehículo (**car.py**) se encarga de navegar por el entorno de simulación. Este deberá moverse de una posición a otra. Se modela como una caja rectangular con movimiento continuo. Su estado interno está caracterizado por una posición (x, y), un ángulo que determina su orientación, velocidad y una puntuación. Este último sirve para el aprendizaje del algoritmo genético

El coche ajusta su dirección basándose en la información obtenida de los sensores y su posición relativa al objetivo. En cada frame, calcula su movimiento, actualiza los sensores y puntuaciones y se redibuja su posición en la pantalla.

El controlador (**controller.py**) transforma las lecturas del sensor de proximidad en acciones de navegación (velocidad). Se inicializa una población de controladores y cada controlador evalúa ejecutándolo en la simulación en función de la distancia recorrida. Se penalizan las colisiones y se premia acercarse al objetivo. El ciclo se repite hasta alcanzar un criterio de parada (número máximos de generaciones o desempeño satisfactorio).

Está implementado con la RNA con la siguiente arquitectura:

- Capa de entrada: 10 neuronas que corresponden a los datos de los sensores de proximidad y la información relativa al objetivo (ángulo y distancia).
- Capas ocultas: tres capas con tamaños decrecientes (64, 32 y 16 neuronas) que permiten capturar relaciones complejas entre las entradas y las acciones.
- Capa de salida: 2 neuronas que determinan la dirección y la velocidad del coche.

Al principio, los pesos de estas capas son aleatorios y se van optimizando mediante los algoritmos genéticos, adaptando sus parámetros para maximizar la puntuación del vehículo.

Por último, la población (**population.py**) se encarga de implementar un algoritmo genético completo, carga los pesos y se ocupa de las redes neuronales. Contiene la siguiente información:

- Población: conjunto de N individuos (controladores).
- Score o fitness: puntuación obtenida por cada controlador tras ejecutar la simulación, basada en puntuar por distancia recorrida y aproximación al objetivo, penalizaciones por colisiones y aproximación al obstáculo.
- Operadores evolutivos:
    - Selección: se eligen los individuos con mejor fitness para reproducirse.
    - Cruce: se combinan partes de dos controladores para generar un nuevo individuo.

- Mutación: se aplican pequeñas modificaciones aleatorias gaussianas a algunos genes para introducir diversidad.

Cada generación produce nuevos individuos, reemplazando parcialmente a la población anterior y permitiendo la aparición gradual de comportamientos de navegación más eficientes.

También habría que mencionar la generación de obstáculos (**obstacle.py**). Estos se crean en los extremos de forma aleatoria y se mueven hacia el otro a una velocidad constante.

En resumen:

1. main.py: punto de entrada del programa. Gestiona la simulación, la interfaz gráfica, la selección de modo de ejecución (entrenamiento o simulación del mejor controlador) y la ejecución del ciclo evolutivo.
2. car.py: define el agente, sus sensores, actualización de estado, score y dibujo en pantalla. Se comunica con el controlador neuronal para recibir acciones y con el entorno para detectar colisiones.
3. controller.py: implementa la Red Neuronal y proporciona funciones para el avance, obtención y modificación de pesos, y guardado/carga de modelos.
4. population.py: gestiona la población de controladores, aplica el Algoritmo Genético (selección, cruce y mutación) y coordina la evolución de las generaciones.
5. obstacle.py: gestiona los obstáculos del entorno, incluyendo posición, velocidad, actualización y dibujo en pantalla.

La implementación entera se puede encontrar en el siguiente repositorio:

https://github.com/fperezs/TrabajoIA

5. Modos de ejecución.

En cuanto al modo de ejecución, al iniciar el programa, te da la opción de entrenar o mostrar el mejor controlador (Figura 2).

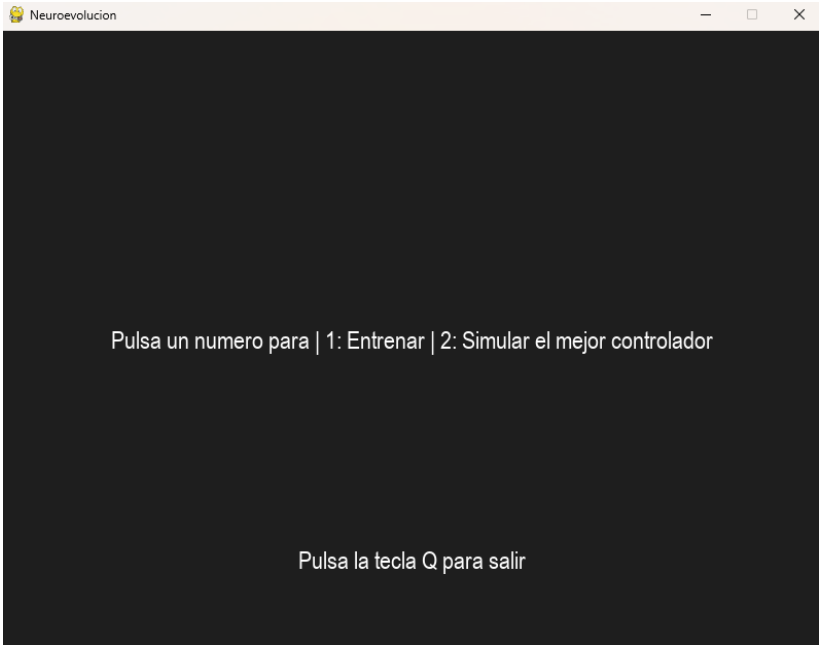

Figura 2. Selección de modo.

Al simular el mejor controlador (Figura 3), un único vehículo alcanza distintos objetivos (cuadrados verdes) evitando los obstáculos, haciendo crecer su score. Toma los valores almacenados en *wights_final.npy* para el control del coche. La velocidad es normal.

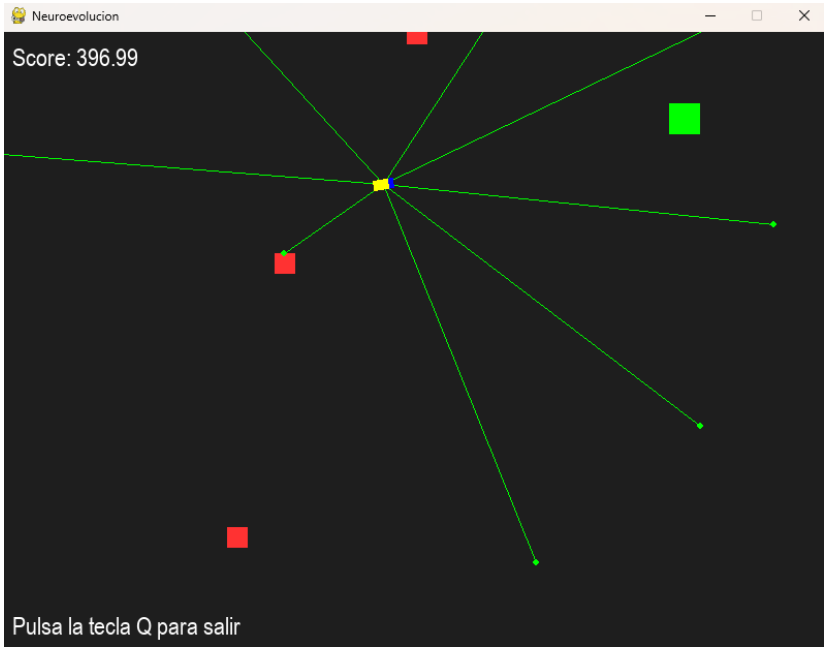

Figura 3. Modo Mejor controlador.

Al seleccionar la opción de entrenamiento, el sistema dispone de tres modos de ejecución diferentes:

1. Modo generación: visualización de todos los agentes vivos en la población. Dibuja obstáculos y la trayectoria de todos los coches (Figura 4).
2. Modo rápido: acelera el tiempo de entrenamiento mediante la eliminación de la simulación visual (Figura 5).
3. Modo líder: muestra sólo al agente de la generación con mayor score (mejor controlador). Permite analizar comportamiento y sensores del agente en detalle (Figura 6).

Se puede alternar entre los distintos modos de ejecución mediante la tecla TAB. Además, durante el entrenamiento es posible guardar en cualquier momento el mejor controlador obtenido hasta ese instante (checkpoint), finalizar el entrenamiento guardando dicho controlador, o cancelar el entrenamiento sin guardar.

Independientemente del modo de entrenamiento activo, se genera un archivo *training_metrics.csv* que almacena las métricas utilizadas para analizar la evolución del modelo al finalizar cada generación. Dichas métricas son:

- Score_max: puntuación del mejor individuo de la generación.
- Score_min: puntuación del peor individuo de la generación.
- Score_promedio: puntuación media de la generación.
- SD_score: desviación estándar de la puntuación de la generación.
- Tvida_max: tiempo de vida del individuo más duradero de la generación.
- Tvida_min: tiempo de vida del individuo menos duradero de la generación.
- Tvida_promedio: tiempo de vida medio de la generación.
- SD_tvida: desviación estándar del tiempo de vida de la generación.

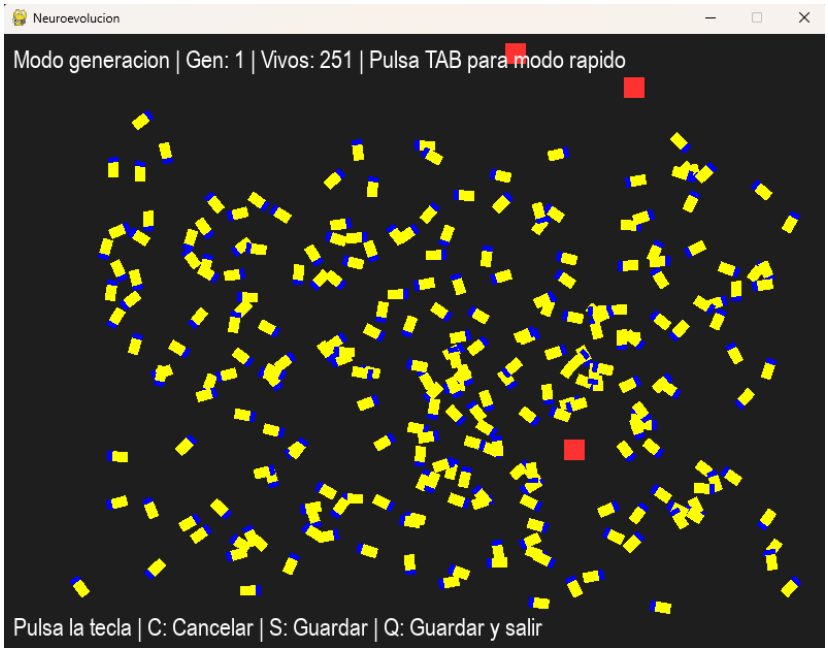

Figura 4. Entrenamiento en modo generación.

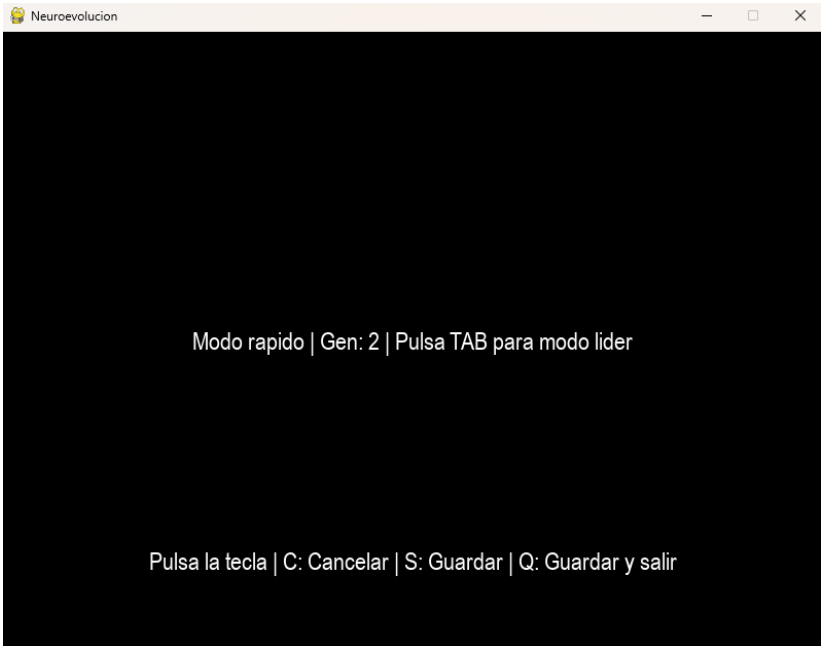

Figura 5. Entrenamiento en modo rápido.

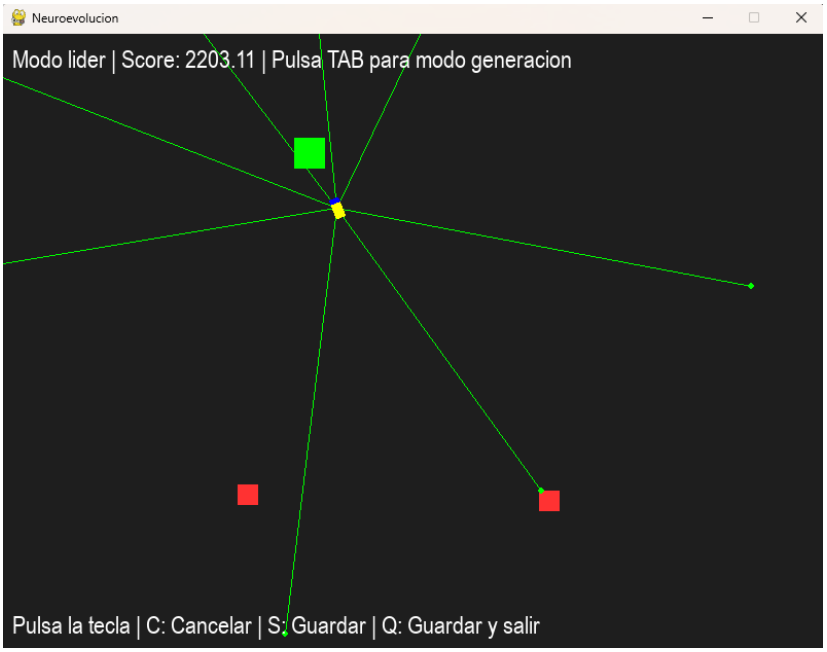

Figura 6. Entrenamiento en modo líder.

6. Resultados.

Tras la implementación del sistema y la aplicación del algoritmo genético para el entrenamiento de la red neuronal, se realizaron diversas ejecuciones con el objetivo de observar la evolución del comportamiento de los agentes y evaluar la efectividad del enfoque propuesto. Los resultados obtenidos permiten analizar tanto el proceso de aprendizaje como las estrategias emergentes desarrolladas por los vehículos autónomos.

En las primeras generaciones, el comportamiento de los coches es mayoritariamente errático. Al iniciarse con pesos aleatorios, los agentes tienden a moverse sin una dirección clara (o quedándose quietos sin moverse), colisionando con los obstáculos en pocos segundos o saliendo de los límites del escenario. En esta fase inicial, las puntuaciones obtenidas son bajas y existe una gran variabilidad entre individuos, lo cual es esperable en sistemas evolutivos sin conocimiento previo del entorno.

A medida que avanzan las generaciones, se observa una mejora progresiva en el rendimiento medio de la población. Los coches comienzan a mantenerse activos durante más tiempo, evitando colisiones inmediatas y mostrando trayectorias más suaves y coherentes. Este progreso se debe a la selección de los individuos con mayor puntuación, cuyos pesos neuronales se transmiten a la siguiente generación mediante los operadores genéticos. Como consecuencia, el sistema va descartando comportamientos claramente ineficientes y reforzando aquellos que contribuyen a una mayor supervivencia y mejor navegación. Se debe señalar que, en fases iniciales del desarrollo, antes de incluir un objetivo para los agentes y la función de fitness estaba basada en el tiempo de supervivencia del agente, muchos dejaron de moverse, una actitud que interpretaba el algoritmo como buena. Esto señala la dificultad de elegir una función de fitness que evite comportamientos no favorables para el problema.

Un aspecto especialmente relevante es la aparición de comportamientos emergentes sin haber sido programados explícitamente. Algunos agentes aprenden a reducir la velocidad cuando los sensores detectan obstáculos cercanos, mientras que otros desarrollan estrategias de estar girando constantemente para compensar la falta de sensores (lo cual se ve en tiempo de simulación). Asimismo, se aprecia una mejora en la orientación hacia el objetivo, lo que indica que la red neuronal es capaz de interpretar correctamente la información relativa a la posición del destino y convertirla en decisiones de control adecuadas.

La puntuación acumulada de los agentes refleja este proceso de aprendizaje. Los individuos que logran equilibrar la evitación de obstáculos con el desplazamiento hacia el objetivo obtienen valores de score significativamente superiores. Esto provoca que, con el paso de las generaciones, la diferencia entre los mejores individuos y los peores se haga más evidente, evidenciando la presión selectiva ejercida por el algoritmo genético.

Cabe destacar que, aunque el sistema muestra una mejora clara, el comportamiento no llega a ser perfectamente óptimo ni determinista. En algunas ejecuciones se producen estancamientos temporales, donde la mejora del score se ralentiza durante varias generaciones. Este fenómeno es común en algoritmos genéticos y está relacionado con la pérdida de diversidad genética o la convergencia hacia soluciones subóptimas. No obstante, pequeñas mutaciones permiten en muchos casos escapar de estas situaciones y continuar el proceso de mejora.

En conjunto, los resultados obtenidos demuestran que la combinación de redes neuronales

artificiales y algoritmos genéticos resulta eficaz para abordar el problema de la navegación autónoma en entornos dinámicos. A partir de reglas simples y sin supervisión directa, el sistema es capaz de generar comportamientos complejos y adaptativos, validando el enfoque propuesto desde un punto de vista práctico y experimental.

En la siguiente tabla (Tabla 1) se muestran las métricas del entrenamiento en la primera generación y en la última.

Generación	**Score máximo**	**Score mínimo**	**Promedio**	**SD score**	**Tiempo de vida máximo**	**Tiempo de vida mínimo**	**Promedio**	**SD tiempo de vida**
1	1659	-102	37	201	1764	18	202	234
1000	24037	-19	4711	4163	7570	32	1461	1268

Tabla 1. Métricas de entrenamiento.

Se puede observar que la diferencia es evidente. Principalmente, las puntuaciones obtenidas son mayores, demostrando que los vehículos consiguen alcanzar más objetivos evitando los obstáculos. También los tiempos de vida son mayores, evidenciando que los coches no colisionan, ni se salen del mapa diseñado.

Si se viera esta evolución a lo largo de cada generación, el resultado serían las siguientes figuras (Figura 6 y Figura 7).

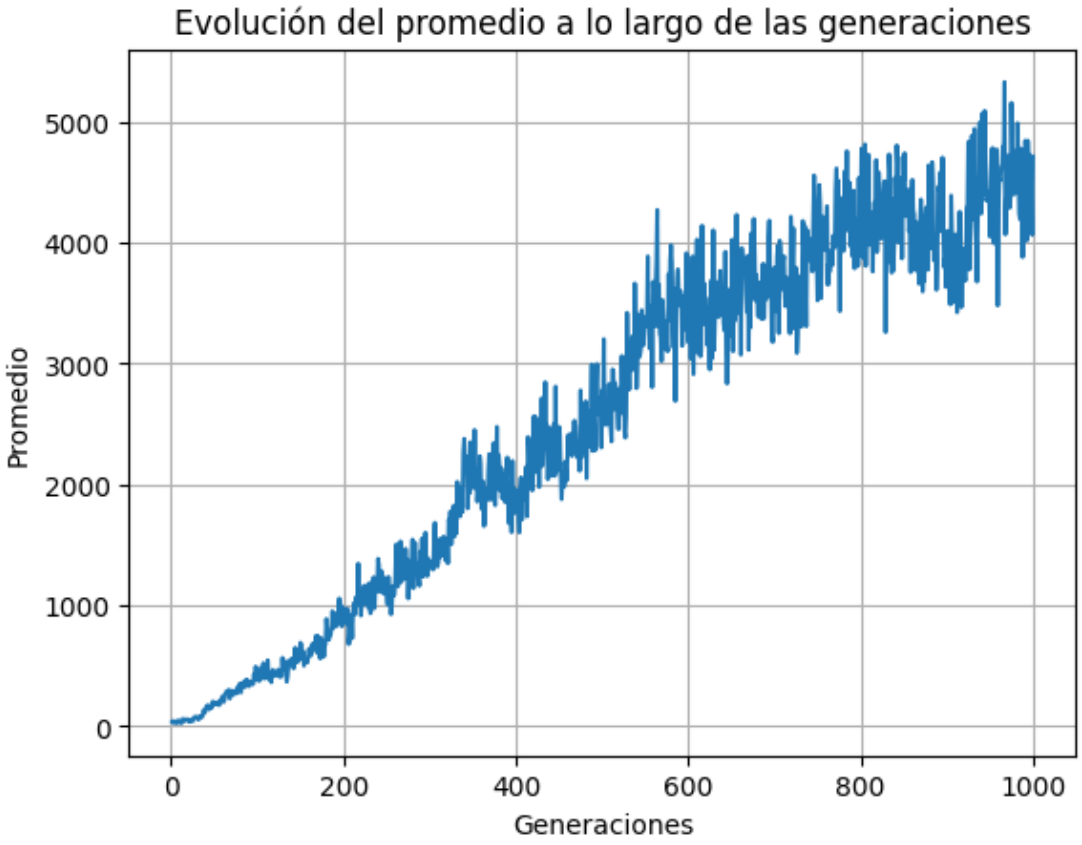

Figura 6. Generación vs Score_Promedio.

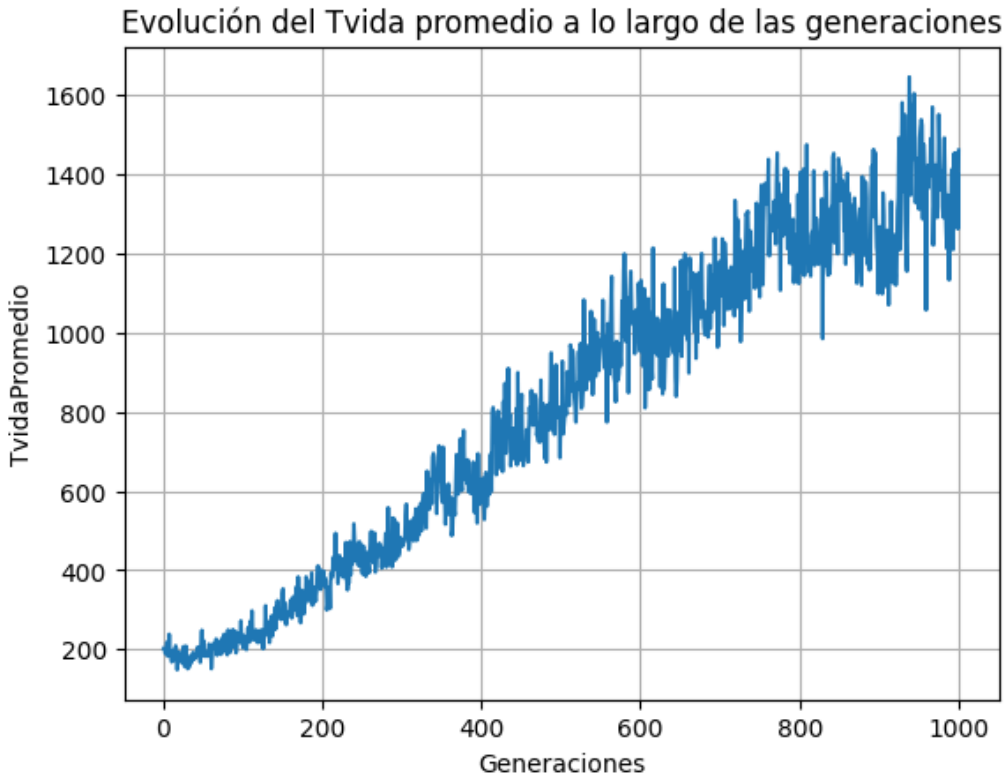

Figura 7. Generación vs Tvida_Promedio.

Estas gráficas seguirán creciendo puesto que ni la puntuación ni el tiempo de vida tienen limite. El robot, en cuanto aprende a esquivar obstáculos y alcanzar objetivos encadenará un objetivo tras otro, haciendo crecer su score. Lo mismo pasa con el tiempo de vida.

7. Conclusiones.

El proyecto ha permitido validar la eficacia de la neuroevolución como una técnica adecuada para el desarrollo de controladores autónomos en entornos 2D dinámicos. A lo largo del trabajo se ha comprobado que la combinación de RNA y Algoritmos Genéticos posibilita la mejora progresiva del comportamiento de un agente móvil sin necesidad de datos de entrenamiento supervisado, lo que constituye una ventaja significativa frente a enfoques tradicionales. Los controladores obtenidos son capaces de navegar de forma fluida, evitar obstáculos de manera reactiva y aproximarse a objetivos dentro del entorno de simulación.

Uno de los aspectos más relevantes observados es la aparición de comportamiento emergente. Los agentes desarrollan estrategias de navegación adaptativas, como maniobras de evasión o ajustes dinámicos de la velocidad, sin que dichas conductas hayan sido programadas explícitamente. Este fenómeno pone de manifiesto la capacidad de la neuroevolución para generar soluciones complejas a partir de reglas simples y procesos de optimización evolutiva. Asimismo, los controladores finales muestran una notable capacidad de generalización, ya que mantienen un rendimiento estable frente a diferentes configuraciones del entorno y ante obstáculos generados de forma aleatoria, lo que indica un comportamiento robusto y adaptable.

La función de evaluación utilizada ha resultado adecuada para guiar el proceso evolutivo, ya que combina recompensas asociadas al avance hacia el objetivo con penalizaciones por colisiones o proximidad excesiva a los obstáculos. Este enfoque permite discriminar de manera efectiva entre controladores de mayor y menor calidad, favoreciendo la selección de soluciones más eficientes a lo largo de las generaciones.

No obstante, el proyecto presenta algunas limitaciones. La simulación se desarrolla en un entorno 2D simplificado con obstáculos rectangulares, lo cual no refleja completamente la complejidad de escenarios reales. Además, el sistema de percepción del agente se limita a sensores de proximidad y a la información relativa al objetivo, sin considerar otras fuentes de información más ricas, como visión artificial. Por otro lado, el proceso evolutivo requiere un número elevado de generaciones para alcanzar resultados óptimos, lo que puede implicar un coste computacional significativo.

A partir de estos resultados, se abren diversas líneas de trabajo futuro. Entre ellas destacan la incorporación de entornos de simulación más realistas, incluyendo escenarios 3D, la ampliación del conjunto de sensores para mejorar la percepción del agente y la optimización de los parámetros evolutivos con el objetivo de acelerar la convergencia. También resulta especialmente interesante explorar enfoques híbridos que combinen la neuroevolución con técnicas de aprendizaje por refuerzo, así como el estudio de entornos multiagente en los que se analicen comportamientos cooperativos o competitivos.

En conclusión, este trabajo demuestra que la neuroevolución constituye una estrategia eficaz y flexible para la optimización de controladores autónomos en entornos reactivos. Los resultados obtenidos evidencian la capacidad de los agentes para aprender comportamientos adaptativos y generalizables, y muestran el potencial de este enfoque como base para aplicaciones futuras en robótica móvil y sistemas de navegación autónoma.

8. Reparto de tareas.

Asil Arnous: Controlador neuronal y algoritmo genético. Diseño e implementación de la RNA utilizada como controlador. Definición de la arquitectura de la red. Implementación del Algoritmo Genético. Gestión del guardado y carga de los mejores controladores. Documentación relacionada a la arquitectura del sistema.

Fernando Pérez Sancho: Diseño del sistema de evaluación del rendimiento del agente. Recopilación de datos (score, generaciones, promedios…). Implementación del sistema de obstáculos dinámicos y generación aleatoria. Pruebas y depuración de código. Documentación relacionada al modo de ejecución y resultados.

Raúl Chamorro: Diseño e implementación del entorno de simulación 2D. Integración gráfica y visualización del comportamiento del agente (sensores de proximidad, velocidades, colisiones…) durante la simulación. Documentación teórica del informe.