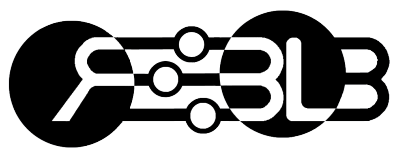
Emergencia de comunicación en robótica colectiva.

Rafael Sendra Arranz
r.sendra@upm.es



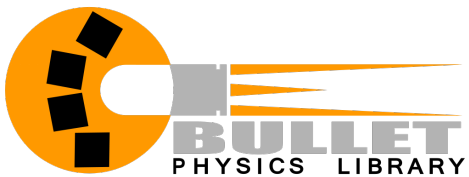
Mereli - Introducción

Mereli: Multi-Environment Robotics simulator with Evolution and Learning Implementations

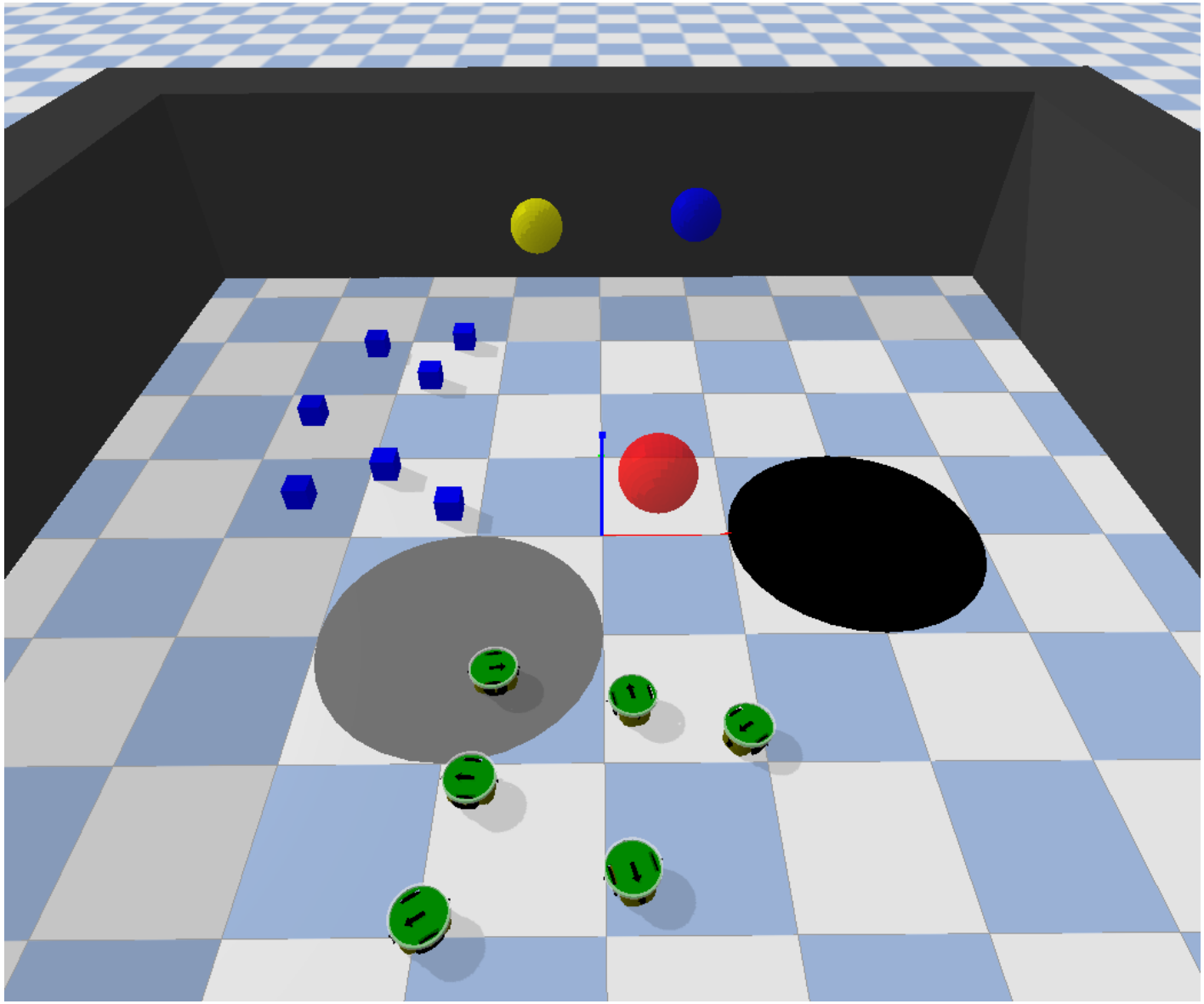
Mereli es un simulador de robótica colectiva escrito en Python y siendo desarrollado en Robolabo.

Sus principales características son:

- Simulaciones físicas de cuerpos rígidos realistas y detección de colisiones precisas.
- Gráficos 3D.
- Implementación de diferentes tipos de redes neuronales (MLP, CTRNN, SNN).
- Implementación de algoritmos evolutivos (GA, NES, NEAT, ...).
- Paralelización de algoritmos mediante MPI (Message Passing Interface).



Mereli - Ejemplo de entorno



Mereli - Ventajas y Desventajas

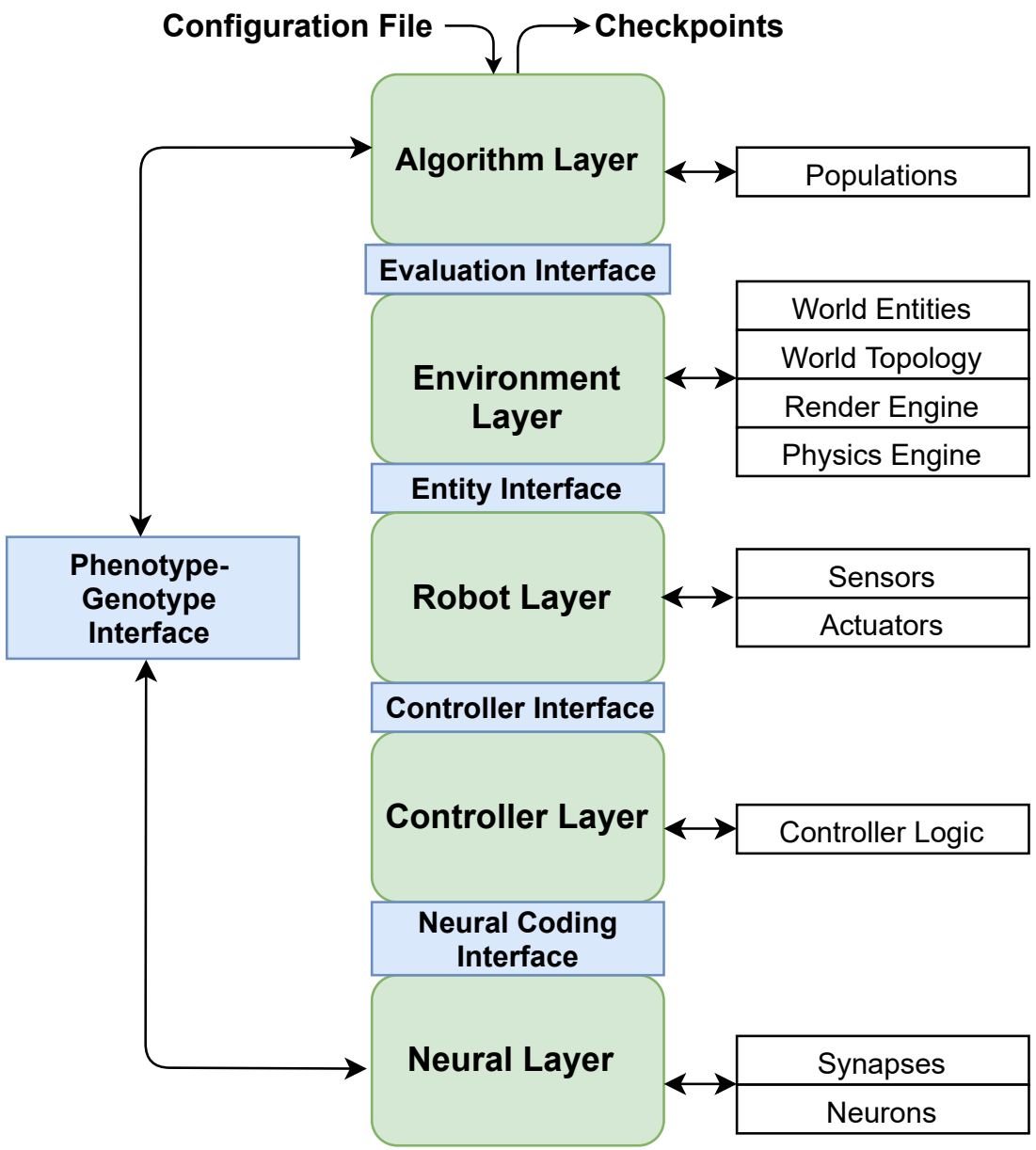
Puntos fuertes de mereli:

- Código altamente modular y escalable.
- Incorporación de cualquier tipo de robot.
- Fácil integración con muchas librerías potentes de deep learning y reinforcement learning (tensorflow, pytorch, stable-baselines, ...).
- Realismo de las simulaciones.

Talón de Aquiles:

- Ejecución lenta frente a otros simuladores desarrollados en C o C++ (IRSIM).

Mereli - Diagrama general

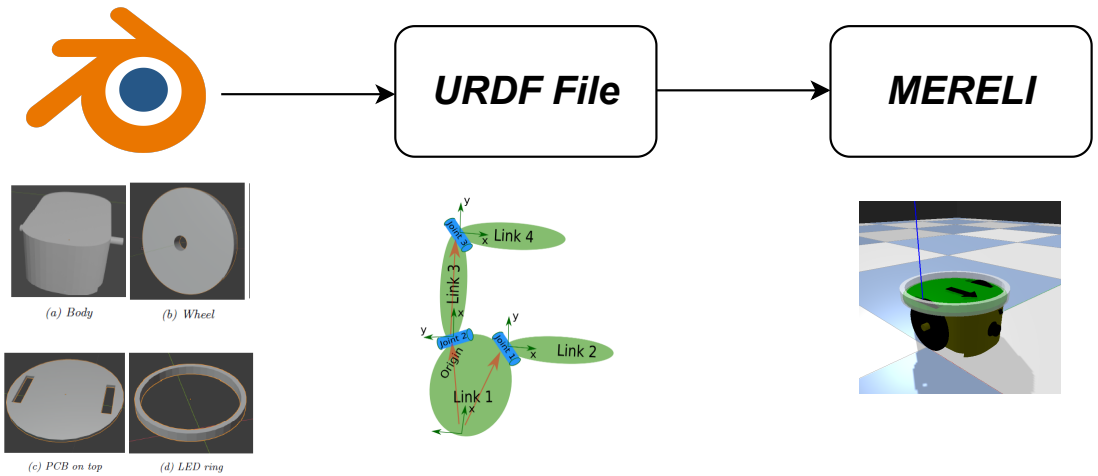


Mereli - Creación de robots

Practicamente cualquier robot puede ser simulado y controlado en mereli. Por ejemplo, podemos instanciar brazos robóticos, robots hexápodos o, incluso, robots humanoides.

No obstante, es necesario realizar los siguientes pasos previos:

1. Crear el modelo 3D del robot en Blender.
2. Representar al robot (enlaces, articulaciones, texturas, colisiones, etc) mediante ficheros Unified Robot Description Format (URDF).
3. Instanciar el robot dentro del simulador.



Mereli - API vs Fichero de configuración

El simulador se puede ejecutar de dos formas **aternativas**:

Mediante ficheros de configuración

- Especifican todos los parámetros de la simulación y de la optimización.
- Fichero de configuración formato JSON.
- Tenemos que ejecutar mereli siempre mediante main.py

Mediante la API

- Podemos usar mereli dentro del código.
- Creación de simulación más compleja.
- Posibilidad de usar mereli dentro de otros proyectos.

Ejemplo esquivador de obstaculos

Archivo de configuración

```
"world": {
    "name" : "square_arena",
    "arena_params" : {
        "height": 3,
        "width" : 3
    },
    "engine" : "pybullet",
    "physics_dt" : 0.04,
    "T_control" : 0.04,
    "objects" : {
        "robotA" : {
            "type" : "epuck",
            "num_instances" : 5,
            "controller" : "basic_obstacle_avoider",
            "sensors" : {
                "distance_sensor" : {"n_sectors" : 8, "range" : 0.7}
            },
            "actuators" : {
                "joint_velocity_actuator" : {"joint_ids" : [0, 1], "max_velocity" : 4}
            },
            "initializers" : {
                "positions" : {"name" : "random_uniform",  "params" : {"low": [-1,-1], "high" : [1,1], "size" : 2}},
                "orientations" : {"name" : "random_uniform",  "params" : {"low": 0, "high" : 6.28}}
            },
            "perturbations" : {},
            "params" : {"trainable" : true}
        }

    }
}
```

Robótica de Enjambre

Swarm Robotics

La robótica de enjambre es un campo de investigación, que combina ideas de la robótica y la inteligencia artificial, estudia el uso de una gran cantidad de robots simples que cooperan con el fin de resolver tareas complejas.

Los sistemas de robótica de enjambre deben cumplir las siguientes características:

- Autonomía
- Gran número de robots
- Homogeneidad
- Robots simples e ineficientes
- Comunicación y percepción del entorno local

Cuando no se cumplen estas propiedades, se suele denominar robótica colectiva en vez de robótica de enjambre.

Emergencia de la comunicación

En el contexto de la robótica colectiva, existen diversos tipos de comunicación:

- Estigmergia (Comunicación mediante el entorno)
- Comunicación a través del estado
- Comunicación directa

Podemos distinguir dos tipos de comunicación directa:

- **Comunicación abstracta**: toda la información reside en el mensaje.
- **Comunicación situada**: el contexto del entorno es necesario para entender el mensaje.

Emergencia de la comunicación

Pero deseamos que la semantica y el tipo de comunicación emerja a partir de un proceso de optimización (por ejemplo evolutivo).

Problemas de la emergencia de comunicación mediante evolución artificial

La emergencia de comunicación mediante algoritmos de optimización (evolutivos, RL, ...) es un tema muy complejo y difícil de conseguir.

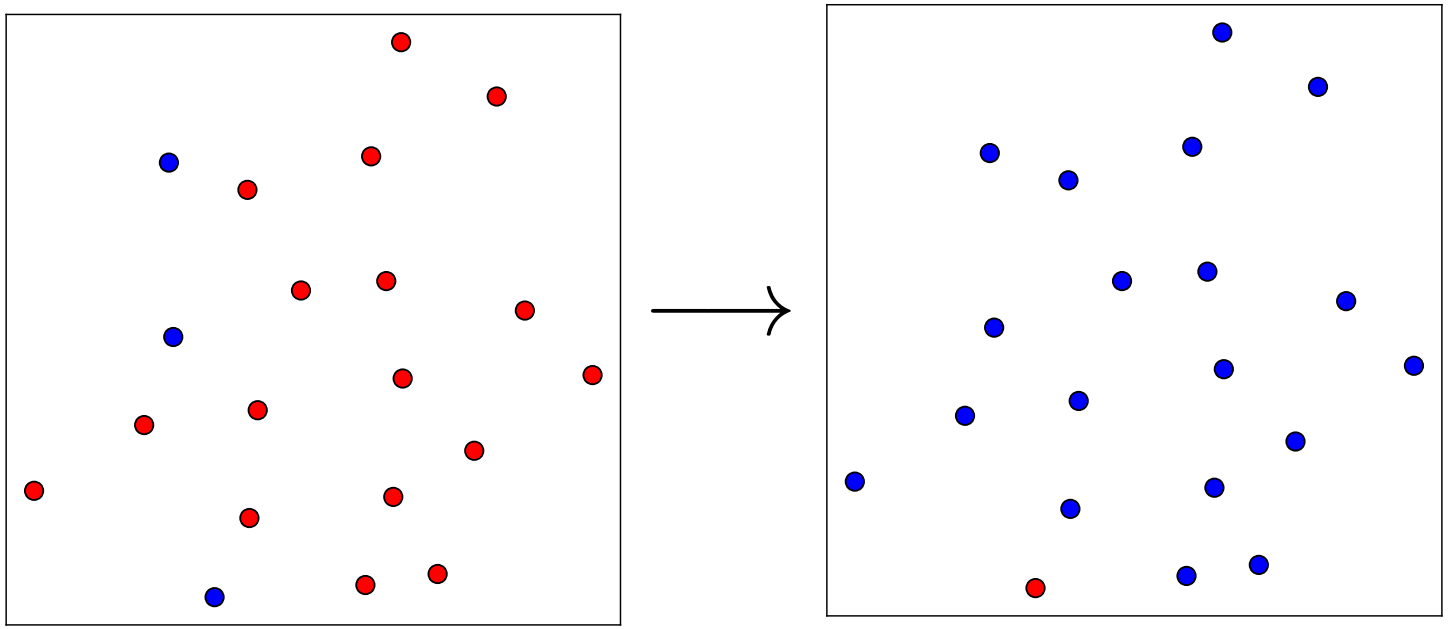
Si la tarea puede resolverse sin necesidad de cooperación entre robots, lo más posible es que no emerja ningún tipo de comunicación.

Los aspectos a los que debemos prestar más atención son:

- Elección de una tarea adecuada que requiera cooperación.
- Formulación de la función de fitness.
- Sistema de comunicación a disposición de los robots.
- Algoritmo evolutivo y red neuronal.

Elección del líder del enjambre

El enjambre de robots debe comunicarse para seleccionar a un único líder del grupo.



Elección del líder - Función de fitness

Fitness score estimate

$$\hat{F} = \frac{1}{N_E\, T_E} \sum_{n=1}^{N_E} \sum_{k=1}^{T_E} f(\mathscr{G}, k, n)$$

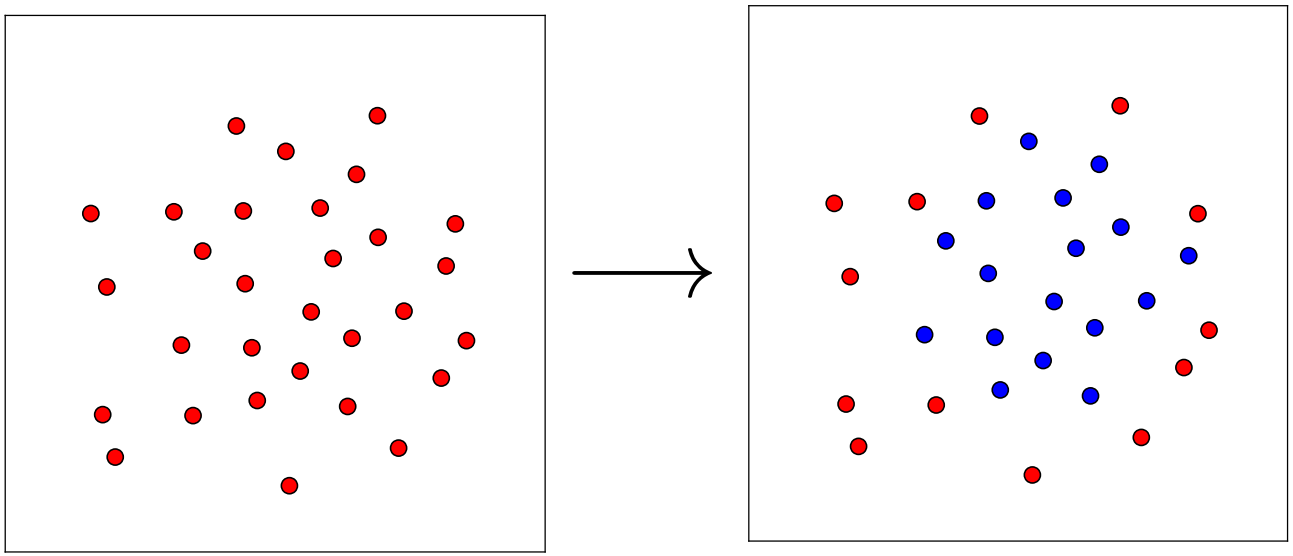
1. **Exp. A: Leader Selection**

$$f_A(\mathscr{G}, k) = \mathbf{w}(k)^\top \mathbf{a}_{LED}(k)$$

$$\mathbf{w}(k) = \begin{cases} \min\{\mathbf{w}(k-1) + 0.1\,\mathbf{a}_{LED}(k),\, 5\} & \text{If } \sum_{r\in\mathcal{R}} \mathbf{a}_{r,LED}(k) = 1 \\ & \text{and } \mathbf{a}_{LED}(k) = \mathbf{a}_{LED}(k-1) \\ 0.1\,\mathbf{a}_{LED}(k) & \text{If } \sum_{r\in\mathcal{R}} \mathbf{a}_{r,LED}(k) = 1 \\ & \text{and } \mathbf{a}_{LED}(k) \neq \mathbf{a}_{LED}(k-1) \\ (\underbrace{0, \ldots, 0}_{R \text{ times}})^\top & \text{Otherwise} \end{cases}$$

Identificación de la frontera

Los robots deben decidir si se encuentran en la frontera del enjambre o en el interior.

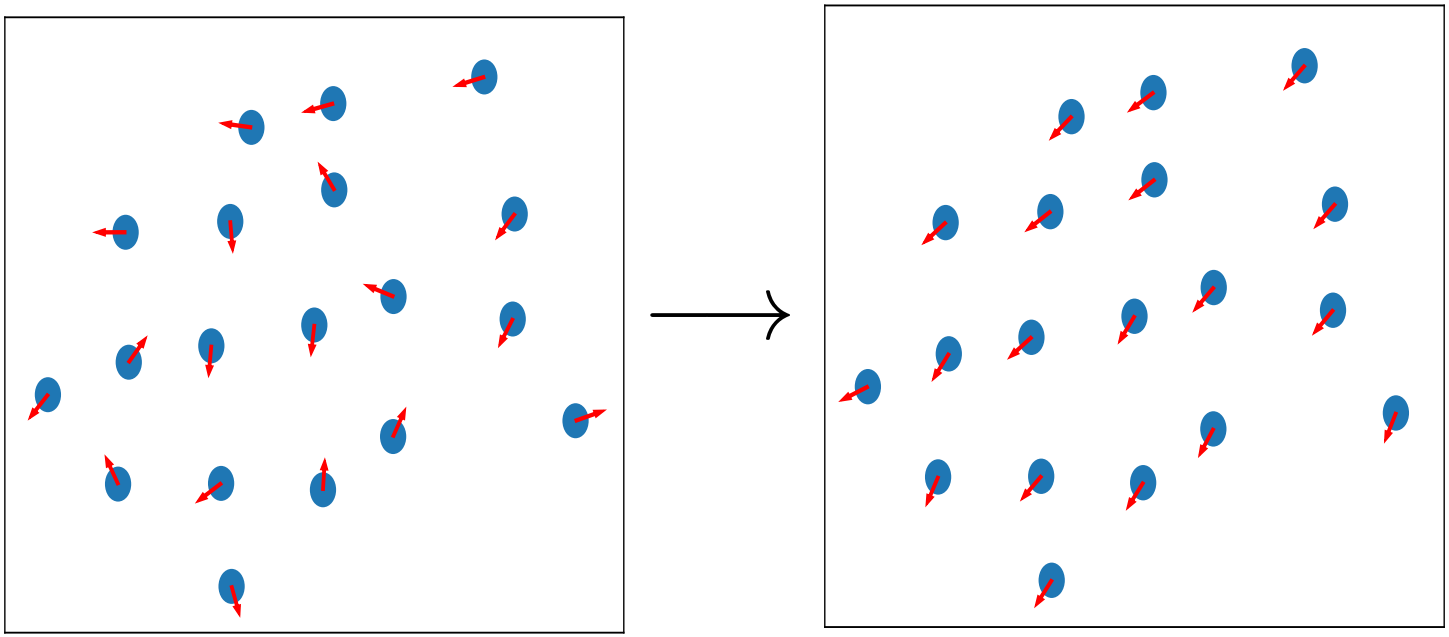


Función de fitness:

$$f_B(\mathscr{G}, k) = \frac{1}{R_B\, R_I} \underbrace{\left(\sum_{r \in \mathcal{R}_B} \mathbf{a}_{r,LED}(t) \right)}_{\text{True Positive Rate}} \cdot \underbrace{\left(R_I - \sum_{r \in \mathcal{R}_I} \mathbf{a}_{r,LED}(t) \right)}_{\text{True Negative Rate}}$$

Consenso en la orientación de los robots

Todos los robots deben apuntar en la misma dirección.



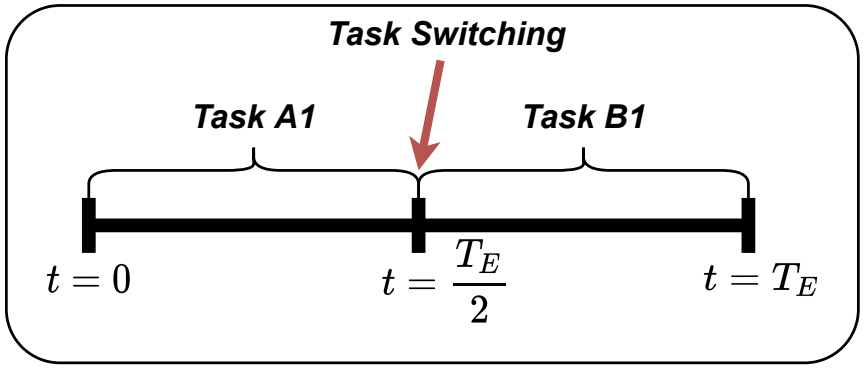
Función de fitness:

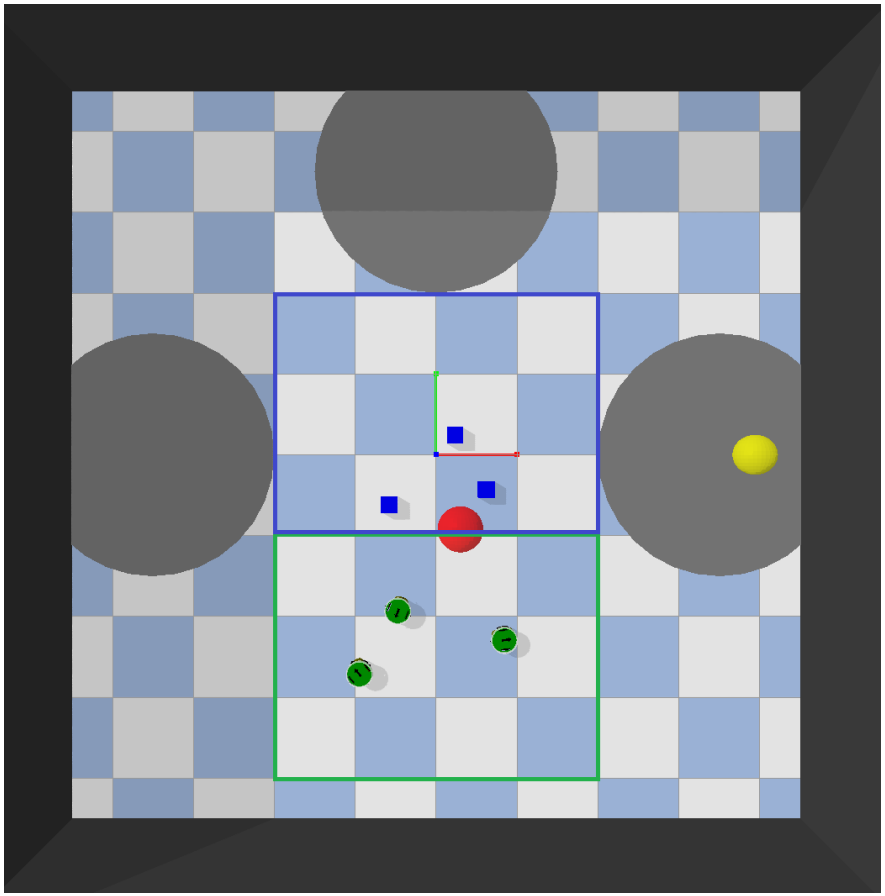
$$f_C(\mathcal{G},k)=\underbrace{\frac{1}{R^2}\sum_{r\in\mathcal{R}}\left(1-\frac{\min\{2\pi-|\theta_r(k)-\overline{\theta}(k)|,|\theta_r(k)-\overline{\theta}(k)\}}{\pi}\right)}_{\text{Orientation consensus}}\cdot\overbrace{\left(R-\|\mathbf{a}_{wr}(k)\|_1\right)}^{\text{Reduce rotation speed}}$$

Cambio de tarea y multi-tarea - Experimento 1

Experimento 1

Estudia los problemas del **cambio de tarea y la multi-tarea** en robótica colectiva usando controladores neuronales CTRNN optimizados mediante el algoritmo NEAT.





Las dos tareas a resolver por los robots son:

- **Task A1**: perseguir a la fuente de luz roja móvil.
- **Task B1**: transportar el máx. número de cubos azules al area gris con la luz amarilla encima.

Cambio de tarea y multi-tarea - Experimento 2

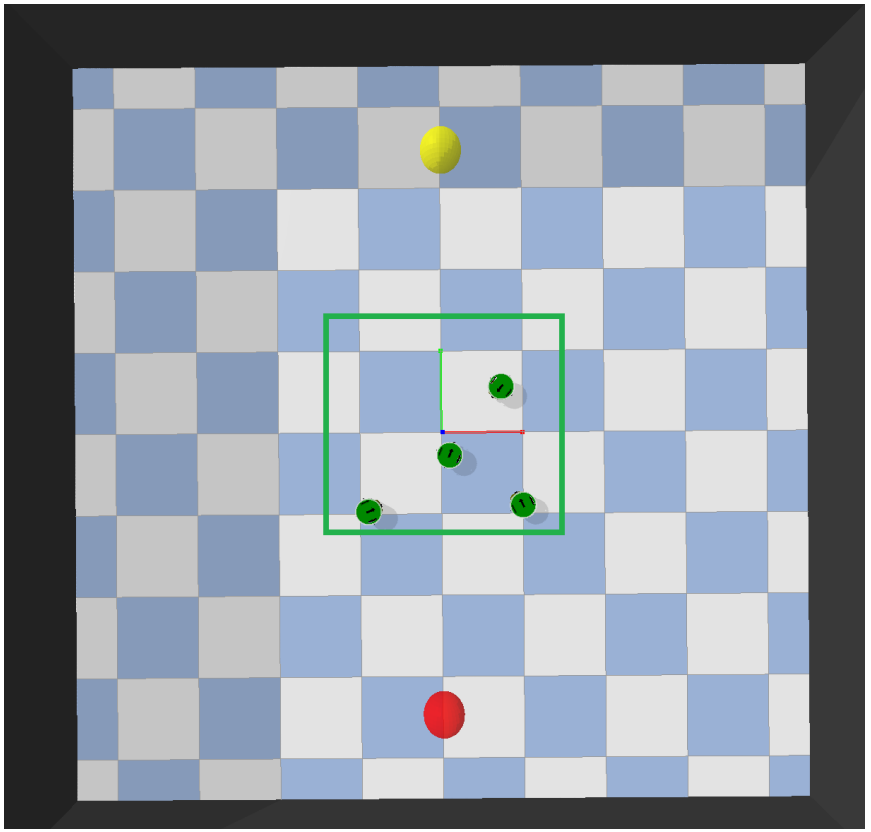
Experimento 2

Estudia la **emergencia de communicación** en robótica colectiva usando controladores neuronales CTRNN optimizados mediante el algoritmo NEAT.

Las dos tareas a resolver por los robots son:

- **Task A2**: perseguir a la fuente de luz roja móvil.
- **Task B2**: perseguir a la fuente de luz amarilla móvil.

Solo uno de los robots conoce la tarea correcta a realizar en cada momento!



Muchas gracias

MUCHAS GRACIAS!