Diseño e Implementación de un Sistema de Percepción Inteligente

para Robots Futbolistas Móviles con ESP32

Cristian León · Santiago Rodríguez · Juan Pablo Chavarro

Universidad Sergio Arboleda — Escuela de Ciencias Exactas e Ingeniería
Ciencias de la Computación e Inteligencia Artificial — Toma de Decisiones

Docente: Ricardo Fonseca
Bogotá, Colombia
7 de mayo de 2026

Documento académico elaborado en LaTeX con formato mejorado, ecuaciones bien escritas, tablas estilizadas y presentación profesional.

1 Planteamiento del problema

En un entorno de fútbol robótico autónomo tipo RoboCup Small Size League (SSL), el robot debe tomar decisiones en tiempo real a partir de señales físicas captadas por sensores. El campo de juego mide 180×90 cm y los ciclos de decisión deben ejecutarse en menos de 100 ms. El problema no consiste únicamente en medir distancias o detectar colores, sino en interpretarlas de manera conjunta para determinar qué objeto tiene enfrente y cómo actuar.

Sea el estado del sistema en el instante t:

$$\mathbf{s}_t = (x_r, y_r, x_b, y_b, x_o, y_o) \tag{1}$$

donde (x_r, y_r) es la posición del robot, (x_b, y_b) la del balón y (x_o, y_o) la del oponente. El robot selecciona una acción

$$a_t \in \{\text{avanzar}, \text{girar}, \text{disparar}, \text{pasar}\}$$

según una política $\pi : S \to A$. El objetivo es encontrar la trayectoria óptima:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in S} J(\mathbf{x}), \qquad J(\mathbf{x}) = w_1 d_{\text{ball}} + w_2 c + w_3 t \tag{2}$$

donde d_{ball} es la distancia al balón, c el número de colisiones, t el tiempo de ejecución y w_1, w_2, w_3 son coeficientes de ponderación.

El desafío adicional reside en que todo ocurre bajo condiciones de incertidumbre: ruido sensorial, pérdida de paquetes en comunicación inalámbrica (hasta 8 %) y posiciones de oponentes en cambio constante. Por estas razones, no fue suficiente utilizar reglas rígidas simples; se optó por un enfoque híbrido de aprendizaje supervisado combinado con algoritmos bioinspirados.

2 Variables medidas por el sistema

El sistema utiliza tres sensores ultrasónicos HC-SR04 y un sensor de color TCS34725. En cada ciclo $\Delta t \approx 30$ ms se genera el vector de observación:

$$\mathbf{o}_t = (\text{dist\_CA}, \text{dist\_CB}, \text{dist\_IZQ}, \text{dist\_DER}, \text{color\_R}, \text{color\_G}, \text{color\_B}) \tag{3}$$

2.1 Distancias ultrasónicas

Cada sensor HC-SR04 emite un pulso ultrasónico y mide el tiempo de retorno. La distancia se calcula mediante:

$$d = \frac{t \cdot v}{2} \tag{4}$$

donde $v = 0.0343$ cm/μs es la velocidad del sonido en el aire, implementada en código como:

$$d = \frac{\text{dur} \times 0.0343}{2.0} \tag{5}$$

Se obtienen cuatro variables de distancia: `dist_CA` (frontal superior), `dist_CB` (frontal inferior), `dist_IZQ` (izquierda) y `dist_DER` (derecha). Las distribuciones presentan sesgo a la derecha con media esperada entre 40–80 cm y desviación estándar de ±30 cm.

2.2 Variación temporal de distancia

Para distinguir objetos estáticos de objetos en movimiento se calcula la variación absoluta entre lecturas consecutivas:

$$\Delta d = |d_t - d_{t-1}| \tag{6}$$

Una pared genera variaciones bajas, un oponente en movimiento variaciones mayores, y un balón disputado presenta cambios dinámicos. Se obtienen así `var_IZQ`, `var_CENT` y `var_DER`.

2.3 Variables de color

El sensor TCS34725 mide componentes crudas R, G, B y un canal de claridad total C. La normalización por canal claro reduce el error de clasificación en un 22 % frente a valores absolutos:

$$R_{\text{norm}} = \frac{R}{C}, \qquad G_{\text{norm}} = \frac{G}{C}, \qquad B_{\text{norm}} = \frac{B}{C} \tag{7}$$

Las componentes normalizadas satisfacen

$$R + G + B \approx 1$$

por construcción. El balón naranja se identifica mediante los umbrales:

$$R > 0.45, \qquad 0.20 < G < 0.45, \qquad B < 0.20 \tag{8}$$

3 Construcción del conjunto de datos

La etapa más importante fue la recolección experimental de datos en condiciones reales. Se diseñaron cuatro escenarios correspondientes a las clases del modelo:

- Clase 0: pared u obstáculo.
- Clase 1: oponente.
- Clase 2: balón libre.
- Clase 3: balón disputado.

Para cada clase se ubicó el robot frente al objeto o situación correspondiente y se registraron las lecturas de sensores en serie. Cada fila del dataset contiene:

$$X = [d_{\text{izq}}, d_{\text{cent}}, d_{\text{der}}, \Delta d_{\text{izq}}, \Delta d_{\text{cent}}, \Delta d_{\text{der}}, R, G, B], \qquad y \in \{0, 1, 2, 3\}$$

3.1 Descripción minuciosa de variables

Variable	Tipo	Rango	Unidad	Descripción
dist_CA	Continua	[0, 400]	cm	Distancia frontal superior (HC-SR04 centro arriba)
dist_CB	Continua	[0, 400]	cm	Distancia frontal inferior (HC-SR04 centro abajo)
dist_IZQ	Continua	[0, 400]	cm	Distancia lateral izquierda
dist_DER	Continua	[0, 400]	cm	Distancia lateral derecha
color_R	Continua	[0, 1]	—	Componente roja normalizada: R/C
color_G	Continua	[0, 1]	—	Componente verde normalizada: G/C
color_B	Continua	[0, 1]	—	Componente azul normalizada: B/C
pso_vel	Continua	[0.2, 1.0]	—	Velocidad de búsqueda PSO en el ciclo
gen_g1	Continua	[0.1, 1.0]	—	Gen GA: peso distancia al objetivo

gen_g2	Continua	[0.1, 1.0]	—	Gen GA: peso detección de color
gen_g3	Continua	[0.1, 1.0]	—	Gen GA: agresividad del empuje
fitness	Continua	$(-\infty, +\infty)$	—	Aptitud acumulada del GA en el ensayo
deteccion_ok	Binaria	$\{0, 1\}$	—	1 = balón naranja detectado correctamente
estado_final	Categórica	$\{0, 1, 2, 3, 4\}$	—	0 =BUSCAR, 1 =ACERCARSE, 2 =VERIFICAR, 3 =EMPUJAR, 4 =EVITAR

3.2 Posibles problemas con los datos y soluciones

- **Ruido sensorial:** los HC-SR04 presentan lecturas erróneas (= 0 cm) cuando el ángulo de incidencia supera 15°. Se filtran mediante mediana móvil de ventana 3.
- **Desbalance de clases:** la variable `deteccion_ok` se espera desbalanceada (proporción positiva ≈ 30–40 %). Se aplica SMOTE o ponderación de clases.
- **Latencia Bluetooth:** introduce jitter de 10–50 ms, modelado con distribución de Poisson.
- **Valores atípicos:** colisiones múltiples o pérdida de señal generan outliers. Se detectan con IQR y z-score.
- **Correlación entre genes:** g_1, g_2, g_3 evolucionan juntos. Se aplica PCA reteniendo el 95 % de varianza.

4 Preprocesamiento de datos

Antes de entrenar la red neuronal fue necesario normalizar las entradas, ya que las magnitudes tienen escalas muy diferentes. Si se entrenara sin normalización, las variables de mayor escala dominarían numéricamente el aprendizaje. La cadena de preprocesamiento aplicada fue:

- Filtrado de lecturas ultrasónicas nulas (dist = 0) mediante mediana móvil de ventana 3.
- Normalización min-max de variables de distancia al rango $[0, 1]$.

- Verificación de la restricción $R+G+B \approx 1$ en datos de color; eliminación de registros con $|R+G+B-1| > 0.05$.
- Codificación one-hot de `estado_final`.
- Reducción de dimensionalidad PCA sobre genes y velocidad PSO (retención del 95 % de varianza).

4.1 Fórmula de normalización min-max

$$x_{\text{norm}} = \frac{x - x_{\text{mín}}}{x_{\text{máx}} - x_{\text{mín}}} \tag{9}$$

En el código, los arreglos `X_MIN` y `X_MAX` no son números elegidos arbitrariamente: representan los valores mínimos y máximos observados en el análisis estadístico del dataset. Son el resultado directo del análisis experimental y permiten al microcontrolador reproducir la misma transformación matemática usada en el entrenamiento.

5 Modelo matemático implementado: Red Neuronal Perceptrón Multicapa

El sistema final no usa una regresión lineal simple, sino una red neuronal perceptrón multicapa (MLP) con arquitectura $9 \to 12 \to 4$. Su selección se justifica porque puede aprender fronteras de decisión no lineales, combinando simultáneamente todas las entradas del sensor.

5.1 Capa de entrada

El vector de entrada normalizado es:

$$\mathbf{x} = [d_{\text{izq}}, d_{\text{cent}}, d_{\text{der}}, \Delta d_{\text{izq}}, \Delta d_{\text{cent}}, \Delta d_{\text{der}}, R, G, B]$$

5.2 Primera transformación lineal (capa oculta)

Cada una de las 12 neuronas ocultas calcula una combinación lineal de las entradas:

$$z_j = \sum_{i=1}^{9} x_i W_{ij}^{(1)} + b_j^{(1)} \tag{10}$$

5.3 Función de activación sigmoide

$$h_j = \sigma(z_j) = \frac{1}{1 + e^{-z_j}} \tag{11}$$

5.4 Segunda transformación lineal (capa de salida)

$$o_k = \sum_{j=1}^{12} h_j W_{jk}^{(2)} + b_k^{(2)} \tag{12}$$

5.5 Conversión a probabilidades: Softmax

$$P(y = k \mid \mathbf{x}) = \frac{e^{o_k}}{\sum_{m=1}^{4} e^{o_m}} \tag{13}$$

5.6 Función de pérdida y entrenamiento

Durante el entrenamiento se minimizó la entropía cruzada:

$$L = -\sum_{k=1}^{4} y_k \log(\hat{y}_k) \tag{14}$$

donde y_k es la etiqueta real codificada y $\hat{y}_k$ la probabilidad predicha.

6 Significado de los números del código

- `X_MIN` y `X_MAX`: parámetros de normalización extraídos del análisis estadístico del dataset.
- `W1`, `BIAS1`, `W2`, `BIAS2`: parámetros aprendidos por la red neuronal durante el entrenamiento.
- Umbrales de color ($R > 0.45,\ 0.20 < G < 0.45,\ B < 0.20$): obtenidos del análisis experimental.
- Parámetros PSO ($w = 0.7,\ c_1 = 1.5,\ c_2 = 1.5$): valores estándar de convergencia.
- Genes iniciales GA ($g_1 = 0.6,\ g_2 = 0.8,\ g_3 = 0.5$): punto de partida experimental del cromosoma.

7 Algoritmos Bioinspirados: GA y PSO

Los métodos tradicionales como A* o Dijkstra funcionan en entornos estáticos, pero fallan ante múltiples agentes en movimiento e incertidumbre sensorial. Los algoritmos bioinspirados son metaheurísticas que convergen hacia soluciones de alta calidad con eficiencia computacional adecuada para sistemas en tiempo real.

7.1 Optimización por Enjambre de Partículas (PSO)

$$v_{t+1} = w\, v_t + c_1 r_1 (pBest - v_t) + c_2 r_2 (gBest - v_t) \tag{15}$$

$$x_{t+1} = x_t + v_{t+1} \tag{16}$$

Cuadro 2. Variables asociadas al PSO

Variable	Algoritmo	Rol	Descripción
dist_CA, dist_CB	PSO + Máquina de estados	Entrada	Detectan objeto cercano; activan transición BUSCAR → ACERCARSE
dist_IZQ, dist_DER	Máquina de estados	Entrada	Determinan dirección de evasión en estado EVITAR
pso_vel	PSO	Variable interna	Velocidad de exploración actualizada cada ciclo

7.2 Algoritmos Genéticos (GA)

$$G = (g_1, g_2, g_3) = (0.6,\ 0.8,\ 0.5) \tag{17}$$

$$t_{\text{emp}} = g_3 \cdot 600 + 200 \text{ ms} \tag{18}$$

$$g_i' = \text{clip}(g_i + U(-0.1,\ 0.1),\ 0.1,\ 1.0) \tag{19}$$

$$\text{fitness}_t = \text{fitness}_{t-1} + g_1 \cdot 5.0 \cdot [\text{empuje}] + g_2 \cdot 10.0 \cdot [\text{naranja}] - 2.0 \cdot [\text{falso positivo}] \tag{20}$$

$$F(C) = \begin{cases} \eta \cdot \dfrac{n_{\text{oponentes}}}{t_{\text{gol}}}, & \text{si se anota gol,} \\ \lambda \cdot \dfrac{d_{\text{inicial}} - d_{\text{final}}}{t_{\text{ejecución}} + \varepsilon}, & \text{si no se anota gol,} \\ 0, & \text{si se pierde el balón.} \end{cases} \tag{21}$$

Cuadro 3. Variables asociadas al GA

Variable	Algoritmc	Rol	Descripción
gen_g1	GA	Parámetro	Pondera la importancia de la distancia en el fitness
gen_g2	GA	Parámetro	Pondera la detección de color; se refuerza con éxito
gen_g3	GA	Parámetro	Controla la duración del empuje
fitness	GA	Salida	Aptitud acumulada; guía la mutación de genes
deteccion_ok	GA (g_2)	Salida	Etiqueta de éxito; retroalimenta el refuerzo de g_2

7.3 Arquitectura Híbrida GA + PSO

- **GA (capa estratégica, lenta $\sim$ 1 Hz):** decide qué hacer y ajusta parámetros globales.
- **PSO (capa táctica, rápida 10–20 Hz):** decide cómo hacerlo y optimiza la trayectoria local.

8 Arquitectura de Control: Máquina de Estados Finitos

El comportamiento autónomo se implementa mediante una máquina de estados con cinco estados principales:

- **BUSCAR:** exploración del entorno mediante PSO.
- **ACERCARSE:** desplazamiento hacia un objeto detectado.
- **VERIFICAR_COLOR:** confirmación de color naranja con el TCS34725.
- **EMPUJAR:** interacción física con el balón.

- **EVITAR:** maniobras de evasión ante obstáculos.

9 Paso del análisis al código embebido

El flujo de inferencia embebido fue:

1. Leer sensores y calcular distancias.
2. Calcular variaciones temporales.
3. Leer y normalizar valores R, G, B.
4. Normalizar las 9 entradas con min-max.
5. Aplicar capa oculta.
6. Aplicar sigmoide.
7. Aplicar capa de salida.
8. Aplicar softmax.
9. Elegir la clase con mayor probabilidad.

Por cada ciclo de 30 ms, el ESP32 ejecuta:

$$9 \times 12 + 12 \times 4 = 108 + 48 = 156$$

multiplicaciones, además del mismo número de sumas y las operaciones de activación.

10 Interpretación funcional del perceptrón en cada robot

10.1 En el robot jugador

El perceptrón decide si el robot ve una pared, un oponente, un balón libre o un balón disputado. A partir de eso, la máquina de estados activa BUSCAR, ACERCARSE, EMPUJAR o EVITAR.

10.2 En el robot portero

El mismo perceptrón MLP clasifica la situación perceptual, pero la interpretación táctica es defensiva. El portero utiliza perceptrones adicionales con pesos diseñados a partir del análisis del dataset.

11 Por qué no se usó una regla fija simple

Se pudo haber intentado una lógica del tipo "si R es alto y G es medio, entonces balón". Sin embargo, ese enfoque resulta débil porque las condiciones reales del campo cambian continuamente. La ventaja del perceptrón multicapa es que combina simultáneamente todas las entradas y aprende relaciones no evidentes.

12 Resultados observados

- Detección correcta del color naranja en condiciones de iluminación controlada.
- Mejora en la eficiencia de búsqueda mediante PSO.
- Reducción significativa de colisiones con el sistema de evasión.
- Ajuste progresivo de parámetros gracias al GA.

13 Conclusión

El trabajo consistió en diseñar un sistema de percepción inteligente para un robot móvil, basado en adquisición experimental de datos, preprocesamiento estadístico, clasificación supervisada mediante una red neuronal MLP y despliegue del modelo en un sistema embebido ESP32. El sistema fue complementado con una arquitectura híbrida GA + PSO para la toma de decisiones adaptativa en tiempo real.

Matemáticamente, el proyecto se fundamentó en el cálculo de distancia ultrasónica, variaciones temporales, normalización min-max, combinaciones lineales ponderadas, activación sigmoidal, softmax, inferencia de clase máxima, actualización PSO y función de fitness acumulada.

14 Referencias

[1] Fujita, M., Nakashima, Y., & Shibata, T. (2022). A self-localization method using a genetic algorithm for soccer playing humanoid robots. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 26(1), 32–45.

[2] Horsevad, N., Kwa, H. L., & Bouffanais, R. (2022). Beyond bioinspired robotics: how multi-robot systems can support research on collective animal behavior. *Frontiers in Robotics and AI*, 9.

[3] Tanner, H. G., Jadbabaie, A., & Pappas, G. J. (2007). Flocking in fixed and switching networks. *IEEE Transactions on Automatic Control*, 52(5), 863–868.