Inha-United 2026 Software Description Paper

Jaeyuk Seung[1], Eunbyeol Ko[1], Eungkyu Kim[1], Mingyu Kim[1], Suhyeon Shin[1], Jihun An[1]
Younggun Cho[1,2], Inwook Shim[1,3], Junwoo Jang[1,4], and Woojin Ahn[1,5]

[1] Inha University, Incheon, Korea,
[2] SPARO Lab., [3] RCV Lab., [4] Artemis Lab., [5] RILS Lab.
Corresponding Email to:
yg.cho@inha.ac.kr, iwshim@inha.ac.kr,
junwoo@inha.ac.kr, wjahn@inha.ac.kr
https://inha-united-soccer.github.io/Soccer2026/

January 31, 2026

Abstract. Team Inha-United is a research-driven group formed by four robotics laboratories at Inha University. Each laboratory focuses on core robotic technologies including perception, sensing, intelligence, decision-making, and control, and the team integrates these elements to develop multi-robot control systems for humanoid soccer. Through this platform, we address the challenge of achieving stable and cooperative autonomy under dynamic, adversarial, and partially observable environments.
Our approach emphasizes accessible system design, practical deployability, and robust integration of robotic subsystems. By validating these design choices in RoboCup competitions, we demonstrate how advanced core robotic technologies can be translated into reproducible, system-level solutions that support cumulative progress within the RoboCup and broader robotics communities.

1 Introduction

Our team, **Inha-United** at Inha University, is primarily composed of undergraduate students and is guided by principal investigators from multiple robotics laboratories. The laboratories collectively contribute expertise across core robotic technologies, including vision, sensing, intelligence, and control, forming the foundation of a unified, system-level humanoid robotic platform.

As summarized in Table 1, the participating principal investigators have a strong track record of success in international robotics competitions, which drives the team's technical direction, system design choices, and research objectives.

This Software Description Paper details the outcomes of our development efforts by presenting the overall system architecture and operational framework of the Inha-United humanoid system.

Table 1. Results of competitions

Competitions	Results
DARPA Robotics Challenge Final'15	1^{st} Place
ICRA'24, Workshop on Construction Robots	Best Research Award
Virtual RobotX Challenge'19	1^{st} Place

2 Our Contributions to RoboCup

The primary scientific contribution of our team lies in the design of a high-level, strategy-centered decision-making framework for humanoid robot teams. While individual capabilities such as perception, localization, and motion control are continuously improved, we focus on how team-level strategies are systematically designed and executed under the physical and computational constraints of humanoid platforms. In line with the league's vision, humanoid robot soccer is used as a representative setting for studying adaptive behavior in incomplete and highly dynamic environments.

To enable effective team performance under such conditions, our framework is built on modular skill representations and their composition into higher-level tactical behaviors. This structure allows strategies to be reconfigured and switched at runtime as situations evolve, while maintaining a consistent decision-making interface. As a result, role assignments, coordination policies, and tactical decision rules can be systematically instantiated and evaluated, with humanoid robot soccer serving as a testbed for rapid strategy iteration.

Beyond a single competition entry, our work establishes an extensible and practical decision-making architecture for multi-robot teams. By sharing this framework with the RoboCup community, we aim to lower the barrier to developing team-level intelligence and to promote reproducible research across diverse robotic applications.

In summary, our contributions to RoboCup are as follows:

1. A modular skill integration framework that supports independent development and evaluation of perception, localization, and motion algorithms.
2. A unified strategy construction and management framework enabling flexible composition, selection, and adaptation of team-level behaviors.
3. An accessible team-level architecture that simplifies the implementation of complex multi-robot strategies beyond humanoid soccer.

3 Overview

3.1 Experimental and Development Environment

To support full-scale operation in the RoboCup Humanoid League, we have constructed dedicated experimental environments that closely reflect real competition conditions. These efforts include the availability and operation of three

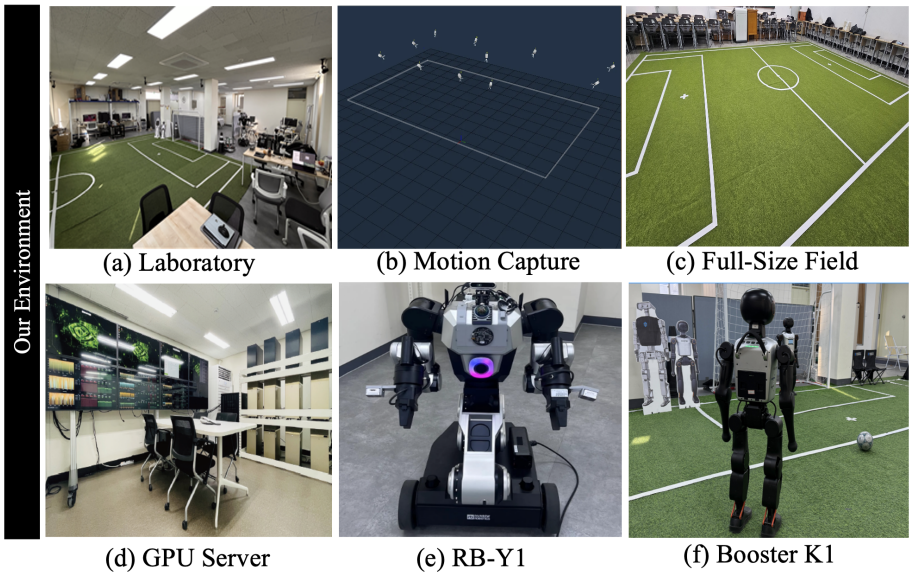


Fig. 1. Experimental environment and robot platform.

humanoid robots, the installation of both half-court and full-court soccer fields, and the integration of a motion capture system for ground-truth localization. In addition, high-performance computing servers equipped with NVIDIA A6000 and RTX 5090 GPUs are utilized for large-scale training, simulation, and model optimization. Figure 1 provides an overview of the experimental environments.

3.2 Overall System Architecture

Fig. 2 summarizes the system architecture used to handle the core behaviors required in humanoid soccer scenarios. At run time, the system receives multimodal inputs through communication interfaces (e.g., vision, joint states and referee state information). Based on these inputs, the system infers the current situation and translates the selected behavior into a set of functional requirements. These requirements are then processed sequentially by the vision (Section 4.1) and localization (Section 4.2) modules, followed by Behavior Tree–based decision making (Section 4.3), control (Section 4.4), and communication (Section 4.5) stacks. Overall functionality is implemented flexibly through the composition of reusable modules.

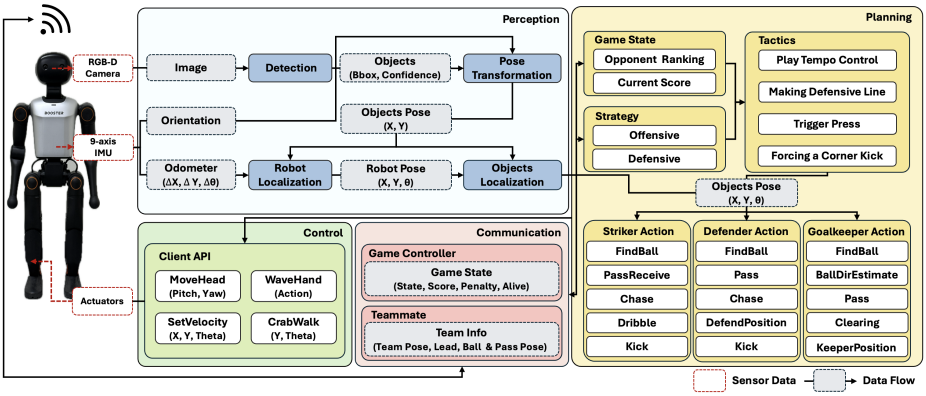


Fig. 2. Software architecture of the overall system.

4 Approach

4.1 Vision

Object Detection and Model Optimization We train a YOLOv8 [5] object detection model using the publicly available Torso21 Dataset [1], enabling real-time detection of field markers (L, T, X), goalposts, balls, and robots. As shown in Table 2, although the Torso21 dataset alone achieves high overall object detection performance, the ball class suffers from limited data diversity, which leads to suboptimal detection performance in real-world match environments. To address this limitation, we augment the training data with publicly available datasets from Roboflow [8] as well as privately collected data, thereby improving ball detection performance. As a result, Table 3 demonstrates that the expanded dataset, which includes balls of diverse appearances and conditions, yields a significant improvement in ball detection inference performance. To deploy the improved detection model in the actual robotic system, the trained network is converted into an NVIDIA TensorRT–based inference engine [7]. This optimization minimizes inference latency on embedded GPU platforms such as the Jetson Orin NX, achieving an average inference latency of 33 ms and enabling real-time operation.

For robot instances, additional post-processing is applied to estimate team color attributes. Specifically, color information within each detected bounding box is analyzed in the HSV color space, and uniform colors are classified based on their color distribution. The resulting color attributes are then utilized by the higher-level decision-making module.

Pose Transformation Subsequently, all detected objects are transformed into three-dimensional spatial information. The spatial information in the camera coordinate system is estimated by selecting the center pixel of the detected

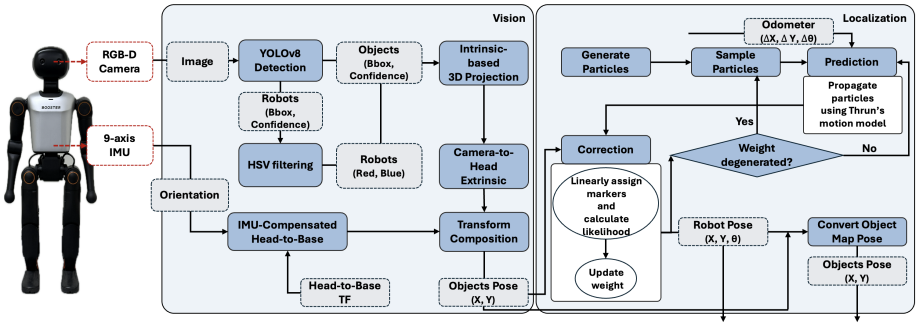


Fig. 3. Software architecture of the vision and localization modules.

Table 2. Detection performance comparison between the baseline trained on the Torso21 dataset and the proposed method with additional data augmentation.

	Precision	Recall	mAP_{50}	$mAP_{50\text{-}95}$
Baseline (All)	0.953	0.916	0.964	0.687
Ours (All)	0.943	0.940	0.970	0.702

Table 3. Ball detection performance before and after data augmentation.

	Precision	Recall	mAP_{50}	$mAP_{50\text{-}95}$
Baseline (Ball)	0.767	0.591	0.631	0.334
Ours (Ball)	0.838	0.693	0.726	0.399

object's bounding box and back-projecting it into a 3D ray using the camera's intrinsic parameters.

The estimated positions are then transformed into the robot base coordinate frame through a head-to-base transformation and represented as Cartesian coordinates. During walking, ground impacts and rapid posture changes are major factors that degrade the reliability of kinematics-based head-to-base coordinate transformations. To address these issues, roll and pitch estimates obtained from the onboard IMU are incorporated into the transformation matrix in real-time. This approach compensates for heading errors induced by body tilt, thereby ensuring the reliability of kinematic-based position estimation. As a result, the mean position error was reduced by 31.9 % compared to the K1 baseline [3], significantly enhancing the robustness of 3D position estimation.

4.2 Localization

Following the vision module, the localization system fuses visual observations with proprioceptive odometry in order to estimate the robot state on the field. Given the inherent uncertainty and ambiguity of humanoid perception, the

objective is not to recover a single pose instantaneously, but to maintain and gradually refine a probabilistic belief over the robot pose.

Problem Definition Humanoid soccer localization is fundamentally ill-posed. Compared to wheeled platforms, humanoid robots experience significant odometry drift caused by foot slippage, impacts, and intermittent ground contact. At the same time, the visual observations available on a RoboCup-style soccer field are highly ambiguous due to strong geometric symmetry and sparse landmarks. As a result, multiple robot poses may explain the same set of visual measurements, particularly during global localization or after tracking failure.

The localization system must therefore maintain multiple pose hypotheses, avoid premature convergence under symmetry, and remain robust to unreliable visual measurements.

Estimation Framework We adopt a particle-filter-based state estimation framework, where the robot pose is represented as a planar state $\mathbf{x} = (x, y, \theta)$. The belief is approximated by a set of weighted particles and updated through alternating prediction and correction steps. Odometry updates are received at approximately 400 Hz, while vision measurements are available at 8 Hz from rectified camera images. The particle filter operates at approximately 180 Hz on a Jetson Orin Nano platform.

Prediction Model During prediction, particles are propagated using an odometry-driven motion model. Incremental odometry is decomposed into a rotation–translation–rotation sequence, and motion noise is injected proportionally to the magnitude of translational and rotational motion. This model captures the stochastic nature of humanoid locomotion while allowing uncertainty to grow naturally during aggressive walking or rapid turns.

Measurement Representation The vision module provides detections of semantic field markers, including line intersections (L, T, X) and goal posts. Each detected landmark is expressed in the robot frame and transformed into the field frame for each particle hypothesis. Due to occlusions and partial visibility, the number and configuration of detected landmarks vary significantly over time.

Landmark Association To robustly associate visual observations with the known field map, a global one-to-one assignment formulation is employed. For each particle, a cost matrix is constructed between observed markers and map landmarks, constrained by marker type. The optimal assignment is solved using the Hungarian algorithm, ensuring consistent correspondence even under symmetric or cluttered observations.

This explicit assignment was found to be essential, as nearest-neighbor matching frequently produced unstable likelihoods when multiple symmetric landmarks were visible.

Likelihood Computation Given an assignment, the measurement likelihood is evaluated using a distance-anisotropic error model. For each matched landmark, the residual is decomposed into radial (normal) and tangential (perpendicular) components with respect to the landmark–robot direction. Uncertainty in both components is modeled as a function of the distance to the landmark.

This formulation is motivated by practical observations on a humanoid platform. Despite IMU-based stabilization, camera SE(3) estimates remain imperfect due to body sway, pitch variation during walking, and posture changes such as bending or leaning. These effects cause image-to-field projection to fluctuate in real time, often resulting in landmarks being projected systematically closer to the robot than their true field positions. Since the proposed likelihood model is highly informative, even a momentary projection error can introduce significant distortion into the posterior distribution if trusted excessively.

To mitigate this issue, the likelihood contribution is deliberately down-scaled, and the particle distribution is kept relatively compact. This design choice is particularly important for handling angular instability in image-based projection: small orientation errors in the camera frame can induce large lateral displacements on the field when particles are widely spread. By maintaining a tight hypothesis set, the filter reduces its sensitivity to such angular jitter, allowing particles to respond coherently to consistent orientation cues. At the same time, projection shortening at long range is addressed through distance-adaptive uncertainty modeling, where the radial variance increases more aggressively than the tangential one. Together, these mechanisms allow dominant, geometrically consistent hypotheses to persist while preventing transient SE(3)-induced projection errors from destabilizing the posterior distribution.

Resampling Strategy Particle degeneracy is monitored using the effective sample size (ESS). Resampling is triggered only when ESS falls below a predefined threshold relative to the total particle count. This adaptive strategy preserves multi-modal hypotheses under symmetric ambiguity while allowing efficient convergence once observations become informative.

Initial Pose and Impoverishment Handling Since no unconditional random particle injection is performed during normal operation, correct handling of the initial pose is critical. An incorrect initialization can lead to particle impoverishment from which recovery is not possible using vision alone.

To address this limitation, recovery logic is partially handled at the behavior level. When the behavior tree detects a failure-related state, such as a fall-down event or a sustained loss of localization confidence, a re-initialization routine is

triggered. Rather than re-sampling particles uniformly over the entire field, the injection region is conditioned on the robot's estimated state, including game context and odometry history. For example, particles may be initialized near the expected entry region during start-up, or around an odometry-consistent pose after a fall.

Final Pose Estimation Pose extraction is formulated as a temporally consistent process. When no prior pose estimate exists, the particle with the highest weight is selected to establish an initial reference. Once tracking is established, particles are gated based on position and orientation consistency relative to the previous estimate. If sufficient consensus exists, the pose is computed as a weighted mean; otherwise, the system falls back to the maximum-weight hypothesis.

The final output is stabilized using exponential moving average smoothing, ensuring continuity for downstream control modules.

Evaluation Localization accuracy was evaluated using the *Absolute Pose Error (APE)* with respect to the translational component, computed after *SE(3) Umeyama alignment*. The evaluation was conducted over a trajectory length of 203.226 m and a duration of 455.805 s, comprising a total of 153,555 pose estimates.

Quantitative results are summarized in Table 4. The proposed system achieves a mean translational error of 0.172 m with an RMSE of 0.193 m, demonstrating stable localization performance despite strong field symmetry and noisy vision-based measurements. The spatial distribution of the translational error over the evaluated trajectory is illustrated in Fig. 4.

Table 4. APE with respect to translation (SE(3) Umeyama alignment)

Metric	**Value (m)**
Max	0.535280
Mean	0.171813
Median	0.152359
RMSE	0.192973

4.3 Decision Making

Behavior Tree–based Decision Framework Our decision-making system is designed with a decentralized architecture, rather than a centralized architecture. Each robot independently executes its own Behavior Tree [4], evaluating its internal state and surrounding environment at every simulation tick and selecting

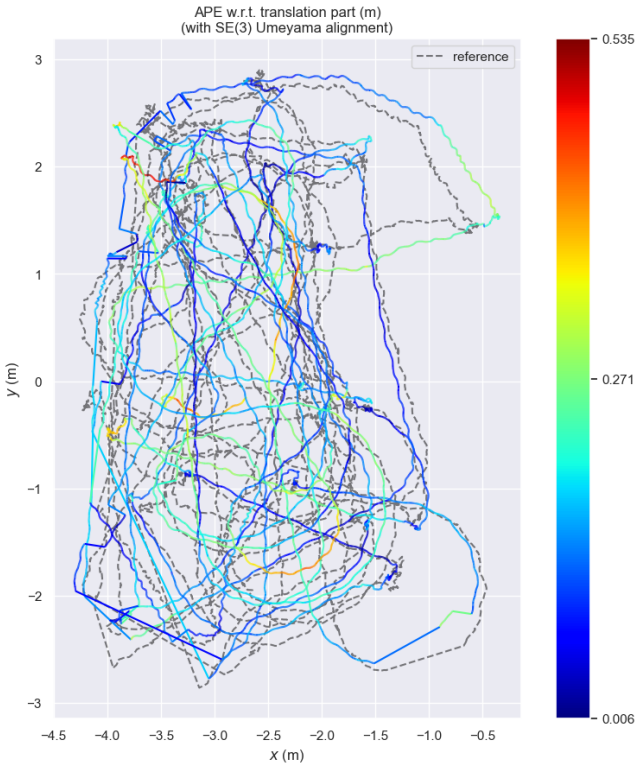


Fig. 4. Absolute Pose Error (APE) over the evaluated trajectory after SE(3) Umeyama alignment.

the most appropriate action node accordingly. Each robot is assigned one of three roles—Striker, Defender, or Goalkeeper—and each role is associated with distinct tactical objectives, including goal protection, ball contention, space occupation, and support for passing and shooting.

Dynamic Behavior Control via Context-Aware Parameter Composition To mitigate the limitations of rigid, discrete state transitions inherent in traditional Behavior Tree implementations, we introduce a Context-Aware Parameter Composition architecture. This system decouples strategic intent from mechanical execution through a hierarchical "Director-Tuner-Engine" pattern.

At the top level, a Tactic Selector (Director) analyzes macroscopic game states—such as score differentials and remaining time—to update context flags (e.g., isDesperate, isCounterOpportunity) on the blackboard. Based on these flags, a set of specialized Tactic Nodes (Tuners) executes concurrently as domain experts. Rather than triggering isolated behaviors, each node injects specific setpoints into a Composite Parameter Map. For example, in an isDesperate

scenario, the LineDefense tactic advances the defensive line deep into the opponent's half, while the Finishing tactic lowers the shooting threshold and the Pressing tactic maximizes locomotion intensity.

To ensure stability, the architecture employs a priority-based arbitration layer that resolves potential conflicts between overlapping tactics. This finalized map is then consumed by the Execution Layer (Engine), which remains agnostic to the high-level strategy. This decoupling allows the robot to exhibit fluid, context-sensitive transitions—such as shifting seamlessly from a cautious defensive posture to an all-out offensive press—by dynamically recomposing underlying parameters in real-time.

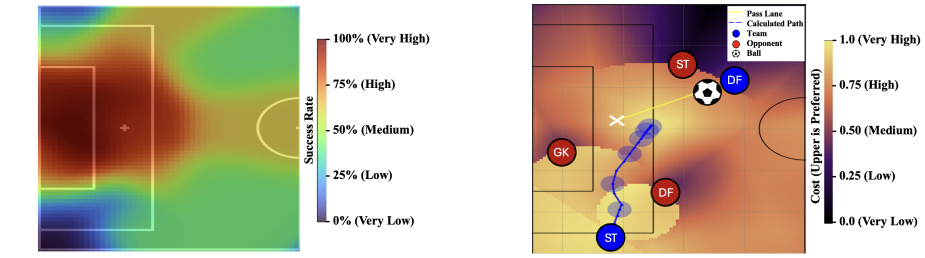
Grid-based Utility Maximization for Positioning Both the Defender and Striker employ a grid-based utility maximization approach to determine optimal positions that simultaneously avoid obstacles and maintain tactical advantages. The positioning algorithm aims to maximize pass-receive probability, secure valid shooting angles, and minimize collision risks with opponent robots. To this end, the field is discretized into a uniform grid, and for each candidate position P, a cost function defined in (1) is evaluated. The position that maximizes $Cost(P)$ is selected as the target position.

$$\begin{aligned} Cost(P) = \sum W_{pos} \cdot \Delta Pos + W_{def} \cdot OpenSpace_{def} \\ - \sum W_{occ} \cdot Occlusion(P) - Safety(P) \end{aligned} \tag{1}$$

where ΔPos encourages positions with high shooting success reliability, obtained through a quantitative analysis of kick success rates within a 3 m radius around the goal based on the experimental results shown in Fig. 5(a). $Occlusion(P)$ represents the degree to which a candidate position is obstructed by opponent robots. $OpenSpace_{def}(P)$ promotes positioning symmetric to the opponent defenders' distribution in order to secure spatial advantages for passing and offensive progression. $Safety(P)$ is a penalty term that suppresses position selection near high-risk regions, such as areas close to the goal or along the goal line.

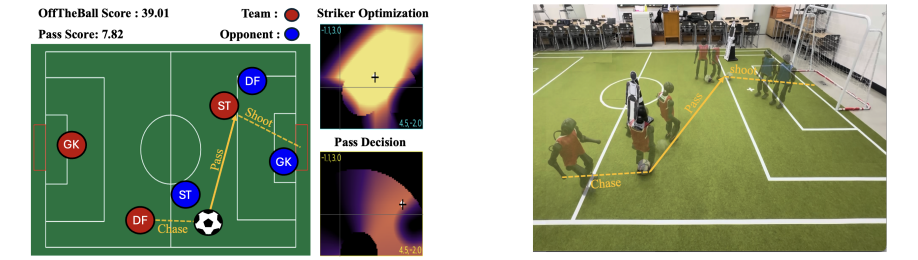
Fig. 5(b) visualizes the optimal movement trajectory derived from the proposed cost-function-based position evaluation. It illustrates how a path that maximizes Cost(P) is formed in a dynamic environment containing multiple robots and obstacles.

Fig. 6(a) presents the process of experimentally tuning the weight parameters of the cost function within a custom simulation that emulates a multi-robot environment, together with an example of the resulting pass trajectory. This analysis confirms that the positioning and passing strategies are adjusted to satisfy the intended tactical characteristics in simulation. Fig. 6(b) shows the application of the tuned parameters and strategies from Fig. 6(a) to the real robot system. The result demonstrates that the trajectories and tactical behaviors derived in simulation are consistently reproduced on physical robots.



(a) Position-dependent shooting success rate (b) Cost-Based optimal movement trajectory

Fig. 5. Trajectory from multi-term cost function, including shooting success rates.



(a) Simulation results of a defender-to-striker passing scenario (b) Reproduction of simulation results on the real robot

Fig. 6. Pass-and-shoot example with cost-driven pass targeting and receiving.

The actual motion toward the selected target position is executed by recomputing the movement direction and velocity at every control tick based on the robot's current position, the target position, and nearby obstacle information. Through this process, the robot continuously updates its target position in real time and incrementally forms a trajectory that maximizes Cost(P) even in dynamic environments.

Striker Decision Logic The Striker's decision-making process is structured into four high-level states: Ball Search, Cooperative Play, Tactical Approach, and Precision Attack.

The Ball Search state is activated when the ball position is uncertain from both the robot's own perception and team information. In this state, the robot performs active search behaviors to reacquire the ball.

During Cooperative Play, the Striker considers team-level information such as pass signals and attack leadership status. Grid-based utility maximization is used to compute positions that simultaneously maximize pass-receive likelihood and tactical advantage.

In the Tactical Approach state, the Striker selects between chasing and dribbling based on the distance between the ball and the center of the opponent's goal. Once the robot enters a viable shooting range, it transitions to the Precision

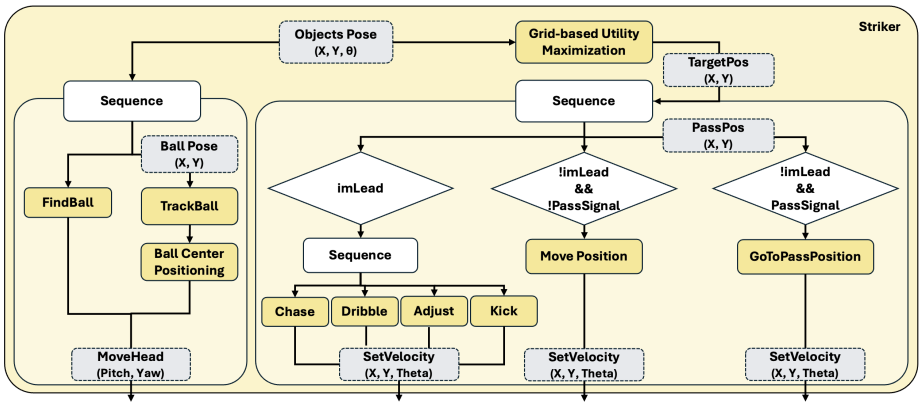


Fig. 7. Software architecture of the Striker behavior modules.

Attack state. Depending on the location-dependent risk level around the defensive danger zone, the system distinguishes between precise and quick shooting scenarios.

The Striker then executes adjustment and kick actions sequentially. Within the penalty area, alignment constraints are relaxed to prioritize responsiveness and enable rapid shots, whereas in more structured attacking situations, stricter alignment requirements are enforced to maximize accuracy. To prevent decision chattering caused by minor posture fluctuations, a Kick Lock mechanism is employed. Once shooting conditions are satisfied, the kick action is enforced for a fixed duration. The lock is released only when the distance to the ball exceeds a predefined threshold, ensuring robust handling of exceptional situations.

Defender Decision Logic The Defender's decision-making process is organized into four high-level states: Ball Search, Pass/Kick Decision, Clearing, and Off-the-ball Positioning. The Defender determines whether it currently holds the attacking initiative (imLead) based on ball observability and its relative spatial relationship to the ball.

When the Defender possesses the ball and is in the imLead state, it prioritizes advancing the attack through passing. To this end, the Defender jointly evaluates teammates' forward positioning and inter-robot distances to select the most tactically advantageous pass recipient between the Striker and the Goalkeeper. Grid-based utility maximization is then applied in the vicinity of the selected teammate to generate candidate pass target positions. If the maximum $Cost(P)$ exceeds a predefined pass feasibility threshold, the pass is executed. Otherwise, when no valid passing route is available, the Defender performs a forward kick to minimize defensive risk.

When the opposing team possesses the ball while the Defender is in the imLead state, the Defender immediately prioritizes defensive transition and ex-

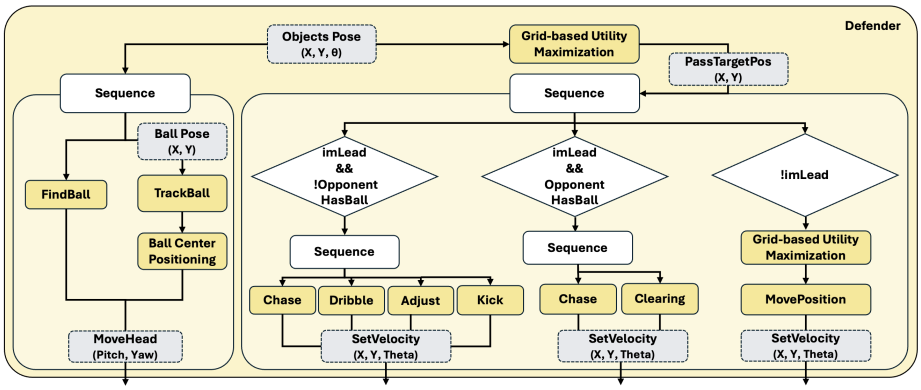


Fig. 8. Software architecture of the Defender behavior modules.

ecutes a clearing kick. In this case, the clearing direction is chosen away from the home goal, thereby disrupting the opponent's attacking progression and enhancing defensive stability.

When the Defender is not in the imLead state, the Off-the-ball Positioning state is activated. The Defender moves to positions that facilitate future pass connections while maintaining defensive coverage. Through this behavior, the Defender improves spatial occupation efficiency and preserves tactical continuity even when not directly involved in ball possession.

Goalkeeper Decision Logic The Goalkeeper's decision-making framework consists of three high-level states: Hold, Clearing, and Find.

In most situations, the Goalkeeper remains in the hold state, continuously adjusting its position to minimize the opponent's shooting angle. Ball trajectory prediction is performed using a Kalman filter with a uniformly accelerated linear motion model, and the optimal defensive position is computed based on the geometric relationship between the predicted ball position and the center of the goal.

When the ball enters a critical region within 2 m in front of the goal, the Goalkeeper transitions to the clearing state and kicks the ball toward the opposite direction of the goal. The computation of the kick direction is selectively enabled depending on the relative configuration between the robot and the ball.

If the ball position is lost, the Goalkeeper transitions to the find state and performs an active search behavior combining head rotation and body rotation, enabling full-area visual coverage.

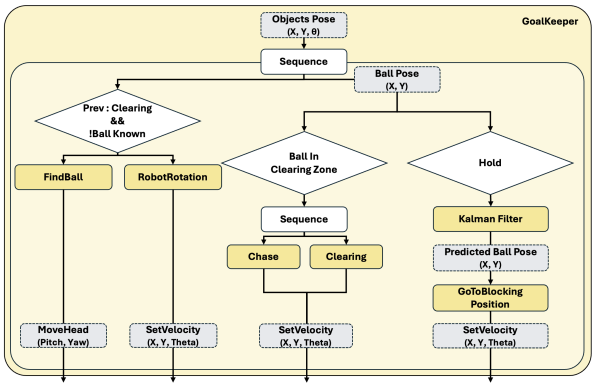


Fig. 9. Software architecture of the GoalKeeper behavior modules.

4.4 Control

The current system operates on top of the stability-verified baseline control modules provided by Booster Robotics. The control layer exposes low-level APIs such as SetVelocity, MoveHead, and GetUp, while the high-level decision-making module governs action execution through explicit rules that incorporate velocity constraints, distance conditions, and alignment requirements.

For example, the dribble, walk, and kick actions share the common SetVelocity interface but are configured with different locomotion speeds. Depending on the situation, these actions employ linear velocities of 0.2 m/s, 0.5 m/s, and 1.0 m/s, respectively. The chase action follows a trajectory generated by Grid-based Utility Maximization and applies P-control–based directional correction to continuously track the ball direction. During this process, the robot approaches the ball at a linear speed of 0.5 m/s. The adjust action corresponds to a fine alignment phase preceding a kick execution. In this stage, only the lateral (y-axis) velocity component and the rotational component (theta) of SetVelocity are modulated to align the robot with respect to the ball in both position and orientation.

To ensure robot stability, the CheckAndStandUp action is executed in parallel with all other behaviors. This module continuously monitors fall events and automatically triggers the default GetUp motion when a fall is detected.

4.5 Communication

Our team employs a UDP broadcast–based standard communication protocol defined by the RoboCup Humanoid Soccer League for inter-robot information sharing. Each robot periodically transmits its internal state and perception results, enabling team-wide situational awareness.

The received teammate information includes the robot's alive status (isAlive), assigned role (role), self-reported position (robotPose), and the timestamp of

the last received communication (timeLastCom). Based on this information, the teammate list is continuously updated. Robots from which no communication has been received for a predefined duration are regarded as inactive and removed from the list.

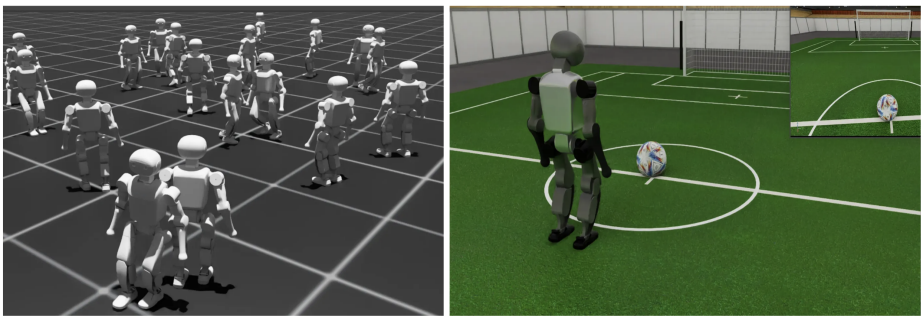
When a robot fails to directly observe the ball, it selects and utilizes the most reliable ball information received from teammates. This strategy enables stable ball position estimation even under partial perception loss or occlusion. In addition, the Defender can transmit pass signals and target coordinates to the Goalkeeper or Striker based on tactical considerations. Upon receiving such information, the recipient robot prioritizes the indicated location as its movement target, facilitating cooperative passing behaviors.

To ensure overall communication robustness, any received message whose memory size does not match the predefined format is discarded, preventing the propagation of corrupted or invalid data.

4.6 Simulation

As shown in Fig. 10, our team constructed an Isaac Sim [6] experimental environment based on the URDF model and simulation assets provided by Booster Robotics [2]. This simulation environment is used to analyze and validate tactical performance according to each robot role (Striker, Defender, and Goalkeeper) using ground-truth position data defined in the world coordinate frame.

In addition, building on the control codebase provided by Booster Robotics, we perform reinforcement learning for motion generation in the Isaac Lab environment, and plan to comprehensively validate the learned motions within the MuJoCo simulation environment.



(a) Isaac Lab for reinforcement learning (b) Isaac Sim for system evaluation

Fig. 10. Simulation environments used in this study

5 Future Plan

This work presented a modular humanoid robot soccer system developed together with a large-scale competition environment, providing a practical testbed for evaluating high-level intelligence through the integration of research-level perception, estimation, planning, and control modules. The proposed modularization and open-source design facilitate extensibility and reuse, enabling both researchers and educators to readily experiment with multi-robot decision-making and coordination.

Based on the current system, our team plans to pursue the following research directions and system enhancements.

5.1 Depth–based Perception Enhancement

In the current system, 3D object positions are estimated from detection results using IMU-based posture compensation for real-time perception. However, during walking and ground contact, residual uncertainties may remain due to subtle posture variations that cannot be fully addressed by kinematic compensation alone. To mitigate this issue, we plan to incorporate depth information to estimate the actual ground structure and directly integrate this information into the 3D position estimation process.

5.2 Behavior Tree Expansion and Tactical Refinement

We plan to extend the Behavior Tree–based decision framework by comparing and analyzing a structure that assigns the Goalkeeper a tactical coordination role with the existing fully decentralized decision-making architecture, and selecting the most suitable structure for team-level tactics based on this analysis. In addition, we will design and apply team-specific tactical Behavior Tree subtrees to handle various special match situations, such as penalty kicks and goal kicks, and analyze how these situation-dependent tactical structures affect team cooperation and overall match flow.

5.3 Reinforcement Learning–based Motion Expansion

Starting from the already trained locomotion policy, we plan to progressively extend reinforcement learning–based control policies to more complex motions such as kicking and passing. In particular, these motions will be designed to be optimized for the Behavior Tree–based decision framework, strengthening the coupling between high-level tactical decisions and low-level motion execution.

References

1. Bit-Bots Team: Torso-21 dataset (2021), https://github.com/bit-bots/TORSO_21_dataset, accessed: 2026-01-27
2. Booster Robotics: Booster k1 technical documentation (2024), https://booster.feishu.cn/wiki/E3q5wF5SnitXZgkY18Uc8odBnXb, accessed: 2026-01-27
3. Booster Robotics: Booster k1 robocup demo repository (2025), https://github.com/BoosterRobotics/robocup_demo, accessed: 2026-01-27
4. Faconti, D.: Behaviortree.cpp: A behavior tree library for c++ (2023), https://www.behaviortree.dev, accessed: 2026-01-27
5. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics yolov8, version 8.0.0 (2023), https://github.com/ultralytics/ultralytics, accessed: 2026-01-27
6. NVIDIA Corporation: Nvidia isaac sim: Extensible robotics simulation (2023), https://developer.nvidia.com/isaac/sim, accessed: 2026-01-27
7. NVIDIA Corporation: Nvidia tensorrt: Programmable inference accelerator (2024), https://developer.nvidia.com/tensorrt, accessed: 2026-01-27
8. Roboflow Inc.: Roboflow: Computer vision dataset management platform (2024), https://roboflow.com, accessed: 2026-01-27