Software Survey 2026

Team name

Inha-United

Which division(s) are you applying for? If your used software differs between divisions, please fill out the survey once per division.

Middle Size (height < 125 cm, weight < 25 kg)

Is your software fully or partially OpenSource? If so, where can it be found?

Our team's software is fully open source and is built on top of the open-sourced Booster Robotics RoboCup Demo software.
We also utilize publicly available open-source software and libraries, including YOLO, BehaviorTree.CPP, and NVIDIA TensorRT.
All software developed by our team is released as open source and is publicly available through our team's GitHub repository.

Open-source resources:

1. Booster Robotics RoboCup Demo (base framework): https://github.com/BoosterRobotics/robocup_demo/tree/sandbox/feat/k1_3v3_demo
2. Ultralytics YOLOv8 : https://github.com/ultralytics/ultralytics
3. TensorRT : https://developer.nvidia.com/tensorrt
4. BehaviorTree.CPP : https://www.behaviortree.dev/
5. Inha-United team software repository: https://github.com/Inha-united-soccer

Are you using any software developed by other teams? If so, list every component that you are reusing and the team that originally developed it.

Yes. We reuse the open-source RoboCup Demo software developed by Booster Robotics as the base framework of our system. Beyond the basic libraries and code provided by the purchased platform, we do not use any algorithms or code developed by other RoboCup teams.

Open-source resources:

Booster Robotics RoboCup Demo(base framework):
https://github.com/BoosterRobotics/robocup_demo/tree/sandbox/feat/k1_3v3_demo

Are you using any datasets in your research? If you are using your own datasets, are they public?

For training our vision models, we used publicly available datasets, including the Torso-21 dataset (approximately 10,000 real images and 24,000 simulated images) and a Roboflow dataset consisting of approximately 7,000 images. In total, including approximately 500 privately collected images, 41,467 images were used for training.

Open-source resources:

1. bit-bots Torso-21 Dataset : https://github.com/bit-bots/TORSO_21_dataset
2. Roboflow Public Dataset : https://universe.roboflow.com/search?q=robocup

Please list the scientific publications your team has made since the last application to RoboCup (or if not applicable in the last 2 years).

Although Inha-United is a newly formed team, its constituent laboratories have made sustained contributions to the advancement of robotics by publishing key research findings over the past two years.

Park, J., Lee, J., Choi, E., & Cho, Y. (2024, May). Salience-guided Ground Factor for Robust Localization of Delivery Robots in Complex Urban Environments. In 2024 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1701-1708). IEEE.

https://ieeexplore.ieee.org/abstract/document/10611696

Lee, M., Park, M., Lee, J., & Cho, Y. (2024, October). Dynamic Multi-Object Analysis Using Particles for Social Navigation. In 2024 24th International Conference on Control, Automation and Systems (ICCAS) (pp. 922-926). IEEE.

https://ieeexplore.ieee.org/abstract/document/10773378

Lee, G., Lee, J., Kim, J., Shin, Y., & Cho, Y. (2025, Dec). MSG-Loc: Multi-Label Likelihood-based Semantic Graph Matching for Object-Level Global Localization. In 2025 IEEE Robotics and Automation Letters (RA-L) (pp. 2066 - 2073). IEEE.

https://ieeexplore.ieee.org/document/11297765

Park, S. G., Kim, H. B., Lee, Y. J., Ahn, W. J., & Lim, M. T. (2025). TARG: Tree of Action-reward Generation With Large Language Model for Cabinet Opening Using Manipulator. International Journal of Control, Automation and Systems, 23(2), 449-458.

https://link.springer.com/article/10.1007/s12555-024-0528-6

Kathuria, T., Liu, K., Jang, J., Yang, X. J., & Ghaffari, M. (2025). Learning Implicit Social Navigation Behavior using Deep Inverse Reinforcement Learning. IEEE Robotics and Automation Letters.

https://ieeexplore.ieee.org/abstract/document/10947583

Bai, J., & Shim, I. (2025). SceneAware: Scene-Constrained Pedestrian Trajectory Prediction with LLM-Guided Walkability. arXiv preprint arXiv:2506.14144.

https://arxiv.org/abs/2506.14144

Park, S. G., Kim, H. B., Kim, Y. G., Ryu, S. W., Yoo, B. G., Chung, S., ... & Lim, M. T. (2025). Replay: Robot embodiment via intent-aware policy imitation by replicating human demonstrations from video. International Journal of Control, Automation and Systems, 23(12), 3599-3609.

(Accepted)

Are there any other contributions you would like to share with the RoboCup community?

None.

Which approach are you using to generate the robot walking motion?

The current system is operated based on the stability-verified baseline control modules provided by Booster Robotics.

Building on a reinforcement learning–based control codebase, we plan to progressively enhance and refine the walking motion.

Open-source resources:

Booster Robotics RL Demo : https://github.com/BoosterRobotics/booster_train

Which approach are you using to generate other motions of the robot (e.g. kicking, standing up)?

The current system is operated based on the stability-verified baseline control modules provided by Booster Robotics.

Higher-level motions are generated by composing fundamental locomotion primitives such as SetVelocity and MoveHead.

Using this approach, the robot aligns itself with the ball and realizes various ball manipulation behaviors, including dribbling, passing, and shooting, by appropriately adjusting the velocity command of SetVelocity.

In addition, by jointly controlling its motion path and body orientation, the robot can control the direction of the ball or perform defensive behaviors such as blocking incoming shots.

Do you have a kinematic or dynamic model of your robot? If so, how did you create it (e.g. measure physical robot, export from CAD model)?

We are using the Booster Robotics K1 model, and its URDF is publicly available.

Open-source resources:

Booster Robotics Public Document :

https://booster.feishu.cn/wiki/E3q5wF5SnitXZgkY18Uc8odBnXb

Booster Robotics K1 Assets : https://github.com/BoosterRobotics/booster_assets

What approaches are you using in your robot's visual perception?

We use the YOLOv8 detection model for visual perception.

The model is trained to detect field markers(L, T, X), ball, robots, and goalposts. To ensure a stable frame rate under real-time operation, the trained model is converted into an NVIDIA TensorRT–based inference engine, minimizing inference latency on embedded GPU platforms such as the Jetson Orin NX.

In addition, we are actively evaluating the performance impact of modifications to the model backbone and network architecture.

Are you planning with objects in Cartesian or image space? If you are using Cartesian space, how do you transform between the image space and cartesian space?

Yes, the system operates in Cartesian space.

The vision module selects the center pixel of the detected object’s bounding box and back-projects it into a 3D ray using the camera intrinsic parameters, enabling the estimation of the object’s position in the camera coordinate frame. The estimated position is then transformed from the head coordinate frame to the robot base coordinate frame and represented as a Cartesian object position relative to the robot.

To mitigate errors caused by ground impacts and posture variations during walking, roll and pitch estimates from the onboard IMU are incorporated into the transformation matrix in real-time. This compensates for heading errors induced by tilting, thereby ensuring the reliability of kinematic-based position estimation.

In future work, depth information will be incorporated to estimate ground geometry, reducing reliance on ground plane assumptions and enabling more stable 3D position estimation.

Do you have some form of active vision (i.e. moving the robots camera based on information known about the world)?

Yes. The system employs an active vision strategy by controlling the robot’s head motion.

During the search for the ball or field landmarks, a scanning pattern is applied in which the head moves from the lower-left to the upper-right of the visual field, either smoothly or rapidly depending on the situation, in order to efficiently cover the scene and increase the likelihood of object detection.

Once the ball is detected, the system switches to a tracking mode, where the head angles are continuously adjusted to keep the ball close to the center of the image, thereby ensuring stable ball observation and reliable visual information.

What approach are you using to localize your robot?

The system estimates the robot’s 2D position using a particle filter based on field marker observations obtained from the vision module.

The landmarks used include L, T, and X field markers as well as goalposts, and each landmark is provided as a 3D Cartesian coordinate relative to the robot.

During the initialization stage, competition rules specifically that the robot is placed facing outward from its own half at the start of the game are applied to restrict the initial pose space, thereby reducing pose ambiguity caused by field symmetry and accelerating convergence.

State prediction is performed using an odometry-based stochastic motion model, while measurement updates are computed by associating observed landmarks with map landmarks through a Linear Assignment Problem (LAP), evaluating distance-based likelihoods, and excluding low-confidence observations.

To prevent particle degeneracy, selective resampling based on the effective sample size (ESS) and limited random particle injection are applied. The final robot pose is estimated as the weighted mean of a high-weight particle cluster and stabilized over time using an exponential moving average (EMA).

Is your team performing team communication? Which communication protocol are you using?

Yes, our team performs inter-robot communication to share essential game state information such as robot status and high-level commands.
The communication is implemented in compliance with the RoboCup Humanoid Soccer League (HSL) standard communication protocol based on UDP broadcast.

What approach are you using for navigation? Are you avoiding obstacles?

Instead of moving in a straight line, the robot considers the surrounding environment and obstacles, and selects a tactically advantageous target position based on a cost map, in which the field is divided into a grid and each location is scored according to its tactical utility. This allows the robot to determine its movement goal while taking into account tactical factors such as pass success probability, shooting opportunities, and offensive progression, while the actual movement is executed using real-time local path planning.

At each control tick, the robot recalculates its movement direction based on its current position, the target position, and nearby obstacles and opponent robots detected by the vision module. Among adjacent movement directions, it selects the one with the relatively highest score, while areas with high collision risk are assigned lower scores in the cost map to naturally avoid them. By repeating this process, the robot progressively forms an optimal path and safely reaches the target position even in dynamic environments.

How is the behavior of your robots structured? (e.g. Behaviour Trees)

The behavior of our system is structured using a hierarchical Behavior Tree (BT) framework. At the highest level, behaviors are organized according to the robot’s role: Striker, Defender, and Goalkeeper. Each role has distinct tactical objectives and responsibilities, but all roles share a common set of initial behaviors and a unified decision-making flow.

In the initial stage of the Behavior Tree, all robots execute a set of fundamental behavior nodes designed to stabilize localization and environmental perception. These nodes include head motion control to support localization, ball search, and ball tracking. Through these behaviors, the robot establishes a reliable understanding of the current game situation and its own state. Once sufficient information is obtained, the robot transitions into the branch of the behavior tree corresponding to its assigned role.

Each role-specific behavior structure is composed of relatively large-scale tactical behavior nodes. The Striker focuses on offensive actions such as off-the-ball positioning, pass receiving, dribbling, and kicking, aiming to support attack buildup and scoring. The Defender prioritizes defensive stability and ball recovery through behaviors such as passing, clearing, and kicking. In contrast, the Goalkeeper performs role-specialized decision making for goal defense, using a limited set of behaviors including holding position, clearing the ball, and object search.

The Striker and Defender share the characteristic of operating across wide areas of the field and frequently interacting with opponent robots. Accordingly, both roles continuously update tactically advantageous positions and select movement targets by referencing the cost map used in the navigation system, allowing them to avoid obstacles while moving.

Within each high-level tactical behavior node, actions are further decomposed into fine-grained nodes such as chasing, adjusting alignment, and kicking. Transitions between these nodes occur naturally based on execution conditions, including the distance to the ball, alignment status, and opponent robot positions. These transitions are not governed by explicitly defined state rules; instead, they are automatically handled by the tick mechanism of the Behavior Tree and the execution result of each node (success, running, or failure), enabling flexible adaptation to dynamic changes in the environment.

Are you simulating your robot? If so, which simulator are you using and for what purpose do you use simulations?

Our team built an Isaac Sim experimental environment based on the USD model and simulation assets provided by Booster Robotics.

This environment is used to quantitatively evaluate localization performance using ground-truth data defined in the world coordinate frame, and to analyze and validate tactical performance according to each robot role.

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.

What operating system is running on your robot and which middleware are you using (for example Ubuntu 22.04 and ROS2 Galactic)?

Ubuntu 22.04 and ROS2 Humble

Is there anything else you would like to share that did not fit any previous question?

We designed the overall system with a modular architecture, allowing each low-level action and functional component to be easily replaced or extended in a plug-in manner. Based on these modular components, high-level strategies can be flexibly constructed using a Behavior Tree (BT), enabling efficient strategy modification and functional extensibility.

more information:
Inha-United team software repository: https://github.com/Inha-united-soccer