Software Survey 2026

Team name

Mountain

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

Small Size (height < 110 cm, weight < 15 kg)

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

https://github.com/HighTorque-Robotics/RoboCup_Workspace

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.

https://github.com/HighTorque-Robotics/RoboCup_Workspace

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

No

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

NO

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

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

We utilize Deep Reinforcement Learning (RL) with Adversarial Motion Priors (AMP) for gait generation. By training RL policies in NVIDIA Isaac Lab and incorporating an AMP discriminator to learn from high-quality reference motion data, we achieve walking gaits that are both physically stable on varied terrains and naturally biomimetic.

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

Kicking: Instead of treat kicking as a simple trajectory-following task, we incorporate human motion priors via the AMP framework. The RL policy is trained within the NVIDIA Isaac Lab environment to optimize the trade-off between striking power and the stability of the support leg. Through adversarial training with an AMP discriminator, the robot learns human-like whole-body coordination, allowing it to maintain dynamic balance even during high-velocity kicks.

Standing Up: Recovery motions are modeled as end-to-end RL tasks. By applying AMP constraints to the reward function, the robot learns smooth, efficient, and physically consistent stand-up behaviors. This approach avoids the rigid collisions and unnatural compensation movements typically found in traditional keyframe-based methods.

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)?

Yes, we have a comprehensive URDF model. It was exported directly from the industrial CAD models (SolidWorks), including precise data for link masses, Center of Mass (CoM), and inertia tensors, which were further calibrated against the physical hardware.

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

We employ deep learning for real-time object detection. Specifically, we use YOLOv8/v10 models optimized for embedded deployment to identify the ball, goals, field lines, and opponents, accelerated by TensorRT.

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?

We primarily plan in Cartesian space. Transformation is achieved using the camera's intrinsic matrix for undistortion and extrinsic parameters derived from the robot's current joint state (kinematic chain) to map pixels to world coordinates.

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

Yes. We implemented a 2-DOF head tracking strategy. The robot executes a scanning pattern when the ball is lost and performs real-time visual servoing to keep the ball centered in the frame once detected.

What approach are you using to localize your robot?

We use Monte Carlo Localization (MCL). Our system fuses visual features (e.g., field corners, goal posts) with IMU and odometry data to maintain a probabilistic pose estimate on the field map.

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

Yes, we adhere to the official RoboCup SPL Standard Message Protocol for team communication, supplemented by a lightweight custom UDP protocol for internal status broadcasting.

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

We use the Dynamic Window Approach (DWA) for local path planning. Opponents detected by the vision system are treated as dynamic obstacles for real-time avoidance.

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

Our behavior logic is structured using Behavior Trees (BT), developed via the Groot2 framework. This allows for modular and flexible switching between game states such as attacking, defending, and searching.

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

We use NVIDIA Isaac Sim/Lab primarily for: 1. Massively parallel RL policy training; 2. Motion style transfer via the AMP framework, converting reference motion data into executable joint commands; 3. Validating Sim-to-Real transfer algorithms to ensure robust performance on the physical hardware.

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

Our robots run on Ubuntu 22.04 LTS.

The middleware used is ROS2 Humble (LTS).

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