Software Survey 2026

Team name

Naova

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?

The software is partially open source.
A public release of last year's code is available at this link (https://github.com/Naova/NaovaRelease). However, that codebase is not reused in the current setup. Some high-level behaviour concepts are reused, as well as the line detection we developed last year.
The current year's software is currently in development and is not yet public.

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.

While we are not directly reusing another team's software, our project builds upon Booster Robotics' RoboCup demo framework (https://github.com/BoosterRobotics/robocup_demo), retaining its core infrastructure including the booster_interface, game controller module, and ROS2 Humble architecture. We are also reimplementing certain logic from our previous codebase—originally based on the 2021 B-Human release—without reusing complete components. This means we are drawing on established concepts, such as behavioural strategies and our approach to robot localization, which derives from the B-Human framework.

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

Yes, we use custom private datasets for both vision and motor skills training:

Vision: Custom image dataset for YOLOv11n object detection. The dataset includes labelled images of balls, goalposts, and field markers under various lighting and match conditions.

Training applies data augmentation and semi-supervised learning (teacher-student distillation).

Motor Skills: Custom motion trajectory datasets in CSV format, collected via teleoperation (joystick control) for specific skills: kicks and walks. These motions are used with the BeyondMimic motion tracking framework provided by BoosterRobotics, trained via RSL-RL PPO in Isaac Lab simulation with domain randomization for sim-to-real transfer.

Both these datasets will be kept private for the time being.

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

Our team has focused on developing technical solutions for this year, especially in the context of the platform transition. However, we submitted the article Line Detector for NAO Robot las year:
St-Pierre, O., & Bisson, M.-O. (2024). Détecteur de lignes pour robot NAO [Line detector for NAO robot]. Technical Report, [ÉCOLE DE TECHNOLOGIE SUPÉRIEURE]

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

In 2022, we contributed to the development of a walking controller for NAO:
Kali, Y., Saad, M., Boland, J. F., & Fallaque, C. (2022). Walking control using TDE-based backstepping SM of position-commanded NAO biped robot with matched and unmatched perturbations. Journal of Control, Automation and Electrical Systems, 33, 1633–1642.

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

Our walking motion is trained via reinforcement learning using BeyondMimic framework. The walking trajectories (from teleoperation) are tracked by a PPO policy trained in Isaac Lab. The trained policy is deployed via Booster Deploy framework.

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

Other than walking, we are only working on improving kick motions. Again, the kicks are

trained using BeyondMimic/PPO framework with task-specific reward design. The reference kick data is collected from teleoperation. Each type of kick is trained as a separate policy and deployed independently via Booster Deploy based on game state decisions. Other specific motions such as stand up are already automatically implemented by Booster in our code base.

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 use kinematic and dynamic models of the K1 robot provided by Booster Robotics:
Model sources:
URDF format: robots/K1/K1_22dof.urdf (Unified Robot Description Format)
XML format: robots/K1/K1_22dof.xml (MuJoCo simulation format)
source: https://github.com/BoosterRobotics/booster_assets

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

We rely on a hybrid approach: object detection (ball, robots, goal posts) is based on deep learning, while line detection uses a more classical computer vision approach implemented with OpenCV.
Object Detection : Our strategy is based on Transfer Learning, using the YOLOv11n architecture, pre-trained on the COCO dataset. This lightweight architecture offers an excellent trade-off between accuracy and inference speed. The model is fine-tuned for RoboCup specific classes (ball, robots, goal posts) and exported to the NVIDIA TensorRT format to fully exploit GPU acceleration on the Jetson Orin NX. At runtime, the vision module loads the TensorRT engine (.engine) from a YAML-specified path and publishes detections for downstream localization and behavior control
Field and Line Detection : Field lines and the center circle are detected using a vision pipeline developed by our club. The method combines adaptive thresholding, skeletonization of the painted lines (to extract the middle axis of the lines), and a probabilistic Hough transform to extract line segments under different lighting. Detected lines are further exploited as tangents to estimate the center circle position via clustering method.

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 implement Cartesian-space planning leveraging the K1's RGB-D stereo depth camera to extract depth values for objects of interest, which are then projected into the robot frame and subsequently into the global field frame via rigid transformation. To stabilize vision measurements during dynamic locomotion, we fuse a 9-axis IMU (accelerometer and gyroscope) with the depth data, decoupling the camera's ego-motion from the ball's motion to maintain an accurate Cartesian map even during sprinting and walking sway. However, stereo depth error increases quadratically with distance, preventing reliable full-field localization. To address this, we implement a hybrid fallback: for distant objects or depth dropouts, we switch to geometric ray-casting (ground plane projection) based on camera extrinsic calibration, stabilizing these raw coordinates using IMU data to compensate for walking sway. Both measurement streams are fused in a Kalman filter to produce stable position and velocity estimates in the global field frame, enabling predictive interception tasks and robust object localization across varying distances and lighting conditions.

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

Yes, our robot implements a form of active vision: it orients its camera according to its position, its objectives, and the current state of the game. To achieve this, it relies on several head-control skills that adapt the camera direction in real time.
In practice, it prioritizes tracking the ball when it is close or in motion, and it can also anticipate an action by alternating its gaze toward the shooting direction (only if no obstacle threatens ball possession). When no critical target is visible and no precise information is available, the robot performs a systematic scan of the field (following a predefined grid) to update the positions of important elements. It is also capable of orienting its camera toward the referee during appropriate phases of the game, while activating dedicated perception modules—which would be too computationally expensive to run continuously.

What approach are you using to localize your robot?

We use an approach derived from the B-Human framework, based on a particle filter (Monte Carlo Localization) combined with Unscented Kalman Filters (UKF) to estimate the robot’s position (x, y, theta) in the field frame in real time.

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

The communication protocol will use UDP broadcast-only sending a single compact message (≤128 bytes) each cycle that includes the robot's role, behavioural state from the behaviour tree, ball information, and pose estimate. All robots will simultaneously receive the same message via broadcast.

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

The current navigation uses a simple reactive approach where the robot calculates velocity commands directly from its current pose to a target pose without global path planning, while obstacle avoidance is handled by processing depth images into a 2D occupancy grid and using distance-to-obstacle checks in multiple directions to find safe movement angles when collisions are imminent.

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

The desired robot behaviour is structured as a hierarchical behaviour tree with subtrees organized on several layers such as:

1. Safety check: (battery, camera, IMU) at the root
2. Critical states (fallen recovery, calibration, penalties) that override gameplay
3. Game state management
4. Team coordination with dynamic role assignment
5. Strategy selection (e.g. ball player chooses between duel, kick at goal, pass, or dribble) based on a score system. Each possible action (shoot, pass, dribble) is given a score based on predetermined calculations, while other actions like clearing and duel are managed individually
6. Atomic skills (go to ball, walk, kick, look for ball...)

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

We are using multiple simulators:

- Webots for vision simulation
- Isaac Sim with Isaac Lab for RL training
- Isaac Sim and Webots for behaviour simulation

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

We are using ROS 2 Humble on Ubuntu 22.04.

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