Software Survey 2026

Team name

EWS Bascorro

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?

Partially open source. Selected stable modules and tools are published here https://github.com/ProgramBascorro; more modules will be released as they stabilize.

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 (reference + adaptation) like

• Field feature / line-localization baseline: inspired by / adapted from UTRA RoboSoccer's public perception and localization modules (soccer_perception and soccer_object_localization). We are adapting it to our stack by: (1) aligning camera calibration format to our OP3 setup, (2) converting extracted field features into a 2D "laser-like" observation format compatible with particle-filter localization, and (3) integrating with Webots simulation odometry for rapid iteration.

• Localization framework: ROS 2 Nav2 AMCL (off-the-shelf), integrated with our custom observation generation.

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

Yes. We are collecting our own labeled images for ball/goal detection. The dataset is not public yet (planned to release after cleaning/licensing review).

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

We are a newly established team in this league and do not yet have publications; we are

actively building the platform and baseline stack.

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

Not yet (we plan to share tooling and documentation once our baseline is stable).

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

We use the ROBOTIS OP3 walking framework (parameterized open-loop gait). Walking is generated from tunable gait parameters (step length/height/period, torso offsets, etc.) and executed through the OP3 control stack. Current focus is stability, repeatability, and systematic parameter tuning.

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

We use predefined keyframe motion sequences created with the ROBOTIS OP3 Action Editor, executed as timed joint trajectories (e.g., kick, get-up, recovery).

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 (kinematic). We use the official ROBOTIS OP3 URDF/robot description (link lengths, joint limits, joint hierarchy) for kinematic reasoning and consistency across simulation and hardware. In parallel, we are developing a custom humanoid platform; its model is still under refinement. The same ROS 2 software stack runs across all three platforms; only hardware-specific parameters (URDF, calibration, actuator maps) differ.

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

We use a YOLOv8-Tiny object detector trained for soccer ball and goal post detection. The network runs as a perception node that outputs bounding boxes + confidence per frame.

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?

Currently planning is mostly image-space (pixel offset for turning / centering; bbox scale as a rough distance cue). Planned switch to Cartesian (robot-relative) estimates by applying camera

calibration + ray/ground-plane projection back-project the detected pixel centroid using intrinsics, transform via camera extrinsics (head pan/tilt + robot kinematics), then intersect the viewing ray with the ground plane to estimate ball position ((x,y)) in meters.

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

Yes (ball-only). We implement active vision through head pan/tilt control. When the ball is detected, we servo the head to keep the target near the image center and maintain stable observations. If the ball is lost, we switch to a scanning behavior biased by the last known direction. Goal-post tracking via active vision is planned next.

What approach are you using to localize your robot?

Planned / under development a vision-based particle filter (AMCL-style) using field features. Pipeline concept segment white field lines in HSV, apply morphological filtering, project pixels to ground coordinates using calibration, convert observations into a 2D scan-like format, then fuse with odometry in a particle filter (Nav2 AMCL as the baseline). Primary goal is a reproducible baseline before adding more advanced optimization-based localization.

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

We currently use a Finite State Machine (FSM) with states such as, search -> track -> approach -> align -> kick, plus fall/recovery. We Planned to evaluate migrating parts of the decision logic to Behavior Trees for better modularity and debugging.

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

Current reactive navigation approach targets directly (ball/pose) without global planning. Obstacle avoidance currently limited (handled at behavior level). Planned add dynamic obstacle avoidance (e.g., local reactive steering) once localization + perception stability is sufficient.

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

Not implemented currently. Planned for future iterations (likely UDP broadcast compatible with league expectations once single-robot autonomy is stable).

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

Yes. We use Webots for robot/sensor/odometry simulation and RViz2 for visualization and debugging (especially localization pipeline development and tuning).

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 LTS + ROS 2 Humble.

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

N/A