# Software Survey 2026

**Team name**

Water

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

Large Size (height < 190 cm, weight < 80 kg)

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

We haven't open-sourced it yet, but we might do so in the future.

**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.**

We have not used any software developed by other teams.

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

No datasets were used in this research.

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

Not applicable.

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

No other contributions to share.

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

Reinforcement Learning.

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

Reinforcement Learning.

**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 designed the CAD model of our robot using Solidworks, and can easily convert it to the Unified Robot Description Format (URDF) via the open-source ROS package sw_urdf_exporter. Once the URDF is created, its dynamic properties can be loaded into simulations and algorithms.

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

We use YOLOv8 for real-time object detection in our robot's visual perception pipeline.

**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?**

All planning is performed in the Cartesian space, with reference to a newly defined robot base frame whose origin is set at the midpoint between the robot's feet. The x-axis points forward, the y-axis points left, and the z-axis points upward.
Given the uv coordinates of an object in image space, a ray originating from the camera can be projected using the camera's intrinsic parameters. This ray is then transformed from the camera frame to the robot's base frame using two transformations: the head-to-base frame transformation (derived from the robot's kinematics) and the camera-to-head frame transformation (obtained via extrinsic calibration).
Assuming all objects lie on the ground plane (where their z-coordinate in the base frame is zero), the length of the ray can be fixed, allowing us to compute the object's coordinates in the robot's base frame.

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

Yes, the robot's camera actively rotates to maximize the probability of detecting the ball and improve its own localization accuracy.

**What approach are you using to localize your robot?**

A particle filter based on field lines and intersections, combined with odometry based on an Inertial Measurement Unit (IMU) and state estimation.

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

No. Because in a soccer match, human players only rely on sound and visual cues for information exchange.

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

Yes, we do have a path planning module, and it is obstacle-avoidance oriented. The global path is planned using traditional sampling-based methods, such as the Bidirectional Rapidly-exploring Random Tree (Bi-RRT) with kinematic constraints. This global path is then converted into a local path using the Dynamic Window Approach (DWA) and B-spline curves.

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

We use an open-source behavior tree library (https://github.com/BehaviorTree/BehaviorTree.CPP). It receives information about the positions of the ball, goals, and obstacles from the vision node, information about the robot's position, orientation, and game state from the game controller node, angle state information (yaw, pitch) from the head node, and motion state information from the gait node. Based on this information, after calculation and processing by the decision nodes, corresponding decisions are made, and motion commands are sent to the head and gait nodes, enabling the robot to perform appropriate actions during the match.

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

No, we are not simulating our robot.

### 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

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

No, there is nothing else we would like to share.