Software Survey 2026

Team name

Invic

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?

Yes, we built our own software system based on the Bhuman framework, which can be found at https://github.com/bhuman/BHumanCodeRelease.

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, Bhuman: SimRobot.

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

Yes, but they are not public

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

It has not been published yet. We will release it once we have further results.

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?

The ZMP Preview Controller, based on a 3D linear inverted pendulum model, generates walking motions. Its core module calculates the center of mass and foot trajectory in real-time to achieve dynamic equilibrium, integrating real-time feedback from IMUs and foot sensors for

posture stabilization, arm coordination, and reactive stride adjustments. The entire system is integrated into a modular motion framework, supporting multi-gait switching and smooth motion transitions.

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

All robot motions beyond basic walking are primarily generated using predefined keyframe animations, with each action encapsulated as a reusable MotionPhase object. These phases are dynamically created by the MotionGenerator according to behavioral requirements and executed within a unified motion engine. The entire framework supports automatic phase switching, sensor feedback adaptation, and exception handling, ensuring robustness while maintaining highly modular code.

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)?

measure physical robot

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

Visual Perception Solution:
Deep learning is employed only for the most critical task (site boundary detection),
while other tasks rely on efficient classical algorithms + geometric constraints.
Deeply integrated robot models (BodyContour, CameraMatrix),
fully automated calibration and exposure, eliminating manual parameter tuning.
The entire system is specifically designed for NAO's limited CPU resources, maintaining <30ms latency while delivering exceptional environmental adaptability.

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?

Planning is performed entirely in Cartesian space (world coordinate system).
The conversion process from image space to Cartesian space is as follows:
Perform rolling shutter and motion correction on raw pixels;
Calculate the camera's pose relative to the robot using joint angles + IMU + kinematic model;

Map pixels to the robot's coordinates on the ground plane via inverse perspective projection;
Then combine with self-localization results to transform into global field coordinates.

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

Yes.

What approach are you using to localize your robot?

Self-localization employs a hybrid probabilistic framework combining "multi-hypothesis UKF + lightweight particle filtering."
It relies entirely on field markers without utilizing goalposts;
Rapidly recovers from falls or disturbances via a sensor reset mechanism;
Resolves field symmetry ambiguities using game rules + temporal continuity;
Ultimately outputs high-precision, low-latency global pose estimates, enabling accurate walking, passing, and tactical coordination.

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

Yes, use UDP

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

Its navigation system employs a layered design:
Long-range Utilizes A + visibility maps for global path planning, avoiding teammates and opponents;
Short-range or sudden situations Employs reactive local obstacle avoidance, dynamically adjusting movement direction;
Obstacle information integrates visual detection with team communication, with avoidance logic spanning both planning and execution phases.

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

Finite State Machine

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

Yes, use SimRobot, Pre-test before computer lab session

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

The NAO robot runs on a customized Linux system and utilizes a proprietary middleware framework developed by B-Human.

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