Software Survey 2026

Team name

B-Human

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?

We fully release our software on a yearly basis. It can be found at https://github.com/bhuman/BHumanCodeRelease . During the past 17 year, at least 35 RoboCup teams based their works on our software framework or used at least parts of the code we provided.

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 use a rewrite of the whistle detector developed by the SPL team Nao Devils.

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

We are using many datasets. They are available at https://www.b-human.de/datasets-en.html

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

Since the RoboCup 2024 book appeared after the last application to RoboCup, we list our papers in that book as well:

1. Mach, P., Laue, T., Hasselbring, A., Frese, U.: DiDiGen – dirty dishes generator for randomized visual training data. In: Mascarenhas, A. P. F. M., Ferrein, A. A., Villing, R. (eds.) RoboCup 2025: Robot World Cup XXVIII. Lecture Notes in Artificial Intelligence, Springer (2026), to appear
2. Reichenberg, P., Laue, T.: Stand up, NAO! Increasing the reliability of stand-up motions

through error compensation in position control (2025), https://arxiv.org/abs/2510.02129

3. Reichenberg, P., Röfer, T.: Dueling behavior leveraging advanced motion control for the NAO robot. In: Mascarenhas, A. P. F. M., Ferrein, A. A., Villing, R. (eds.) RoboCup 2025: Robot World Cup XXVIII. Lecture Notes in Artificial Intelligence, Springer (2026), to appear
4. Röfer, T., Müller-Groh, L., Hasselbring, A., Laue, T.: Automated game statistics for the RoboCup Standard Platform League. In: Barros, E., Hanna, J.P., Okada, H., Torta, E. (eds.) RoboCup 2024: Robot World Cup XXVII. Lecture Notes in Computer Science, vol. 15570, pp. 281–293. Springer (2025)
5. Röfer, T., Laue, T., Reichenberg, P., Gittner, L., Oppermann, M., Sablotny, R.: B-Human 2025 – Towards referee understanding and dynamic motions. In: Mascarenhas, A. P. F. M., Ferrein, A. A., Villing, R. (eds.) RoboCup 2025: Robot World Cup XXVIII. Lecture Notes in Artificial Intelligence, Springer (2026), to appear
6. Röfer, T., Laue, T., Reichenberg, P., Gittner, L., Schiefelbein, L.: B-Human 2024 – enhanced vision and faster ball handling. In: Barros, E., Hanna, J.P., Okada, H., Torta, E. (eds.) RoboCup 2024: Robot World Cup XXVII. Lecture Notes in Computer Science, vol. 15570, pp. 483–494. Springer (2025)

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

B-Human developed and maintained the GameController (https://github.com/RoboCup-SPL/GameController, https://github.com/RoboCup-SPL/GameController3) referee application, together with tools such as the TeamCommunicationMonitor and the LogAnalyzer. We also developed the Video Analysis App (https://github.com/bhuman/VideoAnalysis). For more than a decade now, we published the game statistics after each RoboCup and each RoboCup German Open for the Standard Platform League. We also maintain the SPLGames (https://github.com/bhuman/SPLGames) repository that links GameController logs, TeamCommunicationMonitor logs and game videos together, and can be used as input for the Video Analysis App to provide statistics that go beyond referee decisions. B-Human team members served in technical committees, in organization committees, and in the executive committee of RoboCup.

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

We use the open source reinforcement learning project from Booster Robotics Booster Gym (https://github.com/BoosterRobotics/booster_gym) as base and expanded on it.

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

For kicking, we expanded on Booster Gym to kick a given ball towards a target direction with a target ball velocity. For standup motions, we plan to use learned motions in the Booster Gym environment, too. For falling and handling the robot, we use static poses to ensure human safety.

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)?

We use the models provided by Booster Robotics (https://github.com/BoosterRobotics/booster_assets). We adapted them a little bit to ease the computation of inverse kinematics.

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

Our vision system was developed for the NAO, which required some compromises to run at full frame rate on its relatively slow processor:

- Scanline-based conventional image processing is used to find object candidates that are then classified using a self-trained neural network (ball, penalty mark). In case of the ball, the network also determines its center.
- The field border is detected by a network that gets scanlines as input.
- For robot detection, the image is scaled down significantly and used as input for a YOLO-style (self-developed) network.
- In the images of the camera, a segmentation network uses the whole image as input (we use a lower resolution for the lower camera) and outputs a low-resolution segmentation of the image (ball, obstacle, penalty mark).

We used our own library (CompiledNN) to run the neural networks. Since it creates Intel code, we will not use it anymore on the Booster robot.

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 plan in Cartesian space. We use the measurements of the IMU and the forward kinematics of the legs to determine the pose of the robot's torso. From there, we use forward kinematics to determine the pose of the camera. The distance to objects is determined by assuming that points detected in the image are located on a certain height in the world, mostly on the field plane.

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

Our head control has different modes. One of them is dynamic, i.e. the robot tries to look at close obstacles and it can be decided whether it should keep the ball in view.

What approach are you using to localize your robot?

We are using Monte Carlo localization with a very small number of samples. Each sample contains an Unscented Kalman Filter.

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

We perform team communication according to the rules of the SPL. On the one hand, our solution ensures that we stay within the message budget by using a sliding budget over the duration of the game and estimating the usefulness of sending a new packet in comparison to the last packet that was sent. We also encode the messages to send as few bytes as possible. In 2023, we played and won the SPL competition with communicating less than 4 bytes/s/robot on average.

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

We use two different methods. For longer distances, we use an A* planner that models obstacles as circles and plans a trajectory to the target on segments of these circles and the visibility graph between them. The planning is repeated in each processing cycle, which means that it ran at 60 Hz on the NAO. When closer to the target, in particular if the target is a position at the ball, we have a reactive behavior that avoids obstacles.

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

The behavior is split into three levels. The lowest level runs at the motion control frame rate (83 Hz on the NAO, 500 Hz on the Booster). It implements skills such as go-to-ball-and-kick. The second and third level run at the frame rate of the modeling thread, which is usually the frame rate at which images arrive. The second behavior level (described in CABSL) handles higher level actions on an individual robot, such as reacting to the different games states. The third level deals with the team strategy. Based on the team communication, each player determines its current role in the team and acts according to it. There are active roles and positioning roles. The roles to fill are defined in the current tactic, which is selected by the team as a whole.

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

We use our own simulator "SimRobot". It simulates both physics (based on MuJoCo) and sensor data such as camera images (using OpenGL). We simulate from individual robots up to whole games of 5v5 robots (7v7 in the SPL). All code is first developed in the simulator, before it is tested on the actual robots. This reduces the amount of time testing on the real hardware significantly. The simulator is also able to play back log files and to function as graphical frontend when connecting to real robots. For deep reinforcement learning of motions, we use Isaac Gym.

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

The Booster robots currently use Ubuntu 22.04. We use the B-Human-Framework.

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