Software Survey 2026

Team name

Berlin United

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);Middle Size (height < 125 cm, weight < 25 kg)

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

Our software is fully open source and can be found here:

https://github.com/BerlinUnited

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.

Berlin United maintains an originally developed code base, which does not incorporate any major unchanged code parts developed by other teams. Ideas and inspiration based on the code and publications by other teams are acknowledged accordingly in the team report and code where they are used.

We are using a custom Ubuntu-based OS on the NAO robots. The script for building this OS was cloned in 2021 from the original script developed by the team NaoDevils and was restructured and extended to our needs.

We are not planning to use any other code of other teams at the RoboCup
2026.

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

We are using our own datasets.

We developed and maintain a recording system that produces videos of RoboCup games that

are synchronized with the game controller. Over the past several RoboCups we recorded nearly all games. All collected videos from the past RoboCup competitions along with all log files recorded by our robots and further information can be found here:
https://berlin-united.org/project-tools.html

(the robot log files contain camera images, all sensor and behavior data)

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

"From Global Localization to Contextual Perception -- A Novel Approach to Stable Decisions in Robot Soccer" Heinrich Mellmann, RoboCup 2025: Robot World Cup XXVIII, pp., 2025, (in press)

"Anticipatory Approach to Combining Local and Global Perception for Stable Decision-Making", Heinrich Mellmann, Proceedings of the Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2024

The team also supervised Bachelor’s, Master’s, and doctoral theses focused to RoboCup research during this period. A list of our publications can be found here:
https://berlin-united.org/pages/publications.html

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

Links to all our various contributions can be found on our website:
https://berlin-united.org/

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

On the NAO, we use a ZMP based walking with modifiable spline based trajectories for the steps, allowing to dynamically change the style of a step during the walk (for instance for in-walk kick steps).

For the robot Booster K1, we use RL to train a combined policy for approaching and kicking the ball. The policy was trained in simulation (Isaac Gym and MuJoCo). Currently we are working on transferring the results to the real robot.

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

Standing up motion is keyframe based. Kicks are dynamicly generated as steps with different parameters.

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, our robots maintain a kinematic model that is continuously updated based on measured joint angles and the IMU. The dynamic components are calculated as well, but they are not used at the moment.

We are using commercially available robot's NAO V6 and Booster K1. The kinematic models are available from the manufacturer.

The kinematic chain is used to calculate a camera matrix. Some of the joints on the NAO robot vary between the robots and can also change slightly over time. To compensate for it, we developed a visual calibration routine. The robot is placed in the middle of the field and uses the perceived lines to calibrate the joints that are relevant for calculating the camera matrix.

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

Our vision system is optimized for robots with limited computational resources and is based on a combination of sparse heuristic processing and highly optimized neural networks.

We are utilizing adaptive scan lines in the images to find interesting locations. Line, center circle and goal detection are based on the scan line approach.

We use integral image and a parametric field color model to estimate the field region and its border. This can also be used to detect obstacle, but this is currently not used.

The ball detection consist of two main steps. We use the robot's kinematic chain to estimate the expected size of the ball for a given location in the image. In the first step we use the integral image to detect gaps in green with locally maximal brightness that have the expected ball size. These candidates are subsampled to a constant size of 16x16 pixels. This makes a ball

mostly scale invariant.
In the second step, the candidates are classified by a small neural network. A second neural network estimates a refined location and radius of the ball.

We might use a yolo-based ball-detection on the Booster K1.

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 done in local and global Cartesian coordinate system.
The transformation between the image space to the local robot coordinate system uses a calibrated camera matrix. The camera matrix is calculated based on the robots kinematic chain, measured joint positions, the IMU and the assumption that the robot stand on a flat surface.

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

Yes. The robot looks actively at points (ball, lines, center circle, referee) when it needs to.
Robots close to the ball actively follow its predicted position based on its previous trajectory.
During the game the robots actively look at lines and crosses to improve their localization.

When the ball is lost, the robot estimates possible locations based on previous ball model.
During a search the head reacts to single percepts. If a single percept is detected, the head movement focusses on the location of the perception for a few seconds to increase a change of detection. If no further detections follow, e.g., in case of a false positive, the search continues.
On the other hand, if the detection is confirmed then a ball model is created and the robot starts following the smoothed estimated model.

The control of the head is performed based on inverse kinematic -- the target points, e.g., a ball, are provided in the local robot coordinates. Those points are projected onto the image plane with the help of the sensor based camera matrix. Because of this, the gaze of the robot is automatically stabilized during the walk.

A dedicated attention engine is also implemented and will most likely be used at the RoboCup

2026.

What approach are you using to localize your robot?

We are using a probabilistic particle filter (Monte Carlo Particle Filter). As features, we mainly use lines and the center circle. The odometry is estimated online based on the measured joint angles, the kinematic chain and the IMU.

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

We use Wi-Fi for communication, following the SPL rules from 2025. UDP based broadcast with a maximum message budget of 1200 package per game for the entire team.

Each message is encoded using Protobuf and contains only necessary information, which can vary depending on the situation.

The messages are sent based on the situation (event based communication). For example, a robot will only send an update about its location to the team, if this robot has moved significantly since last message, e.g., more than 0.5m. A striker sends messages more often to keep the team updated. The robots also actively monitor the total amount of message and limit their communication in case of a danger that the budget would be exceeded.

No sound communication is used at the moment.

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

The robot navigates based on a reactive 2D step plan. A number of different step plans for different situations, like approaching the ball or walking to a position, are available. The robot switches between different planers depending on the situation. All planers work on the same step buffer and create a coherent walking trajectory.

Reactive obstacle avoidance is realized based on the NAOs ultrasonic sensors.
Visual obstacle detection is implemented, but is currently not used. It might be used at the RoboCup 2026.

To detect collisions, the robot monitors the joint position error in its shoulders. If a collision is detected the arms are moved to avoid entangling.

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

On the higher behavior level we use hierarchical state machines (XABSL) to structure our behavior.

On the lower level, we use a predictive action selection based on internal simulation to decide on the best kick action. The robot simulates possible final ball locations after a kick and selects the kick that promises the best outcome, e.g., a ball ends up in a goal. The internal simulation stochastically predicts the movement of the ball and possible collisions. Based on the predictions, statistical expected values for main event such as scoring a goal r kicking the ball out are estimated and used to make a decision.

A reactive step planer is used to execute the kicks, dribbling and navigating on the field.

For team coordination, our robots employ a highly dynamic team strategy,
including role negotiations between players and assignment of locations on the
field. The role assignments depend on the importance of the roles and the sit-
uation in the field. For instance, more important roles (goalie, striker) are prioritized and the overall walking load of the team is optimized.
The parameters for the team strategy were optimized based on a large number of simulated SPL games, which were transferred and used in real RoboCup games.

Currently, we are working on implement the team coordination based on the internal simulation, as described above. A robot can make a role decision by simulating and predicting the actions of other players, which reduces the necessity of the explicit communication.

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

For experiments with the Booster we use Isaac Gym and MuJoCo with the provided environment to train walking and kicking behaviors.

We use our version of Simspark (Simulator of the Simulation 3D League) that was adapted to the SPL environment to simulate and debug team behaviors. We use Webots to simulate single robot behaviors.

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

As the main middleware we maintain our own module framework with all necessary components for serialization, communication, low level device access and others.

On the NAO we use a custom Ubuntu 20.04 with the NAO's proprietary low level middleware "lola" to sent and receive sensor/actuator values.

We are currently working on generalizing our Framework to work with both NAO and Booster K1. This work is in early stages at the moment.

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

We aim to use largely the same code base on Booster K1 and the NAO robots. Some of the modules, such as walking, will differ, but the main core and the majority of the modules should be the same.