Software Survey 2026

Team name

Hamburg Bit-Bots

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?

Our software is fully open-source and available at https://github.com/bit-bots/

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.

Our walking is based on IKWalk from Rhoban. We use the onshape-to-robot tool for converting our CAD model to a URDF.

We use the rot_conv_lib from Nimbro for using Fused Angles in our code. Our goalie behavior is based on the one from BHuman.

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

We use the TORSO21 dataset (https://github.com/bit-bots/TORSO_21_dataset) we published ourselves.

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

Vahl, F., Griepenburg, J., Gutsche, J., Güldenstein, J., & Zhang, J. (2025, July). SoccerDiffusion: Toward Learning End-to-End Humanoid Robot Soccer from Gameplay Recordings. RoboCup Symposium.

Güldenstein, J., Vahl, F., Stahl, B., Wedmann, L., & Zhang, J. (2025, July). YOLO-BF: Enhancing Robot Detection and Localization using Base Footprint Point Detection. RoboCup Symposium.

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

Open-sourcing our code has allowed several teams to use our algorithms (e.g. NUbots using our walking).
Our dataset TORSO21 includes 10464 fully annotated real world images to allow for comparing vision approaches in RoboCup soccer.
Additionally, we contribute to ROS-Sports, which provides standard software solutions for common tasks in the RoboCup.

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

We use an approach based on Rhoban's IKwalk, but heavily modified. It is based on quintic splines generating Cartesian trajectories. Parameters defining these splines are optimized in simulation and adapted by an expert in the real world. The approach is described in: Marc Bestmann, and Jianwei Zhang "Bipedal Walking on Humanoid Robots through Parameter Optimization" RoboCup Symposium 2022, Bangkok, Thailand

We also have an alternative Reinforcement Learning based walking approach which we train in Mujoco Playground: Zakka, Kevin, et al. "Mujoco playground." arXiv preprint arXiv:2502.08844 (2025). The latter approach is the one we will most likely use at the RoboCup tournament.

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

For our kicks we use our walk engine to generate normal steps that include a spline based Cartesian kick trajectory.

We also use Mujoco Playground to learn a kick Motion using Reinforcement Learning.

For our stand up motion we use a combination of a joint space spline keyframe animation which is recorded by puppeteering the real robot and an IK based rise movement in Cartesian space from the squat position onward. Our protective falling reaction is active during the later part of the standup motion, so damages from falls during a standup (e.g. due to a collision with another robot) are minimized.

We generate different simple patterns as well as Cartesian look at actions for the head. We also use a throw-in motion which is developed by puppeteering our robot similarly to our stand-up motion.

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 have a kinematic and dynamic model for the Wolfgang-OP in the URDF and Webots .proto format. The URDF is exported from the CAD software we use, onshape.com, using https://github.com/Rhoban/onshape-to-robot. We used https://github.com/cyberbotics/urdf2webots to convert the URDF to a Webots .proto. Manual adjustments were required. We also have a URDF kinematic and dynamic model for the X02, however this model is not publicly available.

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

We use a custom made derivative of the YOLO and U-Net architecture called YOEO (https://github.com/bit-bots/YOEO), which we train on the TORSO21 dataset.

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 coordinates, depending on application relative to the robot or relative to the playing field. We do inverse perspective mapping (https://github.com/ros-sports/soccer_ipm/) using the orientation measured by an Inertial Measurement Unit and forward kinematics.

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

Our head moves based on the state of the behavior. Several head modes such as looking around to detect obstacles or tracking the ball are implemented.

What approach are you using to localize your robot?

We use a particle filter whose particles are updated by matching a point cloud of observed line points with a precomputed distance map. Additionally, we work on image space localization using neural networks.

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

We are performing team communication using the standard Protobuf based protocol.

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

Our robot plans a path through the environment around obstacles. A* is used for generating a global plan. A carrot planner is used to execute this plan. In ongoing research, we are looking to replace this approach with a reinforcement learned footstep planner policy.

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

We use our own behavior framework called Dynamic Stack Decider described in:
Poppinga, Martin, and Marc Bestmann. "DSD-Dynamic Stack Decider." International Journal of Social Robotics 14.1 (2022): 73-83.
Furthermore, we employ a potential field method incorporating detected opponent robots to decide if we should dribble or pass.

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

We use Webots and Mujoco for several machine learning algorithms: Parameter optimization of motion algorithms, footstep planning using Reinforcement Learning, odometry correction.
We also use Mujoco Playground for Reinforcement Learning to develop walking, kicking and stand-up motions.

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

We are using ROS2 Jazzy on Ubuntu 24.04.

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