Team Description Paper

Johannes Blum[1], Paul Dargel, Ole Felber, Franziska-Sophie Göttsch[0009-0001-9055-217X], Phillip Kammradt, Ben Sampaolo, Alexander Schmidt, Maximilian Schmidt[0009-0005-4532-7669], and Vivienne Schwabe

[1] Hamburg University of Technology, Am Schwarzenberg-Campus 1, 21073 Hamburg

1 Team Information

Team Name	HULKs
Team Leader	Johannes Blum
Team Contact Email	hulks@tuhh.de
Team Website	https://www.hulks.de and https://github.com/hulks/
Country of Origin	Germany
University Affiliation	Hamburg University of Technology

2 Lessons Learned

A key lesson from previous tournaments is that software stability is just as crucial as strategy. We observed that our Rust framework offered a significant reliability advantage, enabling us to implement rapid changes to our codebase while still maintaining a functional and responsive robot.

Building on this stability, we have learned to integrate a high-performance robotics on a low level. Through years of competition, we have cultivated a robust practical understanding of the physical interplay between kinematics, vision, the walking engine, and high-level behavior, allowing us to tightly couple these systems for more agile and responsive gameplay.

3 Challenges

The major challenge this season is adapting our existing Rust code-base from the NAO robot to the Booster K1 platform. To achieve this, we need to develop a thorough understanding of the robot's functionality and the interface provided by Booster Robotics and find a way to establish effective communication between our code and the ROS2 speaking Booster Nodes. Additionally, we aim to update and replace existing features like a new feature detection and newly learned walking to fully utilize and accommodate the new hardware.

4 Current implementation status and Plans

The new hardware capabilities of the Booster Robotics K1′s NVIDIA Jetson Orin NX and its GPU allows for more powerful machine vision to be used on the robot. Because of this, we are completely redesigning our vision and localization pipeline.

4.1 Robot Integration

For the integration of our robotics code onto the robot, we implemented an interface to ROS2 using the network protocol zenoh.

4.2 Feature Detection

We integrated the inference engine of a pre-trained YOLOv11 Object Detection model into our robotics framework. The system is designed for hybrid flexibility: It runs on CPU for efficient development and simulation, while leveraging hardware acceleration via TensorRT on NVIDIA Jetson GPUs for high-performance deployment on the physical robot.

Building on the pre-trained YOLOv11 weights, we fine-tune the model to robustly detect game-critical features necessary for localization and decision-making. Key detection targets include line intersections, the ball, goal posts, the referee, and robots (both allied and opponent). Initial experiments using NAO imagery demonstrate strong generalization. We plan to expand this dataset with real-world footage from the Booster K1 as gameplay data becomes available. Future iterations will extend beyond bounding boxes to include pose estimation (e.g. for referees) and semantic segmentation.

To support a multitude of vision tasks, including oriented object detection, pose estimation, and segmentation, we are transitioning to a unified multi-task model. This architecture utilizes a shared backbone (evaluating YOLOv11 vs. the newer YOLO26) to drive multiple task-specific heads. This "single-backbone, multi-head" approach significantly reduces computational overhead compared to running separate models for each task.

To address the scarcity of labeled real-world gameplay footage, we are augmenting our training set with synthetic data generated in the game engine Unity. We employ domain randomization to bridge the sim-to-real gap. By modeling the field, surroundings, and robots in 3D and randomizing scene parameters (lighting, textures, camera angles), Unity automatically renders diverse scenes with pixel-perfect ground truth annotations. This pipeline enables the rapid generation of large-scale datasets without manual labeling effort.

4.3 Localization

For localization, we are implementing a 3D-SLAM approach utilizing a particle filter. Having transitioned from the resource-constrained NAO to the Booster K1, we can now deploy this computationally intensive method to handle complex 3D geometry during match play. This upgrade enables us to extend localization

beyond reliance on standard field markings (lines and penalty spots) by integrating additional 3D features.

4.4 Motion

We started implementing walking functionality for the robot by using a reinforcement learning (RL) walking policy developed and released by Booster Robotics. Moreover we are planning on using their RL policies for standing up. Additionally, we are working on training our own RL policies. These currently include policies for walking, kicking the ball and standing up. The inference of these RL motion policies is done in our framework.

4.5 Behavior

Our objective for RoboCup 2026 is to adapt our existing behavioral frameworks to the new platform. By migrating our software, we aim to utilize the enhanced capabilities of the Booster K1 robot to ensure stable and efficient gameplay. Our priority is to successfully translate our established logic to the new hardware environment to achieve a reliable performance on the field.

At the time of this application, we have successfully established the foundation of our behavior on the new platform. The robot is currently able to detect the ball, maintain visual focus, and perform autonomous approach maneuvers.

4.6 Framework

Our software stack relies on `hulkz`, a custom middleware framework written entirely in Rust. By leveraging Rust's ownership model, we ensure compile-time memory safety and efficient concurrency management. This choice addresses common pitfalls inherent in C++ based stacks, specifically regarding memory leaks and thread safety, resulting in significantly higher system stability.

The architecture is built around a distributed system where zenoh serves as the backbone network. Each processing unit operates as a node (a zenoh session) capable of publishing and subscribing to data topics. To interface with the Booster K1 hardware, which exposes sensor data and expects actuator commands via ROS2 DDS, we implement a custom bridge. This bridge handles the translation between the ROS2 ecosystem and zenoh using Common Data Representation (CDR), ensuring binary compatibility and efficient serialization.

We chose zenoh for its low-latency and high-throughput capabilities, which are essential for real-time robotic control. Internal benchmarks demonstrate that zenoh handles high message rates with minimal overhead, providing a performance advantage over standard DDS implementations in resource-constrained environments. Zenoh allows for real-time introspection of data flowing through the network. Developers can connect local clients to inspect topics, data rates, and message contents, allowing for rapid diagnosis of issues and performance optimization.

Accurate state estimation requires precise time synchronization. Unlike systems that simply consume the latest available sample, our framework buffers

received data into a time-indexed cache. This allows to look up nearest queries in time, enabling the matching of messages based on timestamps to ensure accurate sensor fusion.

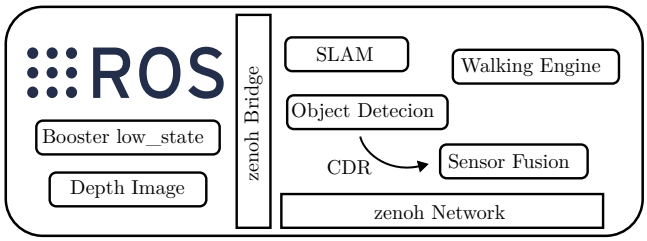

Fig. 1. Overview of the network stack, composed of ROS2 and zenoh interoperable protocol architecture.

4.7 Simulator

To quickly test our developed features we build a simulator based on MuJoCo and the K1 model provided by Booster Robotics. We also integrated a visualizer for this simulator into our existing debug tooling.

5 Own Contributions

Our code base is fully open source and serves as inspiration for other teams. In addition we plan to release a synthetically generated data set with bounding boxes around multiple game features, pose detection for robots and referee, segmentation masks and depth information.

Additionally, the work on a unified multi-task detection model is done in the scope of a project thesis by members of the team and the resulting thesis is planned to be publishable.

6 Impact

The HULKs contribute significantly to the Humanoid Soccer League ecosystem by facilitating community exchange and promoting open research practices. We organized the 12th edition of the Robotic Hamburg Open Workshop (RoHOW) in November 2025, which provided the league with a crucial first opportunity to exchange knowledge regarding the new robot platform in a larger context. Furthermore, as the first team to adopt a fully open-source model, we pioneer collaborative research by sharing our complete framework under the GNU GPLv3. This offers a robust alternative to established C++ stacks, lowering the barrier to entry for new teams wishing to utilize modern programming languages.