Software Survey 2026

Team name

rUNSWift

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?

Fully open source. The 2025 code has not yet been released, but the 2024 code can be found at: https://github.com/UNSWComputing/rUNSWift-2024-release. The 2025 code was a complete rewrite for ROS 2.

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 are using Booster's sample RoboCup code for walking and vision: https://github.com/BoosterRobotics/robocup_demo

and HTWK's adaption of BoosterGym for reinforcement learning: https://github.com/NaoHTWK/htwk-gym

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

Currently we are not using any other datasets, but may acquire our own vision 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).

T. Wiley and C. Sammut, Data efficient online learning of robot behaviours via quali-tative planning and reinforcement learning, Robotics and Autonomous Systems, vol. 194, Dec (2025).

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

We developed the walking engine that became the basis of most of the SPL teams. A movel reinforcement learning algorithm was used to train a NAO. The speed and agility of the walking engine was a major contributor to our wins in 2014 and 2015. Subsequently, B-Human took our code and improved on it, so that is the code that many of the other teams have used.

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

Initially, we are using the pre-trained model in the Booster demo code

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

We are using BoosterGym and HTWK-Gym to train new kicks and goalie actions

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)?

Provided by Booster

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

Initially, using the pre-trained YOLO vision model in the Booster demo code. Later, we will train our own models, but with limited preparation time, we are starting to with the provided model.

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?

From kinematics, we know the pose of the camera, so all vision information is transformed from the camera's frame to the robot's map coordinates. This is used for all game play behaviours, except when the robot has to line up for kicks, in that adjustments used camera data directly.

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

If confidence in localisation falls below a threshold, the robot can look around to find landmarks. If the ball is lost, we use our best guess at the direction in which it was lost to look there first to save time in the "findball" routine.

What approach are you using to localize your robot?

We used a multi-modal Kalman filter that was developed in 2014 and had been quite reliable. However, this relies on a fair amount of communication between robots as it was design to be distributed filter. As the SPL began enforcing packet limits, this became a problem for this approach. We have experimented with particle filters, but still prefer the distributed Kalman filter if we can reduce inter-robot communication.

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

In SPL communication used UDP. Each robot would broadcast its map so that the other robots could use this information in their Kalman update. With the introduction of the packet limits, the amount and methods of broadcast has change to be event driven rather than periodic.

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

The NAO robots have ultrasonic sensors which were used in close quarters to avoid opponents and team members. Vision was used for longer range obstacle avoidance. The reliability depended on the ability of the vision system to distinguish other robots. Moving to the K1, its vision model has been quite reliable so far and has the advantage of a stereo camera to get the distance to obstacles.

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

The behaviours on the NAO are coded as a custom decision tree, implemented in Python. The Booster demo behaviours are implemented as behaviour trees, which we have found to be cumbersome. We are, instead, porting our implementation of a Teleo-Reactive programming language, which allows us to describe universal plans for the robot's behaviours.

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

We are using Booster's models in Isaac Sym. Currently, this is for training new kicks. If time permits, we would like to simulate games to game play behaviour.

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

Ubuntu 22.04, ROS 2 Humble

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