Software Survey 2025

Team Name

UT Austin Villa

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

We are building on top of an open-sourced RoboCup Demo (https://github.com/BoosterRobotics/robocup_demo). We also built our own reinforcement learning training pipeline in IsaacLab. We plan to open source our own software system later.

Do you have a kinematic or dynamic model of your robot(s)? If so, how did you create it (e.g. measure physical robot, export from CAD model)?

Yes, we have the kinematic model in Unified Robot Description Format (URDF). Our robot is a commercial Booster T1 humanoid robot. The kinematic model is publicly available.

Are you using Inverse Kinematics? If so what solution (analytic, (pseudo)inverse jabcobian, etc...) are you using?

No, we don't use Inverse Kinematics.

Are you simulating your robot? If so what are you using simulation for?

Our major simulation is IsaacSim, which we use to train the RL policy. We also use IsaacGym and MuJoCo for sim-to-sim transfer.

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

Reinforcement Learning

What approach are you using to generate motions for standing up?

Reinforcement Learning

What approach are you using to generate kicking motions?

Reinforcement Learning

Do you use any other motions than the previously mentioned? If so, what approaches are you using to generate them?

No

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

No

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

We use YOLO v8 for ball and marker detection and pose estimation.

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?

Yes, the current system plans in Cartesian space.

Specifically, we build upon Booster Robotics' perception pipeline, which utilizes the RealSense camera's intrinsic and extrinsic parameters to transform image-space pixel coordinates into Cartesian positions in the robot's base frame. The base frame is defined at the midpoint between the two feet, with the x-axis pointing forward, the y-axis pointing left, and the z-axis pointing upward.

Booster's existing pipeline computes the positions of detected objects in the base frame by projecting a ray through the center of the object's bounding box. This ray is transformed from the camera frame to the base frame using the head-to-base transform, which is derived from the robot's kinematics, and the camera-to-head transform, which is obtained from extrinsic calibration. Assuming all objects lie on the ground with z=0, their x and y coordinates in the base frame can then be computed.

We have observed significant noise in this method, particularly when the robot is walking. To mitigate these issues, we have developed two methods that leverage RealSense's depth image aligned to the RGB frame. When the robot is in motion and

one of its feet is in the air, the kinematic-based head-to-base transform becomes unreliable. In such cases, we estimate and correct the camera-to-base rotation and height by fitting a ground plane to the point cloud. Additionally, instead of assuming that objects are on the ground, their 3D positions can be directly computed using depth information. The depth value can be taken from either the center pixel or the bottom-center pixel in the bounding box (depending on the object class), after which the extracted 3D point is transformed into the base frame.
We are currently evaluating the performance of these methods in the accuracy of object positions in the base frame.

How is your robot localizing?

We used a localization system combining the perception and odometry. It is built upon the RoboCup Demo from Booster Robotics and the localization approach open-sourced by BHuman SPL team.

Is your robot planning a path for navigation? Is it avoiding obstacles? How is the plan executed by the robot (e.g. dynamic window approach)?

We don't plan a path for the navigation. The motion plan in the current strike behavior is generated using simple heuristic, e.g. assigning linear velocity pointing to the ball, and angular velocity rotating the robot to face the ball. We are training an end-to-end strike skill policy which approaches and strikes the ball by directly performing low-level joint control without planning a path.

How is the behavior of your robot's structured (e.g. Behavior Trees)? What additional approaches are you using?

We use behavior trees to structure the high-level behavior and integrate policies trained with RL.

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

Yes, the camera tracks the ball, or when the localization system confidence is low, it rescans the field.

Do you apply some form of filtering on the detected objects (e. g.

Kalman filter for ball position)?

We are working on integrating a Kalman filter for ball tracking.

Is your team performing team communication? Are you using the standard RoboCup Humanoid League protocol? If not, why (e.g. it is missing something you need)?

We are not performing communication at the moment. We plan to develop team communication with the standard RoboCup Humanoid League protocol, but our immediate short-term plan is to learn an end-to-end strike skill and implement a better perception system.

Please list contributions your team has made to RoboCup

UT Austin Villa team boasts a long history in RoboCup competitions. We began our journey in 2003 with the Legged League, later transitioning to both the Simulation League and the Standard Platform League in 2008. We won countless champions in 3D Simulation and SPL leagues. We have also been participated in RoboCup@Home DSPL since 2017. Many of our team members have served on the RoboCup committees.

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

Please refer to this link for the list of scientific publications our team has made: https://www.cs.utexas.edu/~AustinVilla/?p=papers.

Please list the approaches, hardware designs, or code your team is using which were developed by other teams.

robocup_demo: https://github.com/BoosterRobotics/robocup_demo

booster_gym: https://github.com/BoosterRobotics/booster_gym

BHuman: https://github.com/bhuman

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 and ROS2 Humble

Is there anything else you would like to share that did not fit to

the previous questions?

If you have additional materials you would like to show, please link to them here.