Survey response 7

Software

Team Name
Rhoban

Is your software fully or partially OpenSource. If so, where can it be found:
Some parts are open-source Not all of it, mostly because it represents a huge time investment to do it properly Our tool for CAD-to-URDF conversion: https://onshape-to-robot.readthedocs.io/ Our internal quadratic programming based inverse kinematics library: https://placo.readthedocs.io/ The URDF model of our robot is included in examples models here: https://github.com/Rhoban/placo-examples/tree/master/models

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)?
We have an URDF model that is exported from our CAD model using Onshape-To-Robot It includes robot geometry, dynamics, pure shape approximation for physics simulation, and frames of interrest https://onshape-to-robot.readthedocs.io/

Are you using Inverse Kinematics? If so what solution (analytic, (pseudo)inverse jabcobian, etc...) are you using?
We use quadratic programming solver, which is equivalent to a regularized and constrained pseudo-inverse. We recently open-sourced and documented PlaCo, our library to achieve such task-space inverse kinematics, some examples (including an humanoid with few tasks) can be found here: https://placo.readthedocs.io/en/latest/kinematics/examples_gallery.html

Are you simulating your robot? If so what are you using simulation for?
We use 3D geometrical simulations to test/debug behaviours We use dynamics simulation (pyBullet) to test/debug some motion

What approach are you using to generate the robot walking motion?
Footsteps planning are generated using reinforcement learning, trained against a purely geometrical problem formulation We then plan the center of mass trajectory using constrained optimization Finally, the trajectories are injected in a whole body inverse kinematics solver

What approach are you using to generate motions for standing up?
Standing up is based on manually tuned (piecewise linear) splines

What approach are you using to generate kicking motions?
Kicking motion is based on manually tuned (piecewise linear) splines

Do you use any other motions than the previously mentioned? If so, what approaches are you using to generate them?
We have a separate analytical inverse kinematics for the head

Which datasets are you using in your research? If you are using your own datasets, are they public?
We have a custom annotated images dataset for detection, it is currently not public

What approaches are you using in your robot's visual perception?
We use the end-to-end network YoloV8

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 space We performed cameras calibration using OpenCV Charuco board to obtain intrinsic matrix Extrinsic matrix is obtained using the kinematics model of the robot Pixels then give you a ray that you can express in a world frame, which you can for example intersect with the ground

How is your robot localizing?
The features detected with Yolo are fed to a custom particle filter. Odometry (based on kinematics model and gyroscope integration) is extensively used for particles mutation.

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)?
The robots are planning path using A*. The footsteps are planned to aim at a point further along the path, but is itself not considering obstacles. We are currently working on a better approach for that purpose.

How is the behavior of your robot's structured (e.g. Behavior Trees)? What additional approaches are you using?
Following the rules is currently mainly achieved by state machines. Strategy is based on classical dynamic programming and tree search on simplified and discretized version of the problem.

Do you have some form of active vision (i.e. moving the robots camera based on information known about the world)?
Yes, the camera stares at the ball most of the time by using kinematics model and the estimated position of the ball in the world. When the robot scans for localization information and look back at the ball, we use the head inverse kinematics to look at the ball again.

Do you apply some form of filtering on the detected objects (e. g. Kalman filter for ball position)?
We apply some form of filtering, but they are currently not very formal, mostly moving averages with some heuristic to merge similar candidates.

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)?
The robots are communicating using mostly protobuf serialized messages over UDP broadcasts. We didn't implement the standard RC Humanoid League Protocol, mostly for historical reasons

Please list contributions your team has made to RoboCup
I don't understand this question (If you refer to software contributions, it was already asked before)

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

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

What operating system is running on your robot and which middleware are you using (for example Ubuntu 22.04 and ROS2 Galactic)?
We are running Ubuntu 22.04 We don't use ROS

Is there anything else you would like to share that did not fit to the previous questions?