Survey response 5

Software

Team Name
CIT Brains

Is your software fully or partially OpenSource. If so, where can it be found:
Our Software is partially OpenSource. It can all be found in the CIT Brains organization on GitHub (https://github.com/citbrains).

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)?
I have a model to use for simulation on webots, I created a proto file for webots from a CAD model created in AUTODESK Inventor using a conversion tool. The Collision Box was created by a human. The proto-model is available in the following repository.

Are you using Inverse Kinematics? If so what solution (analytic, (pseudo)inverse jabcobian, etc...) are you using?
We solve by a geometric method using Inverse Kinematics. The reason we use this method is that our walking pattern generator only needs to solve the inverse kinematics of the legs and not the whole body.

Are you simulating your robot? If so what are you using simulation for?
We are simulating our robot with Webots. The reason we simulate our robot is the robot's walking control and image recognition, and to simulate a four-a-side soccer match.

What approach are you using to generate the robot walking motion?
We are using a very computationally efficient algorithm for gait pattern generation. This is because computational resources for robots are not abundant. The ZMP trajectory and center of mass trajectory are stored as tables calculated offline. When walking, the table is multiplied by coefficients according to the walking parameters specified by other modules to generate a walking with hardly repetitive computation. Also, The system has tables for longitudinal, transverse, and turning walking, so it can generate walking in all directions.

What approach are you using to generate motions for standing up?
Except for the walking motion, motions are created manually based on keyframes. Keyframes are a method of specifying the joint angles of all joints of a robot one frame at a time. For each motion, 16 frames are prepared manually, and the motion is generated by complementing and replaying between the frames. The motion for standing up is prepared for four different orientations: prone, supine, lying on the right side, and lying on the left side.

What approach are you using to generate kicking motions?
The same approach is taken with the motion for standing up. Two types of kicking motions are created and used in the game: forward and diagonally forward kicking.

Do you use any other motions than the previously mentioned? If so, what approaches are you using to generate them?
We do not use any other motions than the previously mentioned ones.

Which datasets are you using in your research? If you are using your own datasets, are they public?
We use our own datasets. They are not publicly available.

What approaches are you using in your robot's visual perception?
We use two approaches to visual recognition. One is object detection using YOLO and the other is pixel-level semantic segmentation, both based on deep learning. In object detection, we detect balls, goalposts, and robots. On the other hand, pixel-level semantic segmentation detects green areas and white lines on a soccer field.

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?
Objects are planned in Cartesian space. The transformation between image space and Cartesian space is performed using the camera posture calculated by the angle of each joint of the robot and the length of each link, and the camera matrix and distortion coefficients calculated by the camera calibration.

How is your robot localizing?
We employ Monte Carlo Localization for self-localization. Landmark measurements from the abovementioned computer vision system are fused with motion predictions from kinematic odometry using a particle filter. The landmarks employed are goalposts and white lines on the field.

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 are doing path planning to avoid obstacles. We construct a visibility graph with the soccer field and the robots as obstacles as nodes. The A* algorithm is applied to the visibility graph to plan the path. In order to avoid moving obstacles (e.g., robots), the path planning is frequently redone based on obstacle information that is updated at approximately 10 Hz.

How is the behavior of your robot's structured (e.g. Behavior Trees)? What additional approaches are you using?
Our robot behavior is structured by the HTN Planner (Hierarchical Task Network, Troy Humphreys (2013). Exploring HTN Planners through Example, Steve Rabin, Game AI Pro. CRC Press, pp.149-167.). We have developed a field search algorithm based on frontier search for robots to efficiently search for lost balls. The search for a ball is performed when none of our robots can find the ball to avoid the Dropped Ball game state. The robots play three roles: Attacker, Defender, and Goalkeeper. With the exception of the Goalkeeper, the role are determined dynamically according to the situation. The role of the Attacker is to score a goal by approaching the ball and kicking it toward the goal. The role of the Defender is to position himself between the ball and the goal and prevent his own goal. The role of the Goalkeeper is to defend its goal against an oncoming ball.

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

Do you apply some form of filtering on the detected objects (e. g. Kalman filter for ball position)?
We use a moving average filter.

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 perform team communication. However, we do not use standard protocols. The reason for this is that standard protocols do not contain the information needed for our robot's action plan. We use Protocol Buffers (https://developers.google.com/protocol-buffers) to serialize the data we send.

Please list contributions your team has made to RoboCup
Participated in RoboCup KidSize Humanoid League every year from 2007 to 2023 (except 2020 when it was cancelled). Our main contributions to the RoboCup community are the translation of the RoboCup rules into Japanese (https://github.com/citbrains/RoboCupRule) and the release of the simulation environment for the team's robot(https://github.com/citbrains/GankenKun_webots).

Please list the scientific publications your team has made since the last application to RoboCup (or if not applicable in the last 2 years).
RoboCup2022 KidSize League Winner CIT Brains: Open Platform Hardware SUSTAINA-OP and Software , Yasuo Hayashibara, Masato Kubotera, Hayato Kambe, Gaku Kuwano, Dan Sato, Hiroki Noguchi, Riku Yokoo, Satoshi Inoue, Yuta Mibuchi and Kiyoshi Iri Kubotera, M., Hayashibara, Y.: Sustaina-op™: Kid-sized open hardware platform humanoid robot with emphasis on sustainability. RoboCup 2023: Robot World CupXXVI (In Press)

Please list the approaches, hardware designs, or code your team is using which were developed by other teams.
We designed a piece of hardware based on "High-Frequency Multi Bus Servo and Sensor Communication Using the Dynamixel Protocol, Marc Bestmann, Jasper G¨uldenstein, and Jianwei Zhang" We designed a piece of hardware based on “G. Passault, Q. Rouxel, L. Hofer, S. N'Guyen and O. Ly, "Low-cost force sensors for small size humanoid robot," 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, Korea (South), 2015, pp. 1148-1148, doi: 10.1109/HUMANOIDS.2015.7363498.”

What operating system is running on your robot and which middleware are you using (for example Ubuntu 22.04 and ROS2 Galactic)?
We use Ubuntu 20.04 as our operating system. We basically do not use any middleware such as ROS. ZeroMQ (https://zeromq.org/) and ZeroC ICE (https://zeroc.com/products/ice) are used as interprocess communication libraries.

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