Software Survey 2025

Team Name

RoMeLa UCLA

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

The software run on ARTEMIS is currently not open source. There may be future plans to open source the software in order to include more community driven development once the platform has reached a sufficient level of stability and performance for others to use. All future open source implementations will be hosted on the RoMeLa UCLA github page. [1]

[1] RoMeLa UCLA github page. https://github.com/RoMeLaUCLA

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)?

The kinematic and dynamic models are computed using the open source pinocchio c++ library created at LAAS CNRS [1]. A Solidworks assembly is converted into a URDF through the open source ROS sw_urdf_exporter [2]. Once the URDF has been created, the dynamic properties are confirmed with hardware measurements. During runtime, the URDF is loaded into pinocchio for computation of kinematic and dynamic terms using popularized algorithms like the Composite Rigid Body Algorithm (CRBA) and the Recursive Newton Euler Algorithm (RNEA).

[1] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, “The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives,” in International Symposium on System Integration (SII).
[2] Solidworks to URDF Exporter. http://wiki.ros.org/sw_urdf_exporter

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

Inverse kinematics is solved using the popularized Damped Least Squares method outline in Buss et al. [1]. Each leg/arm is treated as a fixed base system relative to the body. Planners specify task motions in the body frame which the inverse kinematics converts into required joint angles.

[1] Buss S. R.:Introduction to Inverse Kinematics with Ja-cobian Transpose, Pseudoinverse and Damped Least Squaresmethods. Tech. rep., University of California, 2009

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

Locomotion and dynamic motion algorithms are simulated using the Gazebo simulator [1]. Using the previously mentioned URDF generation scheme, an SDF model is created using the Gazebo tools. An in-house Gazebo model plugin is then loaded in the model file to read/command joints, read body and world estimates, and impart disturbance forces on the body. This plugin followed the plugin tutorials on the Gazebo home page [2], but has expanded to include more specific needs of the robot.

Since ROS 2 is heavily integrated into Gazebo, there is an ongoing effort to switch towards using ROS 2 gazebo_ros_pkgs, so that the broader robotics community will have access towards familiar simulations setups. This process is expected to finish after the 2023 competition as higher priority objectives must be completed first.

[1] Gazebo Simulator webpage. https://gazebosim.org/home
[2] Gazebo Model Plugin Tutorial: https://classic.gazebosim.org/tutorials?tut=guided_i5&cat=

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

Due to the line contact feet and 5 DoF legs, ARTEMIS can only be dynamically stabilized to achieve locomotion. In order to do this, a hierarchical control approach is used. At the highest level, template models feed easy to compute trajectories to a mid level model predictive controller (MPC) to compute more achievable body trajectories. The use of the MPC gives the added benefit of providing a time horizon for the MPC to

preempt any sudden, dynamic movements that are planned to occur [1]. The desired body trajectories are then fed to a lower level whole body controller (WBC) to generate desired joint torques at a given time instant based on the floating base dynamics and ground reaction force constraints. The WBC chosen follows the Implicit Hierarchical Whole Body Controller (IHWBC) approach which has seen success on other humanoid robots [2], [3]. The optimized joint torques are then executed by joint level PID controllers to enforce proper tracking. By moving towards this more dynamic form of locomotion, it is hoped that ARTEMIS achieves faster locomotion than the current standard in RoboCup. This locomotion stack currently allows ARTEMIS to achieve walking and running (with air time) at speeds of 0.5 m/s untethered and 1.6 m/s tethered.

[1] Chignoli, M., Kim, D., Stanger-Jones, E., & Kim, S. (2021, July). The MIT humanoid robot: Design, motion planning, and control for acrobatic behaviors. In 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids) (pp. 1-8). IEEE.
[2] Shen, J. (2022). Locomotion Analysis and Control of a Miniature Bipedal Robot. UCLA. ProQuest ID: Shen_ucla_0031D_21422. Merritt ID: ark:/13030/m5bw4t2s. Retrieved from https://escholarship.org/uc/item/52q9d7vk
[3] J. Ahn, S. J. Jorgensen, S. H. Bang, and L. Sentis, “Versatile Locomotion Planning and Control for Humanoid Robots,” Frontiers in Robotics and AI, vol. 8, 2021

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

Standing up from rest is currently not a part of the control structure. If time allows before the competition, hand tuned trajectories will be first investigated to see if a simple, yet reliable configuration can be found to stand up from.

What approach are you using to generate kicking motions?

A basic dribbling strategy has been accomplished through using a robust locomotion stack and the robot simply reacting to the ball as an external disturbance. While undirected, ARTEMIS can currently walk into a soccer ball without falling over. Moreover, a standing kicking motion has been developed by transitioning the swing leg trajectory from a 5th order quintic spline to a 7th order. This modification allows for the

incorporation of waypoints and manual tuning of the trajectory. These waypoints can also be calculated by the system on the fly to ensure optimal contact with the ball.

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

Hand tuned trajectories were made for simple animatronics such as clapping and fist pumping. These were done through posturing the robot, saving keyframes in joint space, and interpolating through joint space at specified time intervals.

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

Currently, we are using RobocCup SPL Instance Segmentation Dataset for training our vision system [1] and our own dataset that was gathered during our last several years of competition. This data set is currently public and opensource [2].

[1] RoboCup SPL Instance Segmentation Dataset: https://www.kaggle.com/datasets/pietbroemmel/naodevils-segmentation-upper-camera

[2] RoboCup RoMeLa Dataset: https://universe.roboflow.com/edmond-nagja/robocup-romela-dataset

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

ARTEMIS uses the ZED 2 to gather stereo images which are then classified using a pre-trained ResNet50 Mask Region-based Convolutional Neural Network (RCNN) model from pytorch [1]. The model marks bounding boxes of the ball position, players, field lines, and goal posts. The distance to the objects and their relative position to the robot are then determined using a combination of the head transformation and size of the objects. In addition, the ZED camera library [2] is currently being investigated as it is a manufacturer provided vision library capable of returning accurate bounding box and depth measurements from the stereo images.

[1] PyTorch Mask RCNN ResNet50 webpage. https://pytorch.org/vision/main/models/mask_rcnn.html
[2] Zed camera library. https://www.stereolabs.com/developers/release/

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?

Online state estimation is performed using the Contact-aided Invariant Extended Kalman Filter (InEKF) outlined in Hartely et al. [1]. This filter is also used to localize the robot position through passing the locations of soccer field landmarks from vision to correct the global state. From this sensor fusion, the robot position relative to a marked field corner can be determined that is consistent with the same state estimation scheme used for control.

[1] R. Hartley, M. G. Jadidi, J. W. Grizzle, and R. M. Eustice, “Contact-aided invariant extended Kalman filtering for legged robot state estimation,” in Robotics: Science and Systems XIV, Pittsburgh, Pennsylvania, USA, June 2018.

How is your robot localizing?

Online state estimation is performed using the Contact-aided Invariant Extended Kalman Filter (InEKF) outlined in Hartely et al. [1]. This filter is also used to localize the robot position through passing the locations of soccer field landmarks from vision to correct the global state. From this sensor fusion, the robot position relative to a marked field corner can be determined that is consistent with the same state estimation scheme used for control.

[1] R. Hartley, M. G. Jadidi, J. W. Grizzle, and R. M. Eustice, “Contact-aided invariant extended Kalman filtering for legged robot state estimation,” in Robotics: Science and Systems XIV, Pittsburgh, Pennsylvania, USA, June 2018.

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)?

NAV 2 underwent testing but faced challenges in seamless integration with the existing code stack. Consequently, a new path planning algorithm was devised, incorporating inputs such as game controller data, robot state, relative ball position, etc. to generate a path to the desired position and pose. The path planner is designed to react dynamically, triggering replanning if the ball position undergoes rapid changes beyond

a specified threshold. Additionally, in the presence of obstacles detected between the robot and its target position, an optimization algorithm was introduced to generate a path that avoids obstacles. A mid-level planner was also developed to receive a desired path and output the corresponding velocity trajectory for effective robot navigation.

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

A behavior tree from the previous year has been implemented with actions dependent upon the location of the players and ball [1]. For example, an “attack” state will be sent if the team’s robots are closer to the ball than the opponents and a “defend” state when the opponents are closer than the team. Reinforcement learning (RL) is currently being investigated with a rudimentary ruleset in a simulated environment. The RL approach currently uses Temporal Difference (TD) Learning to develop an action value function over randomized actions to develop a policy for any given state [2]. It is planned to investigate RL as a form of higher level planning in order to reduce the need to manually maintain the high number of corner cases required for soccer decisions.

[1] M. Ahn, T. Zhu, and D. Hong. “RoMeLa Extended Abstract for Robocup 2022 Humanoid League”. In: Humanoid League Team Descriptions, RoboCup 2022, Bangkok (June 2022)

[2] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

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

Currently, the only planned active vision is to visual servo around the ball to keep it within a centralized area of the camera frame. Since the body centric position of the ball is important for maintaining control of the flow of the game, that is the main concern at this point. Extra movement of the camera is currently avoided as it degrades performance of this main task.

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

There are plans to use a particle filter to keep track of the ball position and player positions in robot centric coordinates to ensure consistent tracking. This should help

prevent discontinuous signals in the case of occlusion and false detection from disrupting accurate behavior of the robot. An approach similar to [1] is being considered.

[1] T. Beppu, S. Aoki and T. Horiuchi, "A Shared Multi-particle Filter for Ball Position Estimation in RoboCup Small Size League Soccer Games," 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), Bragança, Portugal, 2016, pp. 311-316, doi: 10.1109/ICARSC.2016.24.

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)?

For this year's competition, the strategy will mainly be focused on multi-player cooperative soccer behaviors as the basic platform is stable from last year. Intra-robot strategic planning will be pursued.

Please list contributions your team has made to RoboCup

RoMeLa has had a large involvement in past RoboCups, specifically in introducing new hardware platforms for the humanoid leagues with Team DARwIn, Team CHARLI, and Team THORwIn. In RoboCup 2011, 2012, and 2013, Team DARwIn won the KidSize league while in 2011 and 2012, Team CHARLI won the AdultSize competition. Following that success, Team THORwIn won the AdultSize league in RoboCup 2014 and 2015. Professor Dennis Hong leads Team RoMeLa and is the creator of DARwIn-OP, an open-source humanoid platform that inspired the design of many of the KidSize platforms in the competition. It is our hope that Team RoMeLa can once again share and inspire the next generation of platforms through our quasi-direct drive humanoid, ARTEMIS.

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

Several papers have been written on the localization and trajectory planning submodules and one covering the full system for the RoboCup Symposium [1], [2], [3].

[1] Hou, R., Fernandez, G., Hong, D.: Fast and robust localization for humanoid soccer

robot via iterative landmark matching and pose estimation. In: 2025 IEEE International Conference on Robotics and Automation. (Under Review)
[2] Hou, R., Fernandez, G., Zhu, M., Hong, D.: Model predictive control with visibility graphs for humanoid path planning and tracking against adversarial opponents. In: 2025 IEEE International Conference on Robotics and Automation. (Under Review)
[3] Fernandez, G., Hou, R., Togashi, C., Xu, A., Hong, D.: Clap: Clustering to localize across n possibilities, a simple, robust geometric approach in the presence of symmetries. In: 2025 IEEE International Conference on Robotics and Automation. (Under Review)

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

The hardware design was performed completely by RoMeLa down the actuator level. The software system currently uses the following 3rd party libraries as listed in previous sections:

- ROS2 (Foxy)
- Pytorch
- ZED Drivers and Library
- NAV2
- Microstrain SDK
- InEKF
- Gazebo simulator
- Pinocchio c++ dynamic library
- Solidworks to URDF Exporter
- PyTrees

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

ARTEMIS currently runs on Ubuntu 20.04 with the ROS 2 Foxy stack. Posix shared memory has been implemented using the Boost C++ Library [1] for interprocess data sharing for time critical tasks such as joint torque commands and state estimation.

[1] Boost C++ Library webpage. https://www.boost.org/

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.