TurtleRabbit 2025 SSL Team Description Paper

Linh Trinh[1]*, Alif Anzuman[1], Prasiddha Dhakal[1], Aubrey Gamboa[1], Lisa Graf[2], Darpan Gurung[1], Tharunimm Jamal[1], Austin Mckinder[1], Adharshni Mohanbabu[1], Jason Ng[1], Ali Essam Oun[1], Tilman Rosenlicht[2], Wing Lam Tsang[1], Thant Synn Win[1], and Oliver Obst[1]

[1] Cognitive robotics lab, Western Sydney University, Penrith NSW 2751, Australia
[2] Neurorobotics lab, University of Freiburg, Germany
Corresponding author: `o.obst@westernsydney.edu.au`
https://www.turtlerabbit.org

Abstract. This team description paper describes the design and development of the TurtleRabbit 2025 RoboCup SSL team, building on our 2024 experience. We introduce a new chip kicker design and enhancements to our straight kicker to improve power and consistency, while following our cost-effective, open-source approach. Our hardware setup employs off-the-shelf components, including BLDC drone motors and off-the-shelf controllers, all integrated within a chassis milled from aluminium that also acts as a heat sink. We are working on improvements to the dribbler system and incorporated an on-board camera for ball detection. On the software side, we explore multi-agent reinforcement learning and implement a Voronoi-based path planning strategy to navigate dynamic environments more efficiently.

Keywords: RoboCup SSL · Chip Kicker · Reinforcement Learning · Voronoi Path Planning · Open-Source Hardware

1 Introduction

For the TurtleRabbit 2025 RoboCup SSL team, we have addressed some of the challenges of our 2024 team [11]. Our overall strategy remains to create a simple and cost-effective design by utilising readily available electronic components and accessible manufacturing techniques, including 3D printing and CNC milling.

For 2025, we introduce a new chip kicker design that our robots previously lacked. Together with this new development, we also have improved performance and consistency of our straight kicker. In addition, our initial explorations into reinforcement learning are beginning to inform robot control and strategic decision-making. The following sections outline our updated hardware components, the integration of new control mechanisms, and the evolution of our software architecture. As in the previous year, our hardware design is open source.

* Authorship: team lead first, contributors in alphabetical order, academic lead last.

2 Hardware recap and revision

Our overall hardware setup has been described in detail in [11]. We provide a brief summary below: the TurtleRabbit robots use Tarot 4008 BLDC drone motors inside omni-wheels with a 3D-printed core (PETG) and aluminium covers (6061-T6) on either side. Each of the 30 subwheels comprises two flanged ball bearings (2 mm $\times$5 mm$\times$2.5 mm) with NBR X-rings on each bearing.

The motor controller is an MJBots moteus-r4 board [7], unchanged from last year's setup. It employs three-phase FOC-based control and can be operated via a CAN bus interface. For position feedback from the BLDC motors, we use a diametrically magnetised magnet glued to the secondary (rear) shaft of the motors. The encoder is integrated into the controller board, which requires mounting the boards vertically behind the wheels. This design choice also influences the main structure of the chassis, with the aluminium motor mounts securing the controller on the opposite side and connecting the base plate to the midlevel plate. This structure additionally functions as a heat sink for the motors.

The motor controllers are powered from a distribution board that safely pre-charges high capacitance loads (MJBots power dist r4.x), and the on-board Raspberry Pi 4b is paired with a MJBot pi3hat that provides 5 independent CAN-FD interfaces. For our kickers and the dribbler, we use an Arduino Every as a controller, with a drone ESC (20 A XRotor) for the dribbler motor. The robot is powered using a 6S1P lithium polymer battery. Our 2024 robots used an off-the-shelf push-pull solenoid powered using a boost converter as a straight kick device. While functional, this kicker is not particularly strong, and one of the components we investigated for improvement in our 2025 version.

A more detailed table with components and approximate prices can be found in our 2024 team description [11]. In (US) dollar terms, overall costs to build our robot have decreased from last year, e.g., the moteus-r4 board is now listed for 59 USD (down from 75 USD in 2024), making the build more affordable.

3 Hardware development

The main development in our hardware is a combined kicker / chip-kicker. We are also still working on improvements to our dribbler, as well as an on-board camera for ball detection.

To utilise the space between the front wheels of the robot and to build a better kicker, including a chip-kicker, we followed the approach of RoboTeam Twente [5] and designed custom solenoids.

3.1 Chip-Kicker Design

Previously, our robots could only kick the ball in a straight line without any acceleration in the z-direction. This year we have added a chip kicker that can accelerate the ball in the z-direction, allowing it to kick over obstacles. Our design

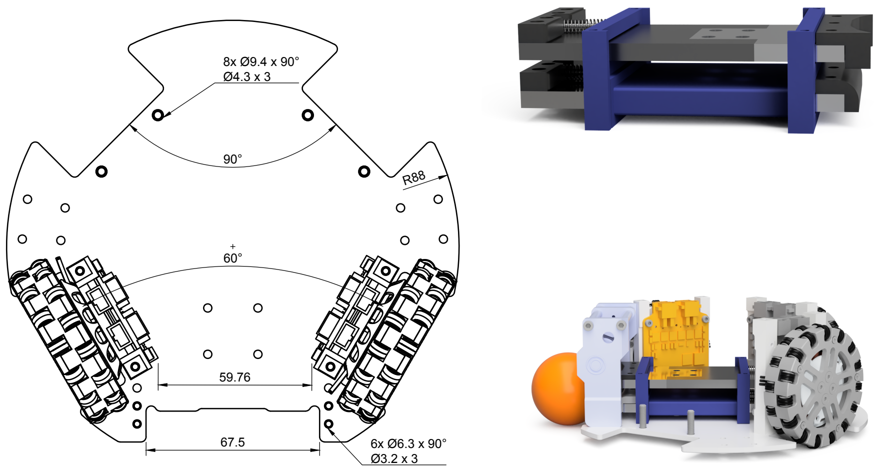

Fig. 1. Left: Base plate with the front wheel groups including motor mount and controllers. The front wheel angle is 60 degrees to increase the space for kicker and dribbler. Right top: A render of the two level kicker device, shown without the wire windings, and the straight (top) kicker without the frame for the windings. Right bottom: Render of the base plate with the new kicker, with the left front wheel group removed.

combines both types of kicker in a single housing. We have also improved the power behind straight kicks, which previously tended to decrease over time.

Our requirements were as follows: The chip kicker should produce enough power to kick over at least one robot; it should be controllable by our on-board Arduino and work with the available power source (6S LiPo, nominal 22.2 V). As in the RoboTeam Twente [5] approach, we are using two rectangular solenoids stacked on top of each other.

Our aim was to fit the new kicker into our existing design, with the base plate and wheel setup providing the main constraints (see Fig. 1). Our prototype was constructed using mild steel as the kicker core, with 3D printing used to create a frame for the enamelled wire windings (0.5 mm wire). The steel was cut using waterjetting, while the front extension of the plunger was milled from 6061-T6 aluminium.

The height and distance of the chip kick were measured using a slow-motion camera and a reference grid in the background. In the course of our experiments, we systematically varied the voltage within the range of 120–220 V, using a voltage booster integrated into the kicker circuit to achieve this adjustment. The results of the experiments are shown in Figure 2. The results suggest that there may be a linear relationship between voltage and kick distance/height. In the experiments, we were able to kick the ball over up to six robots placed directly next to each other in a line. So far, the kicker has only been tested attached to

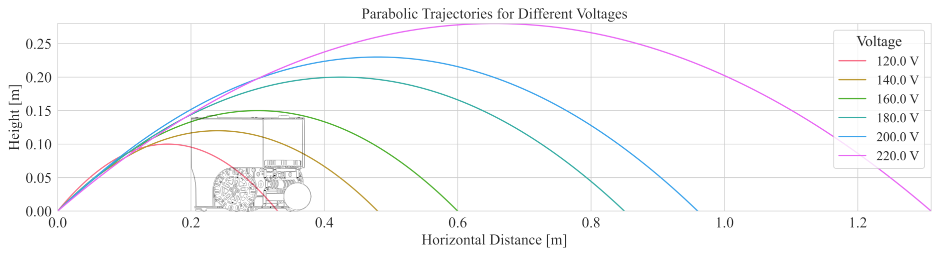

Fig. 2. Experimental results showing the relationship between the voltage and the kick distance and height.

a stationary base plate. The next step will be to finalise the design and fully integrate it into the robots.

3.2 Dribbler and ball detection

Our current dribbler is driven by a small BLDC motor (for RC cars), with an ESC connected to the Arduino. We are currently experimenting with different motors to determine a good speed / torque trade-off. With the current motor, we achieve a high speed but also observe high wear on the (3D printed) nylon gears (Nylon 6, Taulman Alloy 910). The dribbler motor with nylon gearing drives a roller, 3D printed from TPU. We are currently testing different shapes and levels of softness for the roller design.

To complement the dribbler system and improve overall ball interaction, we have implemented an on-board camera for ball detection. The ball detection system implemented with an onboard camera yields real-time position feedback via OpenCV.

4 Software development

One main focus of our software development is to create an environment for reinforcement learning experiments. Other aspects of our software development are to address issues of our robots connecting and executing our strategy, as well as components like ball detection and improvements to the path planning.

4.1 Reinforcement Learning

Multi-agent reinforcement learning (MARL) continues to be a dynamic area of research, and the RoboCup Small Size League (SSL) provides an excellent platform for experimentation. The SSL environment is well-suited for studying MARL techniques because it allows unrestricted communication between all robots, with each robot having access to the positions of all other robots, its own position, and the position of the ball throughout the game. This setup facilitates centralised training and execution, making it an ideal testbed for developing and

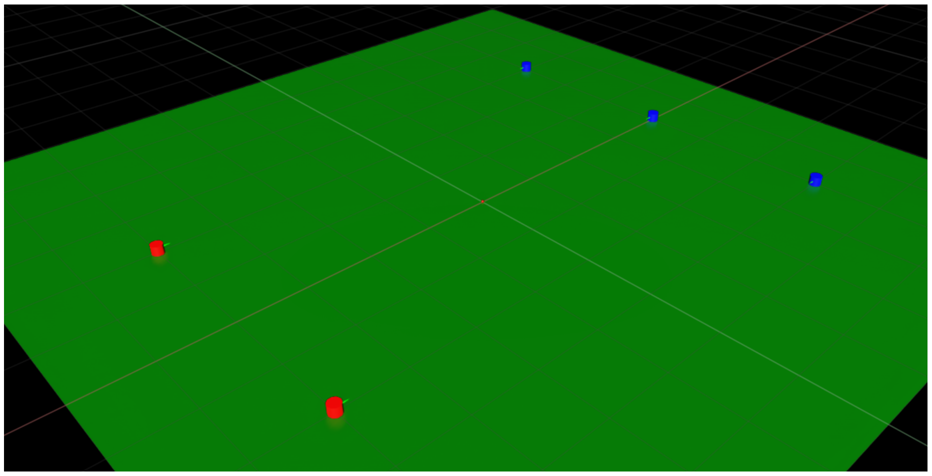

Fig. 3. A simple Isaac Sim environment of the field with multiple robots and a ball.

testing MARL algorithms. We have conducted some initial experiments using NVIDIA Isaac Sim [6] as well as MuJoCo [10]. Isaac Sim gives us the ability to start with a high-level representation of the robot in a powerful physics engine and later increase the robot's complexity to reduce the sim-to-real gap. The robots currently used in the simulation consist of a cylinder with the robot's dimensions and all the essential actions required in a real match (see Figure 3). They can move around freely in all directions, turn on the spot, dribble the ball, and kick the ball either straight or over other robots.

We decided to use Proximal Policy Optimisation (PPO) [8] as a starting point, since it is already a widely used reinforcement learning algorithm in MARL for both cooperative and competitive scenarios. We began with rather simple tasks and then progressed to more complex problems. Initially, we trained a single robot to perform tasks such as moving to specific positions in the environment, approaching the ball, chasing a moving ball, and kicking the ball into the goal. Some of our next goals include introducing more robots into the environment and training them in an adversarial setting. Our ultimate aim is to train the behaviour for an entire team of robots using reinforcement learning and then apply it to the physical robots.

To break down this task of training a whole team strategy into more manageable steps, a simpler scenario is to implement the 3-vs-2 keepaway task that has been successfully used for reinforcement learning in the RoboCup (2D) soccer simulation league [9].

While learning an optimal overall strategy would be the most "automatic" approach, the goal of an intermediate solution is to give us useful behaviours as components that still allow us manually adjust the team strategy. A reinforcement learning approach to legged (humanoid) robots has been described in [2]. The similarities in the approaches are a multi-step approach towards learning the

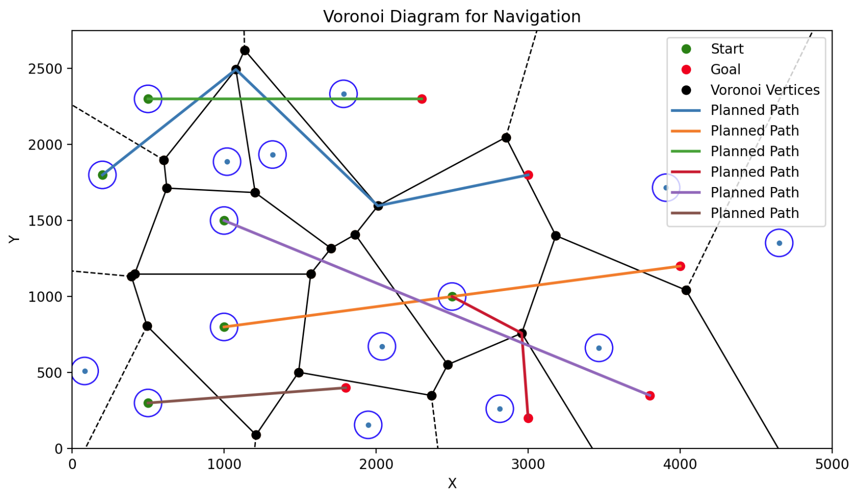

Fig. 4. A Voronoi diagram computed by our planner. The Voronoi edges are used only as part of the path planning if obstacles are present on the initial part of the direct line from the current position to the target.

overall behaviour. Importantly, a key difference is that the humanoid robots in [2] use a single robot on each side. While the individual robots are more difficult to control than the SSL robots, playing in the SSL requires the coordination of multiple robots and complying with the rules of the game.

4.2 Path planning improvements

For our 2024 team, we employed a Probabilistic Roadmap (PRM) approach [3] for path planning. One of the main advantages of this approach is its simplicity: PRM uses a set of n random samples distributed across the soccer field to identify collision-free paths between those points. However, to consistently achieve short paths, many samples are required, which in turn increases processing times (see also [4] for alternatives to random sampling). In dynamic environments requiring frequent replanning, we found this approach to be too inefficient. We modified our planning to adapt to a Voronoi-based approach, where the robots on the field are used as the centroids of the Voronoi cells. The edges of the diagram describe paths that are furthest away from obstacles. We use the Voronoi paths only if the initial portion of the intended trajectory is obstructed, and employ a direct connection otherwise (even if an obstacle appears later along the route). This results in more efficient planning and shorter paths in a highly dynamic environment. The fundamentals of this idea using a generalised Voronoi diagram can be found in [1].

Figure 4 shows a screenshot of the paths planned for the robots to go to different targets. The planning procedure is implemented in python (using

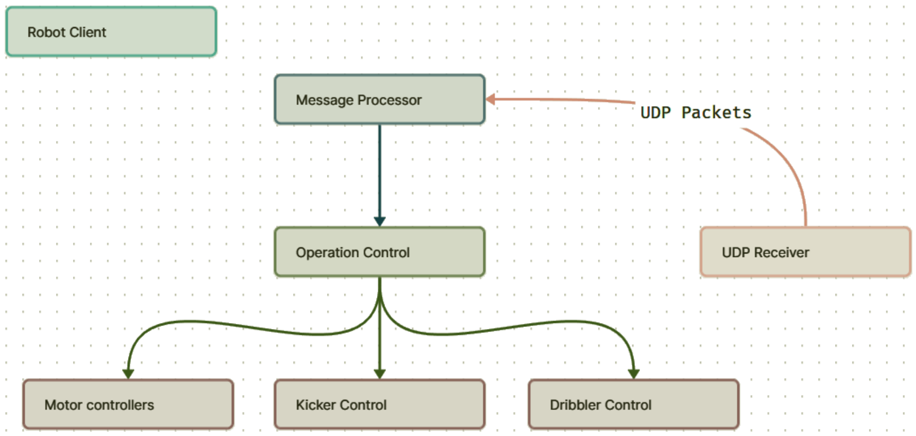

Fig. 5. Robot-client Software Multiprocessing Flow Diagram

`scipy.spatial` for the Voronoi calculations), and takes 5–7 ms per step for the team.

4.3 Software changes and improvements

Our robot controlling software (robot-client) has been developed in Python and improved over the past year. The robot-client is responsible for responsive behaviour of our robots. We use Python multiprocessing to run multiple processes in parallel at the same time. An overview of the different operations within the robot has been shown in Fig. 5.

We make use of multiprocessing to integrate our motor controllers operating in asynchronous mode, receiving data packages via UDP, and making adjustments to the motor velocities or executing other commands. To better cope with the multiple operations, the Raspberry Pi4 had to be equipped with a better heatsink.

Networking One of the issues we faced participating in the 2024 competition was related to the networking. We have addressed some of these issues by implementing a more reliable, scripted control, setting initial static IP addresses, and working on enhancing the backlog and troubleshooting functions. If the robot-client does not receive any UDP messages over a short period of time (0.5 – 1 seconds), the robot stops and cancels its current operation.

5 Conclusion

We presented the continuing development of the TurtleRabbit 2025 RoboCup SSL team. We introduced a chip kicker with enhanced performance compared to our last year's approach, and also ongoing work around dribbler and ball detection. On the software side, our experiments in reinforcement learning and the implementation of a Voronoi-based path planner aim to result in a robust strategy for our team. We are planning to test our approach during the 2025 Japan Open (funding permitting).

Acknowledgments. We received partial support from the Office of the NSW Chief Scientist & Engineer in 2024. We also received partial support from Westpac Banking Corporation to travel to RoboCup 2024 in Eindhoven, NL.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press Academic, Cambridge (May 2005)
2. Haarnoja, T., Moran, B., Lever, G., Huang, S.H., Tirumala, D., Humplik, J., Wulfmeier, M., Tunyasuvunakool, S., Siegel, N.Y., Hafner, R., Bloesch, M., Hartikainen, K., Byravan, A., Hasenclever, L., Tassa, Y., Sadeghi, F., Batchelor, N., Casarini, F., Saliceti, S., Game, C., Sreendra, N., Patel, K., Gwira, M., Huber, A., Hurley, N., Nori, F., Hadsell, R., Heess, N.: Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning. Science Robotics **9**(89), eadi8022 (Apr 2024). https://doi.org/10.1126/scirobotics.adi8022
3. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation **12**(4), 566–580 (Aug 1996). https://doi.org/10.1109/70.508439
4. LaValle, S.M.: Planning Algorithms. Cambridge University Press, 1 edn. (May 2006). https://doi.org/10.1017/CBO9780511546877
5. Monat, C., Dankers, E., Skurule, K., Steenmeijer, L., Sijtsma, S., Diamantopoulos, S., Aggarwal, R., Smit, T., van Harten, A.: RoboTeam Twente Extended Team Description Paper for RoboCup 2022. Tech. rep. (2022)
6. NVIDIA Corporation: Isaac Sim. https://developer.nvidia.com/isaac-sim (2025), accessed: 10 February 2025
7. Pieper, J.: Moteus controller reference manual (2024), https://github.com/mjbots/moteus/blob/main/docs/reference.md
8. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (Aug 2017). https://doi.org/10.48550/arXiv.1707.06347
9. Stone, P., Sutton, R.S., Kuhlmann, G.: Reinforcement Learning for RoboCup Soccer Keepaway. Adaptive Behavior **13**(3), 165–188 (Sep 2005). https://doi.org/10.1177/105971230501300301
10. Todorov, E., Erez, T., Tassa, Y.: MuJoCo: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5026–5033 (Oct 2012). https://doi.org/10.1109/IROS.2012.6386109
11. Trinh, L., Anzuman, A., Batkhuu, E., Chan, D., Graf, L., Gurung, D., Jamal, T., Namgyal, J., Ng, J., Tsang, W.L., Wang, X.R., Yilmaz, E., Obst, O.: TurtleRabbit 2024 SSL Team Description Paper (Feb 2024). https://doi.org/10.48550/arXiv.2402.08205