Article

Outplaying elite table tennis players with an autonomous robot

https://doi.org/10.1038/s41586-026-10338-5

Received: 5 September 2025

Accepted: 27 February 2026

Published online: 22 April 2026

Open access

Check for updates

Peter Dürr[1]✉, Mireille El Gheche[1], Guilherme Jorge Maeda[2], Nobuhiko Mukai[2], Naoya Takahashi[1], Stefan Heusser[1], Hamdi Sahloul[2], Yamen Saraiji[2], Pavel Adodin[2], Yin Bi[1], Sam Blakeman[1], Christian Conti[2], Dunai Fuentes Hitos[1], Yunpu Hu[2], Farshad Khadivar[1], Raphaela Kreiser[1], Luz Martinez[2], Fabian Schilling[1], Ricardo Tapiador Morales[1], Guillem Torrente[2], Mario Ynocente Castro[2], Lison Abecassis[1], Alberto Giammarino[2], Yu-Ting Huang[1], Yannik Nagel[1], Andrea Scotti[1], Alexander Sigrist[1], Tiago Silva[2], Etienne Walther[1], Jengyan Wong[2], Bilan Yang[2], Asude Aydin[1], Divij Grover[1], Apurv Saha[2], Valentina Cavinato[3], Takekazu Kakinuma[4], Taishi Kunori[5], Valentin Monferrato[1], Stefan Richter[6], Stefanos Charalambous[1], Simon Guist[1], Mads Alber Kuhlmann-Jorgensen[1], Lorenzo Miele[1], Agis Politis[3], Mattia Scardecchia[1], Hiroaki Kitano[2], Peter R. Wurman[7], Peter Stone[7] & Michael Spranger[2]

Artificial intelligence (AI) systems now challenge or surpass human experts in many computer games[1,2]. Physical and real-time sports such as table tennis, however, remain a major open challenge because of their requirements for fast, precise and adversarial interactions near obstacles and at the edge of human reaction time[3]. Here we present Ace, to our knowledge the first real-world autonomous system competitive with elite human table tennis players. Ace addresses the challenges of physical real-time interaction through a new, high-speed perception system using event-based vision sensors[4], and a new control system based on model-free reinforcement learning, as well as state-of-the-art high-speed robot hardware. Evaluated in matches against elite and professional players under official competition rules, Ace achieved several victories and demonstrated consistent returns of high-speed, high-spin shots. These results highlight the potential of physical AI agents to perform complex, real-time interactive tasks, suggesting broader applications in domains requiring fast, precise human–robot interaction.

High-speed control of agile robots in dynamic environments is fundamental for the execution of many real-time tasks featuring interactions with humans, and one of the key challenges of many real-world applications, including, for example, in manufacturing or service robotics. Recent advances in deep reinforcement learning (RL) enable high-speed robot control and achieve human-champion-level performance in racing tasks[2,5]. However, these results have so far been demonstrated only in simulated environments[2], without the need to estimate state from noisy sensors, or static real-world settings that are mostly unaffected by the actions of human opponents[5].

Table tennis is a sport that requires fast reaction times as the ball velocity can exceed 20 m s^{-1} in high-level games, and the time between shots is often less than 0.5 s (ref. 6). The spin, that is, the angular velocity of the ball, can reach 1,000 rad s^{-1}, which greatly affects the ball trajectory and its response when bouncing on the table and racket. Spin is used to make shots harder to return or to gain a tactical advantage. Responding effectively requires expert players to master a range of skills for tracking, reacting to, and generating high-speed, high-spin shots. Since 1983, various table tennis robots have tackled this challenge in simplified settings[7], using ball launchers[8–17], reduced court coverage[8–13,15–20] and omitting either robot[3,8–18,20–22] or human serves[8–13,15–17]. Crucially, the role of spin has often been ignored[8,9,11,12,15,17,18,21,22], although it is an important component of competitive human play[23].

From a control perspective, previous work typically relies on assumptions such as heuristic hitting points[9,10,16,17,20,21,24], ball trajectory predictions[8–10,13,14,17,19–22], flight durations[13,16,25], human demonstrations for hitting motions[12,18] and explicit solutions of the racket contact state[9–11,13,14,16,17,19–22]. Other approaches[3,15,26] forgo these assumptions by using learning-based methods to train policies that directly control the robot joints over time based on incoming observations. The actions produced by these policies can be interpreted as low-level commands (for example, air pressure[15]) or explicit joint position or velocity setpoints[3,26].

In this paper, we introduce Ace, to our knowledge the first real-world table tennis AI agent competitive with human athletes. Ace is equipped with a new perception system using event-based vision sensors as well as a control system based on policies learnt using deep RL. In contrast to earlier approaches using RL for robot table tennis, the control policies used by Ace are learnt using an asymmetric actor–critic architecture[27–29], and the actions produced by the policies exist in an abstract space that is then mapped to a hard constraint for a convex optimization problem. This setup allows learning of collision-free, agile motions, addressing the full challenge of human-competitive robot table tennis.

[1]Sony AI, Zürich, Switzerland. [2]Sony AI, Tokyo, Japan. [3]Sony Advanced Visual Sensing, Zürich, Switzerland. [4]Sony Global Manufacturing & Operations, Tokyo, Japan. [5]Sony Techno Create, Tokyo, Japan. [6]Richter Optimization, Vienna, Austria. [7]Sony AI, New York, NY, USA. ✉e-mail: peter.duerr@sony.com

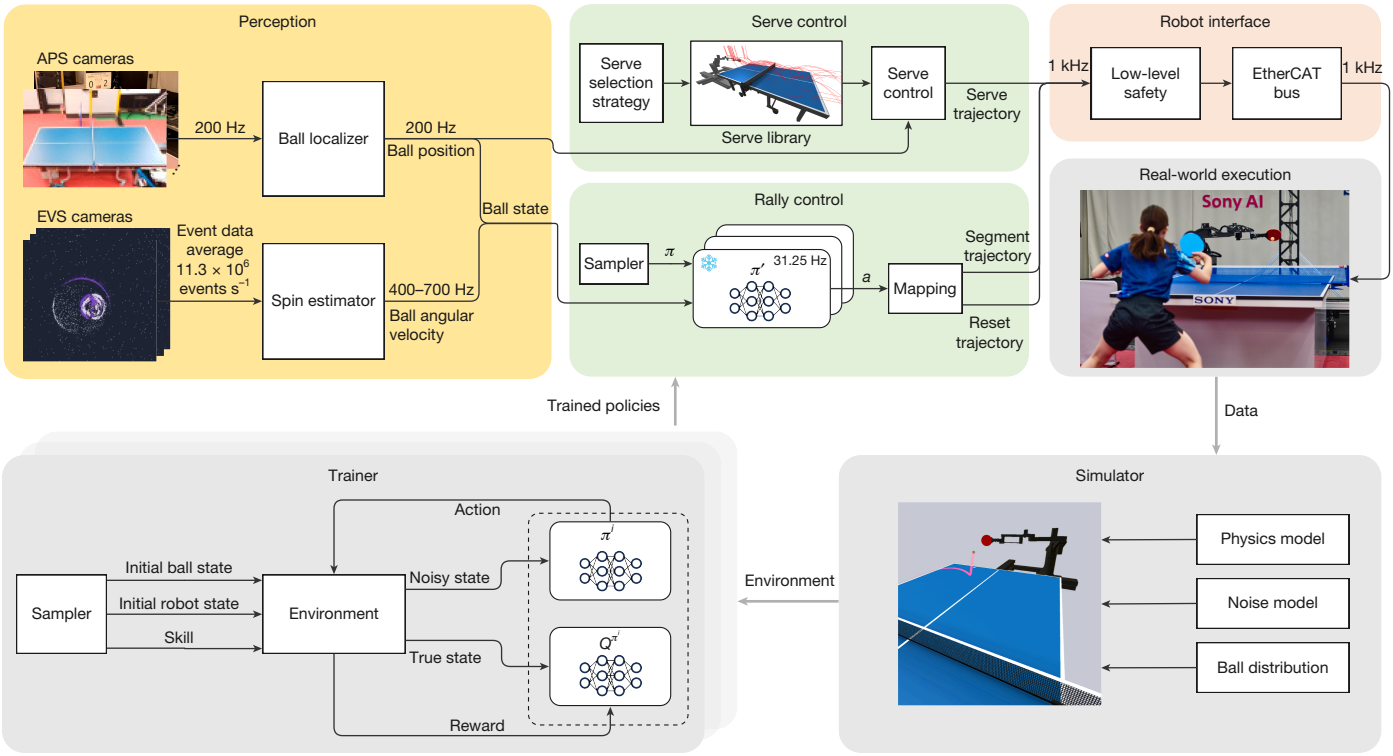


Fig. 1 | Ace integrates perception, control and robot hardware to play table tennis. The perception system uses a combination of conventional APS cameras for ball triangulation and EVS cameras for ball angular velocity estimation to infer the current ball state at high frequencies. The ball state is then provided to two different control components, depending on whether Ace is serving or in a rally. When serving, the robot performs a single-arm serve from a library of serve motions that were found using a genetic algorithm. During the rally, a fixed deep RL policy (π') is queried at 31.25 Hz using the robot joint states and the ball position and spin histories. The policy is sampled during the match from a bank of policies trained to perform different skills. The actions (a) produced by the policy are mapped to a 32-ms segment trajectory, and a corresponding reset trajectory is calculated. If the robot has yet to hit the ball and no collisions are predicted, then the segment trajectory is executed by the robot interface; otherwise, a reset trajectory is executed. The training of all policies is performed entirely in simulation with custom physics models, noise models and data-driven distributions of the initial ball state. Training is performed asynchronously with multiple instances of the training environment. To aid in the learning process, the critic (Q^{π^i}) is provided with the true ball state, whereas the policy (π^i) is given a history of noisy sensor measurements.

To assess its performance, we evaluate Ace in matches against five elite players (defined here as competitive athletes with more than 10 years of intensive training), and two professional players (defined here as athletes who compete in officially recognized professional leagues), following the rules of the International Table Tennis Federation (ITTF; https://www.ittf.com). Ace achieved three victories in five matches against elite players, along with competitive performances in the remaining matches. These results demonstrate the potential of physical AI agents to outperform human experts in interactive, real-time tasks.

Approach

Ace comprises three main components: perception, control and robot hardware (Fig. 1).

Perception

Given the high linear and angular velocity of the ball during professional-level table tennis, accurately measuring the ball state at high frequency and low latency is crucial. To address this need, Ace uses nine active pixel sensor (APS) cameras with Sony IMX273 sensors placed outside of the court to cover the whole playing area (Fig. 2a,b). This system enables Ace to locate the ball in three-dimensional (3D) space at 200 Hz with 3.0 mm error and 10.2 ms latency on an average. To measure the angular velocity of the ball, Ace also uses three gaze control systems (GCSs)[30] (Fig. 2d). Each GCS comprises (1) an event-based vision sensor (EVS) camera with Sony IMX636 sensor; (2) pan and tilt mirrors to track the ball[31,32]; and (3) a telephoto tunable lens to keep the ball in focus. This measurement is obtained by two methods asynchronously: (1) by feeding accumulated event data to a convolutional neural network (CNN)[33] for low latency and (2) by performing contrast maximization (CMax)[34] for high accuracy at the cost of increased latency. Based on the uncertainties computed by each method, the angular velocity estimates are filtered and passed to the control system at a variable frequency of approximately 400–700 Hz.

In combination, this setup overcomes the challenges present in previous EVS-based spin measurement systems[35].

The average measurement error is estimated to be 24.8 rad s^{-1}.

Control

The control component of Ace includes rally and serve. During the rally phase, the control of Ace is guided by a deep RL policy that maps ball and robot states to actions every 32 ms (corresponding to a frequency of 31.25 Hz). The policy is trained entirely on single shots in simulation using the Soft Actor-Critic (SAC) algorithm[36] to return the ball with a desired skill. A skill is characterized by the state of the ball on landing on the opponent's side of the table after a successful return, for example, high top-spin. Multiple policies are trained with a variety of desired skills, and a policy sampler is used to sample them during the match. During training, the critic receives the ground-truth ball state from the simulator while the policy receives a history of the last N noisy sensor measurements. This setup ensures that the policy transfers to the real world while the critic provides an accurate learning signal during training[27–29].

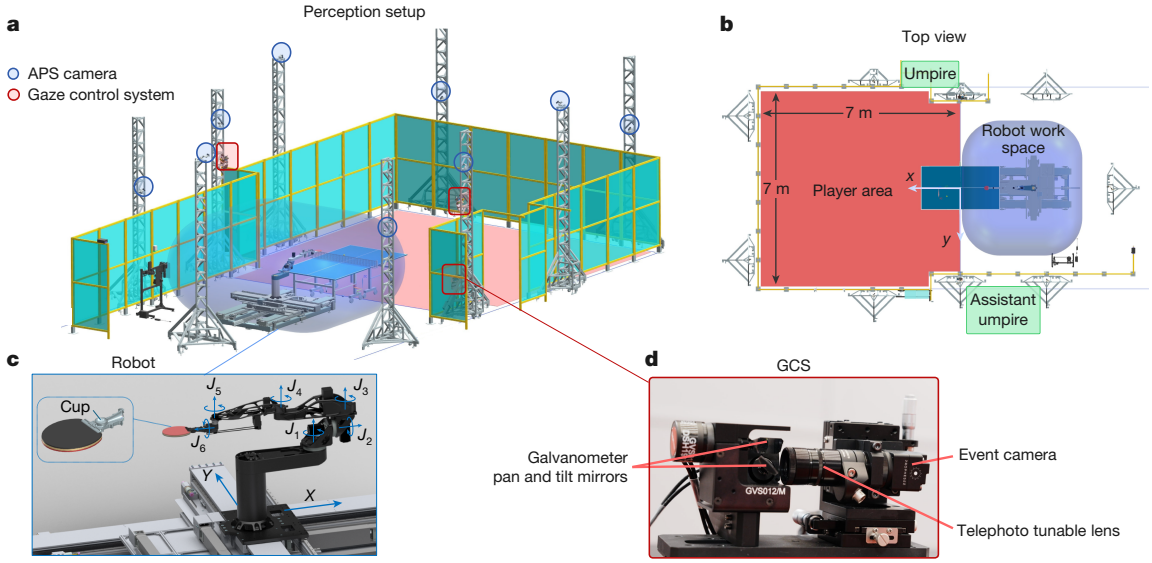


Fig. 2 | The Ace system setup. a, Nine APS cameras and three gaze control systems (GCSs) are installed on towers to cover the whole Olympic-sized court from outside the play area. **b**, The player is allowed to play in half of the court while instructed not to enter the robot side of the court for safety reasons. Two licensed umpires judge the match from both sides of the table. **c**, The robot hardware includes two prismatic and six revolute joints, and an end effector equipped with a racket and a cup to hold the ball, facilitating one-armed serves. **d**, The GCS comprises an event camera, a telephoto tunable lens and galvanometer pan and tilt mirrors.

The produced actions are mapped to feasible joint positions and velocities, which are used as a terminal constraint (that is, a waypoint 32 ms into the future) for an optimization problem that produces a continuous-time trajectory segment sampled at 1 kHz. In parallel, a near-time-optimal model predictive control (MPC) motion planner calculates a 'reset trajectory' from the same terminal constraint to a desired stationary reset position. This desired stationary reset position is either fixed or learned from data so as to maximize the dexterity of the robot (that is, its ability to initiate movements in different directions) for the next shot. The robot interface checks whether executing the segment and subsequent reset trajectory would result in a collision with either the robot or the table. If a collision were to occur, then the reset trajectory associated with the previous segment is executed, which is guaranteed to be safe[37]. This process is repeated until the robot hits the ball or the ball goes out of play. If the robot hits the ball, then the most recent reset trajectory is executed until the player hits the ball, at which point the process starts again by sampling a skill and querying the subsequent policy.

Serves are handled through a specialized process involving three components. First, a tossing motion extracted from human demonstrations is used, respecting official rules (straight tosses, minimum height 16 cm), to set robot and ball trajectories before the hit. Second, striking trajectories are generated in simulation, offline, using a genetic algorithm[38] that adjusts racket poses and velocities to optimize serve characteristics, such as ball landing position, velocity and angular velocity. Third, serves are evaluated on the real robot by expert players, and those serves considered sufficiently challenging are stored in a library for use in matches. Serves are selected from the library based on either their dissimilarity from recent past serves or rally-winning probability estimated from previous experiments.

Robot hardware

To achieve the workspace and agility required for professional-level table tennis, we developed a custom robot platform featuring eight degrees of freedom (two prismatic and six revolute joints; Fig. 2c), which was determined by the minimum number necessary to execute competitive shots: three for the position of the racket, two for its orientation, and three for the speed vector and magnitude of the shot. We set the maximum velocity of the end-effector of our robot based on the initial velocity of a professional-level drive shot (20 m s^{-1}) and the average time it takes to play a single shot back and forth (0.8 s). Its workspace is based on the area used by professional players to perform most shots (3.6 m × 3.6 m).

Topology optimization is used to reduce robot link mass without compromising stiffness. The resulting geometries were then additively manufactured in Scalmalloy. The end effector is composed of a racket (Butterfly Dignics 05 2.1 mm rubbers attached to a VICTAS ZX-GEAR OUT blade modified to be mounted on the robot) and a small cup to hold the ball during serve.

Achieving simple, fast and accurate visual tracking control at the hardware level is essential for reducing the gap between simulation and reality. Therefore, we synchronized all actuators at 1 ms intervals and shared a clock signal with the perception system to obtain hardware-synchronized timing across the entire system. The low-level control is based on position feedback and yields a system that behaves approximately linearly with a position tracking delay under 5 ms, even at maximum velocity.

Evaluation

In April 2025, we evaluated Ace in a series of matches against elite and professional players, all of whom faced the robot for the first time. No specific data about these players was used to prepare Ace before these matches. Ace played best-of-three games against five elite players (three female and two male), each with more than 10 years of active table tennis experience, including participation in national or regional championships, and an average of 20 h weekly training, practising five to six times per week in the past year. Ace also played best-of-five games against two professional players (Pro 1: Minami Ando and Pro 2: Kakeru Sone), both active in the Japanese professional league (T.League).

Unlike previous work in competitive robot table tennis, in which modifications to the rules simplify the problem (for example, modified rackets, special rules around serve and receive, exclusion of certain shots, restricted playing area)[3], we introduced no such constraints and replicated regular competition conditions, including free choice of racket under ITTF rules (https://www.ittf.com), and an Olympic-size

a

Ace versus human player						
Elite A	Elite B	Elite C	Elite D	Elite E	Pro 1	Pro 2
Best of three					Best of five	
11–4	**11**–5	**11**–1	**11**–10	10–11	**11**–4	8–11
9–11	**11**–4	9–11	10–11	10–11	7–11	6–11
11–10		**11**–5	7–11		6–11	9–11
					1–11	

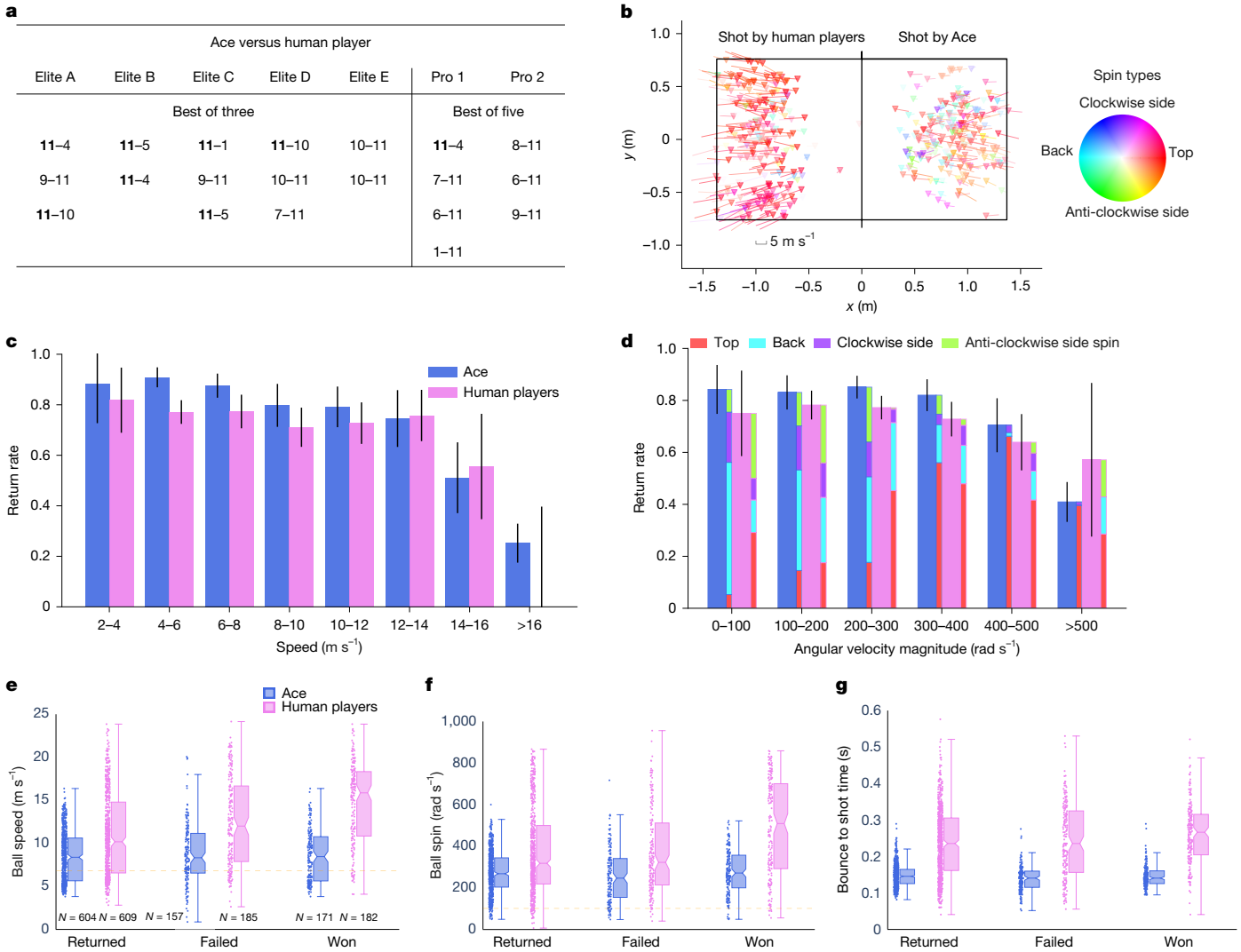


Fig. 3 | Evaluation results. a, Score of matches. **b**, The post-table-bounce ball states of winning shots. The points show the bounce position. Lines from the points represent the velocity, and the colour encodes the spin axis. **c**,**d**, Return rate of Ace and human players at different speeds (**c**) and angular velocities (**d**) at the post-racket hit of the opponent. **e**,**f**, The post-shot ball speed (**e**) and angular velocity (**f**) produced by Ace and human players. 'Returned' shots include 'Won' shots. The dashed lines represent the baseline values reported in the literature[3] corresponding to the average speed produced by the fastest policy (**e**) and the lower border in the histogram of the largest estimated angular velocity the robot can return (**f**). **g**, The time between the table bounce to the racket hit. The notches of the boxes in **e**–**g** indicate the (approximately) 95% confidence interval[43].

court (7 m × 7 m × 5 m on the player side). In all matches, we followed the rules of the ITTF with the exception of introducing the golden point rule (the first player to score 11 points wins the game) as done in the Japanese T.League. Two umpires licensed by the Japanese Table Tennis Association (JTTA) judged points and enforced compliance with the rules from both sides of the court as in T.League matches (Fig. 2b). We use official competition balls (Nittaku Nexcel 40+ 3 star), manually checked for quality as usually done in professional-level matches.

The results of the matches from April 2025 are shown in Fig. 3a. Ace won three matches out of five against elite players, totalling seven games won out of 13 games played. Ace lost the two matches against professional players, winning one game out of the seven games played. The distribution of winning shots is shown in Fig. 3b. The human players mostly won points by high-speed shots with top spin. By contrast, Ace won points with a variety of spin types. Figure 3c shows that Ace returns shots up to 14 m s^{-1} consistently (with a similar or better return rate compared with the human players), with a notable drop above 16 m s^{-1} (similar to human players). Ace also returns a wide range of spins (Fig. 3d), consistently achieving more than 75% return rate up to 450 rad s^{-1}, demonstrating its excellent ability of handling spin, and far exceeding previously reported values in competitive table tennis robots[3].

Ace achieves maximum ball velocities of 16.4 m s^{-1} (linear) and 600 rad s^{-1} (angular) (Fig. 3e,f). The maximum ball linear and angular velocities of the player returned by Ace are 19.6 m s^{-1} and 867 rad s^{-1}, respectively. The distribution of human players' 'Won' shots exhibits higher linear and angular velocity than that of 'Returned' shots, indicating that human players rely on stronger-than-average shots to win the point. By contrast, the distributions of 'Returned' and 'Won' shots produced by Ace are similar, suggesting that Ace won points through consistent returns rather than faster shots.

The notches in Fig. 3e–g allow the assessment of the significance of the differences of the medians. When the notches of two boxes do not overlap, the corresponding medians are significantly different at (approximately) the 95% confidence level. Welch's *t*-test results on the means of the returned and won shots show that the *P*-value comparing the distributions of the shots of the robot is 0.88, whereas

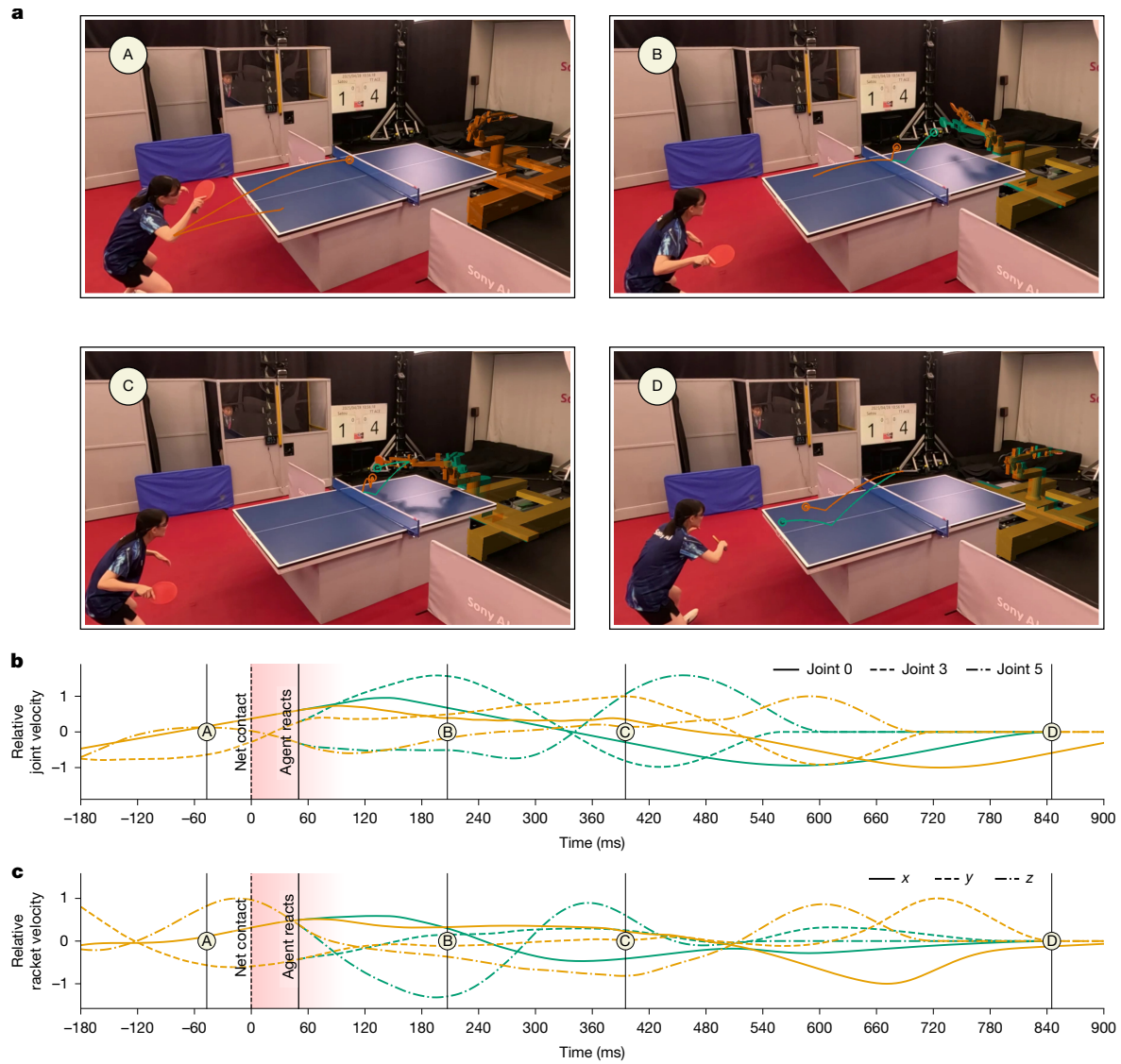


Fig. 4 | Ace demonstrates highly reactive behaviour in a net-contact scenario. Orange overlay indicates real-world trajectories, and green overlay indicates simulated counterfactual trajectories. **a**, The real ball trajectory (orange) shows the ball hitting the net at 0 ms, overlaid with a counterfactual simulated trajectory (green) in which the ball narrowly misses the net. **b**,**c**, Relative joint velocities (for three selected joints) (**b**) and linear racket velocities (**c**) are shown for both scenarios (orange and green), normalized to a maximum value of 1 for the real trajectory. Notably, 49 ms after net contact, the robot trajectories begin to diverge, successfully returning the ball in both cases. This response illustrates the ability of the system to adapt rapidly and safely to unexpected, dynamic events.

that of the shots of the human player is less than 0.001. These results further validate the analysis described above. Figure 3g shows that Ace hits the ball sooner after the table bounce than the human players. The average rally length is 5.0 shots with a standard deviation of 3.0, which is longer than a typical rally length in human games (3.9 ± 2.0 shots)[39].

Using 15 different types of serves, Ace scored a total of 16 direct points after serving, sometimes called 'aces', against the elite players, whereas the elite players collectively scored only eight. Against professional players, Ace used 13 serve types and secured four aces, compared with seven aces by the professionals.

The low-latency perception and control systems of Ace also allowed for quick reaction to unusual shots, such as balls bouncing off the net (Fig. 4). This behaviour illustrates the ability of our approach to generalize to situations that are both rare and hard to model in simulation.

Since peer review, Ace has been further improved and validated through additional experiments, and Supplementary Videos are available at https://ace.ai.sony.

Conclusion

Robot table tennis is a challenging benchmark that requires fast perception and decision-making, as it features real-world interaction, including adversarial actions at the fastest time scales humans are capable of. Ace challenges elite and professional players using unaltered, professional-level equipment and rules, demonstrating for the first time, to our knowledge, that it is possible for AI systems to outperform human athletes in interactive, physical skill-based games. The success of this benchmark, enabled by the new perception system and new learning-based control algorithm of Ace, suggests that similar techniques apply to other areas featuring fast, real-time control and human interaction, including, for example, manufacturing and service robotics. A key challenge with the zero-shot transfer of policies trained in simulation to the real world lies in modelling human behaviour in a high-dimensional physical space. Without such a model, the true objective function (that is, winning or losing) is not accessible, and a surrogate objective must be used.

For avenues of further improvement, ideas from previous robot table tennis literature that focuses on human opponent modelling[40–42] might provide a path to better understand the importance of tactics and strategy, and online learning[20,22] could allow for continuous improvement from interactions in the real world.

Nevertheless, real-world AI systems such as Ace may perhaps already change how humans interact and play games such as table tennis. Observing a shot played by Ace, Kinjiro Nakamura, a table tennis expert and participant in the 1992 Olympics, commented that: "… no one else would have been able to do that. I didn't think it was possible. But the fact that it was possible … means that there is a possibility that a human could do it too".

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41586-026-10338-5.

1. Vinyals, O. et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **575**, 350–354 (2019).
2. Wurman, P. R. et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature* **602**, 223–228 (2022).
3. D'Ambrosio, D. B. et al. Achieving human level competitive robot table tennis. In *Proc. 7th Robot Learning Workshop: Towards Robots with Human-Level Abilities* (ICLR, 2025).
4. Gallego, G. et al. Event-based vision: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 154–180 (2022).
5. Kaufmann, E. et al. Champion-level drone racing using deep reinforcement learning. *Nature* **620**, 982–987 (2023).
6. Tang, H.-P., Mizoguchi, M. & Toyoshima, S. Speed and spin characteristics of the 40mm table tennis ball. *Table Tennis Sci.* **4**, 278–284 (2002).
7. Billingsley, J. Robot ping-pong. *Pract. Comput.* **6**, 99–100 (1983).
8. Senoo, T., Namiki, A. & Ishikawa, M. High-speed batting using a multi-jointed manipulator. In *Proc. IEEE International Conference on Robotics and Automation*,1191–1196 (IEEE, 2004).
9. Mülling, K., Kober, J. & Peters, J. A biomimetic approach to robot table tennis. *Adapt. Behav.* **19**, 359–376 (2011).
10. Liu, C., Hayakawa, Y. & Nakashima, A. Racket control and its experiments for robot playing table tennis. In *Proc. 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 241–246 (IEEE, 2012).
11. Huang, Y., Schölkopf, B. & Peters, J. Learning optimal striking points for a ping-pong playing robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4587–4592 (IEEE, 2015).
12. Huang, Y., Büchler, D., Koç, O., Schölkopf, B. & Peters, J. Jointly learning trajectory generation and hitting point prediction in robot table tennis. In *Proc. 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 650–655 (IEEE, 2016).
13. Koç, O., Maeda, G. & Peters, J. Online optimal trajectory generation for robot table tennis. *Rob. Auton. Syst.* **105**, 121–137 (2018).
14. Yang, L., Zhang, H., Zhu, X. & Sheng, X. Ball motion control in the table tennis robot system using time-series deep reinforcement learning. *IEEE Access* **9**, 99816–99827 (2021).
15. Büchler, D. et al. Learning to play table tennis from scratch using muscular robots. *IEEE Trans. Robot.* **38**, 3850–3860 (2022).
16. Lin, H.-I. & Syu, C.-F. Table tennis ball landing control in a robotic system by cameras. *Robotica* **42**, 3867–3887 (2024).
17. Nguyen, D., Cancio, K. D. & Kim, S. High speed robotic table tennis swinging using lightweight hardware with model predictive control. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 15278–15284 (IEEE, 2025).
18. Mülling, K., Kober, J., Kroemer, O. & Peters, J. Learning to select and generalize striking movements in robot table tennis. *Int. J. Rob. Res.* **32**, 263–279 (2013).
19. Asai, K., Nakayama, M. & Yase, S. The ping pong robot to return a ball precisely: trajectory prediction and racket control for spinning balls. *Omron Technics* **51**, 016 (2020).
20. Tebbe, J., Krauch, L., Gao, Y. & Zell, A. Sample-efficient reinforcement learning in robotic table tennis. In *Proc. 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 4171–4178 (IEEE Press, 2021).
21. Su, Z. et al. HITTER: a humanoid table tennis robot via hierarchical planning and learning. Preprint at https://arxiv.org/abs/2508.21043 (2025).
22. Huang, Y., Xu, D., Tan, M. & Su, H. Adding active learning to LWR for ping-pong playing robot. *IEEE Trans. Control Syst. Technol.* **21**, 1489–1494 (2013).
23. Tebbe, J., Klamt, L., Gao, Y. & Zell, A. Spin detection in robotic table tennis. In *Proc. 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 9694–9700 (IEEE, 2020).
24. Yu, Z. et al. Design of a humanoid ping-pong player robot with redundant joints. In *Proc. 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 911–916 (IEEE, 2013).
25. Matsushima, M., Hashimoto, T., Takeuchi, M. & Miyazaki, F. A learning approach to robotic table tennis. *IEEE Trans. Robot.* **21**, 767–771 (2005).
26. Gao, W. et al. Robotic table tennis with model-free reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5556–5563 (IEEE, 2020).
27. Eschmann, J., Albani, D. & Loianno, G. Learning to fly in seconds. *IEEE Robot. Autom. Lett.* **9**, 6336–6343 (2024).
28. Geles, I., Bauersfeld, L., Romero, A., Xing, J. & Scaramuzza, D. Demonstrating agile flight from pixels without state estimation. In *2024 Proceedings of Robotics: Science and Systems (RSS)* (2024).
29. Vasco, M. et al. A super-human vision-based reinforcement learning agent for autonomous racing in Gran Turismo. In *2024 Reinforcement Learning Conference (RLC)* (2024).
30. Brescianini, D., Dürr, P. & Kamm, M. Tracking camera, tracking camera systems, and operation thereof. Patent WO2022106233A1 (2021).
31. Okumura, K., Oku, H. & Ishikawa, M. High-speed gaze controller for millisecond-order pan/tilt camera. In *Proc. 2011 IEEE International Conference on Robotics and Automation*, 6186–6191 (IEEE, 2011).
32. Miyashita, L. & Ishikawa, M. Saccade Argos: hierarchical robust tracking system for high spatio-temporal resolution vision. In *Proc. 2025 IEEE/SICE International Symposium on System Integration (SII)*, 811–816 (IEEE, 2025).
33. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778 (IEEE, 2016).
34. Nakabayashi, T., Higa, K., Yamaguchi, M., Fujiwara, R. & Saito, H. Event-based ball spin estimation in sports. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 3367–3375 (IEEE, 2024).
35. Gossard, T., Krismer, J., Ziegler, A., Tebbe, J. & Zell, A. Table tennis ball spin estimation with an event camera. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 3347–3356 (IEEE, 2024).
36. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. International Conference on Machine Learning Research*, 1861–1870 (PMLR, 2018).
37. Kiemel, J. C. & Kröger, T. Learning collision-free and torque-limited robot trajectories based on alternative safe behaviors. In *Proc. 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 223–230 (2022).
38. Bäck, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (Oxford Univ. Press, 1996).
39. Zagatto, A. M., Morel, E. A. & Gobatto, C. A. Physiological responses and characteristics of table tennis matches determined in official tournaments. *J. Strength Cond. Res.* **24**, 942–949 (2010).
40. Wang, Z., Boularias, A., Mülling, K., Schölkopf, B. & Peters, J. Anticipatory action selection for human–robot table tennis. *Artif. Intell.* **247**, 399–414 (2017).
41. Muelling, K., Boularias, A., Mohler, B., Schölkopf, B. & Peters, J. Learning strategies in table tennis using inverse reinforcement learning. *Biol. Cybern.* **108**, 603–619 (2014).
42. Ji, Y. et al. Opponent hitting behavior prediction and ball location control for a table tennis robot. *Biomimetics* **8**, 229 (2023).
43. McGill, R., Tukey, J. W. & Larsen, W. A. Variations of box plots. *Am. Stat.* **32**, 12–16 (1978).

Methods

Coordinate system

We use right-handed conventions with the origin of the coordinate system at the centre of the playing surface of the table, in which the x-axis points towards the human player side of the table and the z-axis points upwards.

Perception

Ball triangulation. We use nine cameras synchronized with the actuators of the robot with a 200 Hz trigger signal to accurately locate the ball in the volume of the Olympic-sized court. At each trigger event, cameras capture 1,440 × 1,080 pixel Bayer8 colour images. To reduce data transfer and improve scalability (more cameras can increase robustness and accuracy[44]), each camera is equipped with a hardware-accelerated field programmable gate array to facilitate two-dimensional (2D) ball detection. The field programmable gate arrays process the images through a segmentation pipeline to produce a compressed 2D detection mask, which is streamed to a central server through an embedded CPU. The server verifies the shape of the ball and triangulates its 3D position using pre-calibrated camera parameters. The entire process is completed within 10.2 ms.

Camera placement is optimized using a custom covariance matrix adaptation evolution strategy (CMA-ES) algorithm[45]. The optimizer determines the lens selection, mounting height and orientation for each camera, subject to constraints such as the number of towers, desired coverage volume and a minimum projected 2D ball radius (5 pixels).

Spin estimation. The angular velocity of the ball is estimated by observing the movement of the logo printed on the surface of the official ball. To accurately capture the high-speed moving and rotating logo, we develop a mirror-based event vision tracking system called the gaze control system (GCS). The GCS comprises three components: (1) an event camera[4] for low-latency, low-motion-blur imaging; (2) a telephoto, electrically tunable lens to magnify the ball and keep it in focus; and (3) a set of rotatable mirrors to track the ball smoothly (Fig. 2d). Given the 3D triangulation results, the mirrors and lens are controlled to track and focus on the ball with the system delay compensated by predicting the ball trajectory using the ball aerodynamics. With the ball being tracked, its contour on the event camera frame is first detected by a CNN[46]. Then the events on the ball are processed by two spin estimators, namely, a low-latency estimator based on another CNN[33] and a high-accuracy but slower estimator based on CMax[34]. The CNN estimates the angular velocities with heteroscedastic uncertainties from accumulated events and is trained on pseudo-ground-truth data obtained by CMax using heteroscedastic regression[47].

Events are aggregated into a polarity-separated surface of active events[48] of 15 ms accumulation time window in which timestamps are minimum/maximum normalized to a range between 0 and 1. We use a centred 320 × 320 pixel hardware crop of the original 1,280 × 720 pixel.

The angular velocities estimated by the CNN are refined asynchronously by CMax. To achieve both low-latency and high accuracy, the robot agent Ace uses the angular velocities obtained by the CNN at the beginning of the trajectory and switches to the ones obtained by CMax as soon as they become available with low uncertainty. Because the spin estimation uncertainty increases when the logo is invisible, we place three GCSs to track the ball from multiple perspectives, as shown in Fig. 2a, and combine the multi-view measurements based on the respective uncertainties.

Simulation

Ball aerodynamics. The aerodynamics of the ball in flight are governed by the drag $\mathbf{f}_d$, Magnus $\mathbf{f}_M$ and gravitational $\mathbf{f}_g$ forces. Given that the ball's angular velocity $\boldsymbol{\omega}$ is approximately constant over short flight intervals, the flight dynamics can be modelled as

$$m\dot{\mathbf{v}} = \mathbf{f}_d + \mathbf{f}_M + \mathbf{f}_g = -\frac{1}{2}c_d\rho_{air}r^2\pi\|\mathbf{v}\|\mathbf{v} - c_M\rho_{air}\frac{4}{3}r^3\pi\mathbf{v}\times\boldsymbol{\omega} + m\mathbf{g} \quad (1)$$

where $\mathbf{v}$ is the ball velocity, $\rho_{air} = 1.204$ kg m^{-3} (density of dry air at room temperature and standard pressure), $m = 2.7 \times 10^{-3}$ kg (ball mass), $r = 0.02$ m (ball radius), $c_d = 0.55$ (drag coefficient), and $\mathbf{g} = [0, 0, -9.81]^T$ m s^{-2} (gravitational acceleration). Unlike the base model[49], which treats the Magnus coefficient c_M as constant, we modelled it as $c_M = 0.1\frac{\|\mathbf{v}\|}{r\,\|\boldsymbol{\omega}\|} - 0.001$.

Ball–table contact model. The table contact model[49], which assumes instantaneous point contact, is enhanced to capture some effects of surface contacts on the coefficient of restitution, ε^{table}, by modelling it as $\varepsilon^{table} = 0.98 - 0.02v_z$.

$$\mathbf{v}^+ = C_{v,v}^{table}\mathbf{v}^- + C_{v,\omega}^{table}\boldsymbol{\omega}^- \quad (2)$$

$$\boldsymbol{\omega}^+ = C_{\omega,v}^{table}\mathbf{v}^- + C_{\omega,\omega}^{table}\boldsymbol{\omega}^- \quad (3)$$

$$C_{v,v}^{table} = \begin{bmatrix} 1-\alpha & 0 & 0 \\ 0 & 1-\alpha & 0 \\ 0 & 0 & -\varepsilon^{table} \end{bmatrix} \quad C_{v,\omega}^{table} = \begin{bmatrix} 0 & \alpha r & 0 \\ -\alpha r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$C_{\omega,v}^{table} = \begin{bmatrix} 0 & -\frac{3\alpha}{2r} & 0 \\ \frac{3\alpha}{2r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad C_{\omega,\omega}^{table} = \begin{bmatrix} 1-\frac{3\alpha}{2} & 0 & 0 \\ 0 & 1-\frac{3\alpha}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where superscripts '−' and '+' are pre- and post-contact quantities, respectively, and

$$\alpha = \alpha(\mathbf{v}^-, \boldsymbol{\omega}^-) = \begin{cases} \mu(1+\varepsilon^{table})\frac{v_z^-}{\|\mathbf{v}_T\|} & (v_s > 0) \\ \frac{2}{5} & (v_s \le 0) \end{cases} \quad (4)$$

where the contact type is determined as sliding if $v_s > 0$ and rolling if $v_s \le 0$, with

$$v_s = v_s(\mathbf{v}^-, \boldsymbol{\omega}^-) = 1 - \frac{5}{2}\mu(1+\varepsilon^{table})\frac{v_z^-}{\|\mathbf{v}_T\|}, \quad (5)$$

$$\mathbf{v}_T = \begin{bmatrix} v_x^- - r\omega_y^- \\ v_y^- + r\omega_x^- \\ 0 \end{bmatrix}. \quad (6)$$

ε^{table} and μ are the coefficient of restitution and the dynamic coefficient of friction between ball and table, respectively, modelled as $\varepsilon^{table} = \varepsilon^{table}(v_z^-) = 0.98 - 0.02v_z^-$ and $\mu = 0.25$ from experimental data.

Ball–racket contact model. The linear model proposed in the literature[49] is extended to handle the wide ranges of linear and angular velocities encountered in professional-level table tennis by incorporating (1) a velocity-dependent coefficient of restitution and (2) a residual correction neural network to correct model errors. The base linear model shares the same structure as in equations (2) and (3) and is defined as

$$\mathbf{v}^+ = R^T C_{v,v}^{racket} R(\mathbf{v}^- - \mathbf{v}^{racket}) + R^T C_{v,\omega}^{racket} R\boldsymbol{\omega}^- + \mathbf{v}^{racket} \quad (7)$$

$$\boldsymbol{\omega}^+ = R^T C_{\omega,v}^{racket} R\mathbf{v}^- + R^T C_{\omega,\omega}^{racket} R\boldsymbol{\omega}^- \quad (8)$$

with

$$C_{v,v}^{\text{racket}} = \begin{bmatrix} 1-k & 0 & 0 \\ 0 & 1-k & 0 \\ 0 & 0 & -\varepsilon^{\text{racket}} \end{bmatrix} \quad C_{v,\omega}^{\text{racket}} = \begin{bmatrix} 0 & kr & 0 \\ -kr & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$C_{\omega,v}^{\text{racket}} = \begin{bmatrix} 0 & -\frac{3k}{2r} & 0 \\ \frac{3k}{2r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad C_{\omega,\omega}^{\text{racket}} = \begin{bmatrix} 1-\frac{3k}{2} & 0 & 0 \\ 0 & 1-\frac{3k}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where R is the rotation matrix from the local frame of the racket to the global frame of reference, $\mathbf{v}^{\text{racket}}$ is the racket linear velocity at impact and k is a coefficient relating tangential quantities. $\varepsilon^{\text{racket}} = \gamma_1 e^{\gamma_2 |v_z'|}$ is modelled as a function of the normal relative velocity v_z' by fitting coefficients γ_1, γ_2 on data collected from games. The residual correction neural network is a small multilayer perceptron trained on game data and corrects both velocity and angular velocity error by 4% on average.

Sensor modelling. To model the ball triangulation obtained from APS cameras, we sample latency from a uniform distribution and noise from a zero-mean Gaussian distribution, and apply dropout of sensor measurements with a fixed probability. For the spin estimation, latency and dropout are modelled similarly, with additional dropout applied directly after racket contact to reflect tracking loss of GCS around these events. Both precision (sensor noise) and accuracy (sensor bias) of GCS are modelled using separate zero-mean Gaussian distributions for spin magnitude and axis. However, accuracy is sampled once per contact event to mimic the bias introduced by GCS reinitialization at these events.

Physics perturbations. To improve the simulation-to-reality transfer, ball state perturbations are added after table contact. Each Cartesian component of ball linear and angular velocities is perturbed independently using a zero-mean Gaussian distribution.

Robot dynamics. Robot joints are modelled as decoupled, delayed linear time-invariant systems, in which each joint i is described by

$$\begin{aligned} \dot{\boldsymbol{\zeta}}_i(t) &= A_i \boldsymbol{\zeta}_i(t) + \mathbf{b}_i u_i(t - \tau_{\text{d},i}) \\ \psi_i(t) &= \mathbf{c}\boldsymbol{\zeta}_i(t) \end{aligned} \tag{9}$$

with $\boldsymbol{\zeta}_i(t) = [q_i, q_{\text{aux},i}]^{\text{T}}$, where q_i is the joint position and $q_{\text{aux},i}$ is an auxiliary state (for example, velocity) at time t. ψ_i is the observable output, and $u_i = q_i^*$ is the desired position input. The state transition, input and output matrices are $A_i \in \mathbb{R}^{2\times 2}$, $\mathbf{b}_i \in \mathbb{R}^2$ and $c = [1, 0]$, respectively. $\tau_{d,i}$ is a joint-specific control delay.

The parameters A_i, $\mathbf{b}_i$ and $\tau_{\text{d},i}$ are identified from first estimating the parameters of a continuous-time process model with input delay using recorded data and then converting the process model into a state space model.

Rally

The rally phase in table tennis refers to the sequence of shots after the serve. Ace segments this into individual episodes, from the opponent hit to the robot response. During each episode, the robot is controlled by RL policies that generate segment trajectories every 32 ms ($Q_t^{*[1:T]} \in \mathbb{R}^{N_q \times T}$, where $T = 32$ and N_q is the number of joints).

Reinforcement learning framework. Episode definition. An episode begins when the ball is in free flight, moving towards the robot. An episode ends when the ball meets one of four conditions: (1) the ball is out of play or no longer legal; (2) the robot hits the ball; (3) the ball passes the racket of the robot; and (4) the joint trajectory produced by Ace would result in a collision with itself or the table.

Rewards. The reward function used during training consists of several terms, all of which are calculated after the episode has finished, that is, as a function of the terminal state. Although reward terms vary across policies to induce different skills, they can be categorized by assigning specific rewards for (1) missing the ball; (2) hitting the ball but failing to return it; or (3) successfully returning the ball:

$$\begin{cases} R_{\text{miss}} & \text{if robot fails to hit the ball} \\ R_{\text{hit}}^{\neg\text{return}} & \text{if robot hits the ball but fails to return it} \\ R_{\text{hit}}^{\text{return}} & \text{if robot hits the ball and returns it} \end{cases} \tag{10}$$

A subset of the policies use a reward formulation for $R_{\text{hit}}^{\text{return}}$ that can be parameterized by a desired y-landing position (y_{desired}) and a set of reward weights ($\mathbf{w}_{\text{reward}} = [w_{\text{p}}, w_{\text{s}}]$, where $w_{\text{p}} \in [0, 1]$ and $w_{\text{s}} \in [-1, 1]$). y_{desired} is used to calculate a reward based on the distance between y_{desired} and the achieved y-landing position, w_{p} is used to weight this distance reward and w_{s} is used to weight a term proportional to the angular velocity in the y-axis of the ball frame on landing. By sampling these conditioning variables, these policies can exhibit a variety of different behaviours such as aiming, topspin and backspin.

States. The state in our RL framework can be written as $\mathbf{s}_t = [\mathbf{s}_t^{\text{ball}}, \mathbf{s}_t^{\text{robot}}, \mathbf{s}_t^{\text{skill}}]$. $\mathbf{s}_t^{\text{ball}}$ is the ball state consisting of ball position and spin histories of length N, along with their associated timestamps. $\mathbf{s}_t^{\text{robot}}$ is the robot state and consists of the joint states (position, velocity and acceleration) and end effector state (pose and twist) associated with the terminal state of $Q_{t-1}^{*[1:T]}$ (for further details see Supplementary Information section 1.4.1). For policies trained with parameterized reward functions, the state is further augmented with $\mathbf{s}_t^{\text{skill}}$, which is the fixed skill state composed of y_{desired} and $\mathbf{w}_{\text{reward}}$. $\mathbf{s}_t$ is used to infer actions $\mathbf{a}_t$, which are subsequently mapped to joint trajectories and reset plans (see Supplementary Information sections 1.4.2 and 1.4.3). This process requires a time budget of 5 ms and so $\mathbf{s}_t$ must be constructed 5 ms before the next set of commands is sent to the robot (Extended Data Fig. 1).

Actions. Actions, $\mathbf{a}_t \in [-1, 1]^{2N_q}$, are sampled from a tanh squashed multivariate Gaussian distribution. This forms an abstract space, in which for each joint there are two actions that define a target joint position and velocity 32 ms into the future (see Supplementary Information section 1.4.2).

Transition probability function. The transition probability function is as follows:

$$\begin{cases} \mathbf{s}_{t+1}^{\text{ball}} \sim f_{\text{ball}}(\mathbf{s}_t, \mathbf{a}_t) \\ \mathbf{s}_{t+1}^{\text{robot}} = f_{\text{robot}}(\mathbf{s}_t, \mathbf{a}_t) \\ \mathbf{s}_{t+1}^{\text{skill}} = \mathbf{s}_t^{\text{skill}} \end{cases} \tag{11}$$

where f_{ball} is a stochastic function that depends on sensors and physics modelling in the simulator, and f_{robot} is a deterministic function depending on $Q_{t-1}^{*[1:T]}$ and the robot dynamics.

Initial state training distribution. During training in simulation, an episode starts with an initial state that is sampled from three independent distributions:

1. Initial ball state $\mathbf{s}_0^{\text{ball}}$: the initial state of the ball is sampled from a kernel density estimation (KDE) model fit either to synthetic or human data. For the synthetic dataset, shots are uniformly sampled from a range of initial ball states and checked for validity. KDE models are generated for both returns (that is, shots performed during a rally) and serves. During training, serves and returns are sampled at a ratio of 3:7, and the initial state is sampled with a fixed probability from either the synthetic or the human KDE models (Supplementary Information section 1.4.3).
2. Initial robot state $\mathbf{s}_0^{\text{robot}}$: the initial robot state can be static or dynamic. Static states are sampled with the arm in a neutral configuration, and prismatic actuators are initialized uniformly within their allowed

range, whereas dynamic states are sampled from reset plans stored during previous training episodes.
3. Initial skill state $\mathbf{s}_0^{\text{skill}}$: y_{desired} is sampled uniformly within the bounds of the opponent's side of the table. $\mathbf{w}_{\text{reward}}$ is sampled in a way that is biased towards sparse reward weight vectors and boundary values (Supplementary Information section 1.4.5).

Algorithm. To train the deep RL policy, we use SAC[36] asynchronously with multiple data collection tasks in parallel[2] (see Supplementary Table 8 for hyperparameters). We use asymmetric actor–critic[27–29], providing the ground-truth ball state from the simulator to the critic and sequences of sensor measurements to the actor. Apart from the standard policy loss, an auxiliary loss is added to the policy to reconstruct the ground truth ball state from its ball state embedding. When collecting experience, we apply three different forms of data augmentation as follows:
1. Symmetric augmentation to mirror all states, actions and rewards with respect to the *XZ* plane (that is, the plane containing the centre-line of the table and perpendicular to both the table and the net).
2. Event tables[50] to store transitions leading to predetermined events in separate replay buffers for stratified sampling of the mini-batch. The events used in our training pipeline are defined based on heuristics and include the following events: near miss, ball hit, ball returned, high-speed return, high-top-spin return, high-back-spin return (see Supplementary Information section 1.4.8).
3. Hindsight experience replay[51] to augment RL transitions with an additional copy in which y_{desired} is equal to the achieved position, w_{p} is equal to 1, and the maximum position-based reward is given.

Feasible action for optimal control. Mapping algorithm. The action $\mathbf{a}_t$ sampled from the deep RL policy is mapped from the abstract set $[-1, 1]^{2N_q}$ to the feasible set of joint position and velocity pairs 32 ms in the future, using a mapping algorithm. The generic mapping algorithm can be stated as follows: Let $\mathbb{X} \subset \mathbb{R}^n$ be the compact base set with centre $\bar{\mathbf{x}}$ and $\mathbb{Y} \subset \mathbb{R}^n$ be the compact target set with centre $\bar{\mathbf{y}}$. For a given mapping ($\mathbf{x}_i \in \mathbb{X}$, $\mathbf{y}_i \in \mathbb{Y}$), if $\mathbf{y}_i = \bar{\mathbf{y}}$ then $\mathbf{x}_i = \bar{\mathbf{x}}$, otherwise

$$\begin{aligned} \mathbf{x}_i &= \bar{\mathbf{x}} + \boldsymbol{\delta}_i \\ \mathbf{y}_i &= \bar{\mathbf{y}} + \frac{\beta_i}{\alpha_i} f(\boldsymbol{\delta}_i) \\ \alpha_i \geq 1 &: \bar{\mathbf{x}} + \alpha_i \boldsymbol{\delta}_i \in \partial\mathbb{X} \\ \beta_i > 0 &: \bar{\mathbf{y}} + \beta_i f(\boldsymbol{\delta}_i) \in \partial\mathbb{Y} \end{aligned} \tag{12}$$

where $\partial\mathbb{X}$ and $\partial\mathbb{Y}$ are the boundaries of $\mathbb{X}$ and $\mathbb{Y}$, respectively, $\bar{\mathbf{y}}$ the centre of $\mathbb{Y}$. The ratio $\frac{\beta_i}{\alpha_i}$ determines the location of $\mathbf{y}_i$ between $\bar{\mathbf{y}}$ and $\partial\mathbb{Y}$, whereas the function $f(\cdot)$ modifies $\boldsymbol{\delta}_i$ to account for the shape differences between $\mathbb{X}$ and $\mathbb{Y}$. The mapping is bijective, invertible and centre to centre and boundary to boundary of the map sets.

Optimization problem formulation. We use the result of the mapping as a terminal position and velocity constraint for an optimization problem that computes reference trajectories for each robot joint as cubic splines that minimize jerk. By definition of the problem, the result of the mapping is always inside the maximum control invariant set, which is the largest subset of the feasible state space containing the initial states from which the associated MPC problem is recursively feasible[52] (Supplementary Information section 1.5.1). The result of the mapping forms the initial state for the next optimization problem. The optimization is solved using DAQP[53] and sampled at 1 kHz to generate $Q_t^{*[1:T]}$.

Reset trajectories. For every $Q_t^{*[1:T]}$ produced, a reset trajectory is required that moves the robot from the terminal state of $Q_t^{*[1:T]}$ to a target stationary reset position. Ace uses a near time-optimal variation of MPC (see Supplementary Information section 1.5.2) to generate these reset trajectories. They are executed as soon as one of the termination criteria for the RL episode is satisfied (Supplementary Information section 1.4.1). If the episode is terminated due to a predicted collision, then the reset trajectory from the previous RL step is executed.

The target reset position is chosen as either a constant neutral configuration or a configuration computed by a prepare policy network. The prepare policy is trained using a dataset constructed from elite-level rallies for high-dexterity shot execution. From each recorded rally, we extract (1) the ball state at the start of an episode, $\mathbf{s}_0^{\text{ball}}$; (2) y_{desired}; and (3) subsequent racket position $\mathbf{x}_{t_c}^{\text{racket}}$ executed by robot at contact time t_c. For each $\mathbf{x}_{t_c}^{\text{racket}}$, we compute offline the optimal reset configuration $\mathbf{q}_{\text{reset}}^*$ that maximizes a dexterity objective $D(\mathbf{q}, \mathbf{x}_{t_c}^{\text{racket}})$, considering kinematic constraints. This process yields a training dataset $\mathcal{D} = \{(\mathbf{s}_0^{\text{ball}}, y_{\text{desired}}, \mathbf{q}_{\text{reset}}^*)^i\}_{i=1}^M$ of M samples. During deployment for each shot, the agent receives $(\mathbf{s}_0^{\text{ball}}, y_{\text{desired}})$ as input and predicts the optimal $\mathbf{q}_{\text{reset}}^{\text{desired}}$ that supports dexterous execution of subsequent actions. $\mathbf{q}_{\text{reset}}^{\text{desired}}$ is sampled from a Gaussian distribution estimated from the N nearest reset configurations $\{\mathbf{q}_{\text{reset}}^{*j}\}_{j=1}^N$ to $\mathbf{q}_{\text{reset}}^*$ of $(\mathbf{s}_0^{\text{ball}}, y_{\text{desired}})$ found by KD-tree (k-dimensional tree) search on the dataset.

Policy sampler. Ace uses multiple rally-specific policies trained to optimize different objectives and therefore requires a sampling strategy during matches. Ace uses four different strategies for sampling the policies (see Supplementary Information section 1.7.1 for details):
1. Fixed: a single policy is sampled with a fixed probability of 1.
2. Random: a policy is chosen at random on a shot-by-shot basis from a subset of policies.
3. Heuristic: a set of heuristics dictates the policy sampling on a shot-by-shot basis. The heuristics map the characteristics of the incoming ball to the most appropriate policy.
4. Data-driven: a supervised learning model is trained to classify winning and losing shots based on data from elite table tennis players other than the seven players in the evaluation. The model is used to identify shots with the highest predicted win rate, and the policy most capable of producing those shots is sampled.

For policies conditioned on y_{desired} and $\mathbf{w}_{\text{reward}}$, Ace samples them from the same fixed probability distribution used during training, that is, uniform for y_{desired} and sparse but biased towards boundary values for $\mathbf{w}_{\text{reward}}$. As these policies are conditioned on y_{desired}, they also afford the use of the prepare policy, which requires y_{desired} as input.

Serve design

Ace achieves ITTF-compliant serves by executing a single-arm toss using the ball cup mounted on its end effector (Fig. 2c), followed by striking the ball during its free fall. Although standard ITTF rules require a free-hand toss, one-handed serves are permitted when a player has a physical disability that impedes them from properly tossing the ball with the free hand, providing a precedent for our implementation.

For the serve tossing, we collect human serve demonstrations and re-target them to the kinematics of the robot using an optimization procedure[54]. The resulting motion is a trajectory of joint commands $\mathbf{u}_{\text{toss}}(t)$ that produces a valid ball toss when executed by the robot. We define t_{lift} as the time index of the tossing trajectory in which the acceleration of the ball approximates that of gravity, meaning that the ball has been released from the cup.

In simulation, the ball-striking motion $\mathbf{u}_{\text{strike}}(t)$ is obtained by connecting the current state of the robot at $t = t_{\text{lift}}$ to a racket state produced by a genetic algorithm (GA). This connection trajectory is generated by an MPC in racket space, for which the GA searches the optimal parameters $\boldsymbol{\xi}_s^*$ (Supplementary Information section 1.8.2) that maximize a fitness function $\mathcal{F}_\theta(\cdot)$. This fitness function is designed to capture serve metrics of interest (for example, ball velocity, spin, landing position),

which conditions the type of serves produced. The final serve trajectory is the concatenation of $\mathbf{u}_{\text{toss}}(t)$, $t \in [0, t_{\text{lift}}]$ followed by $\mathbf{u}_{\text{strike}}(t)$.

As $\mathbf{u}_{\text{strike}}(t)$ is based on simulated physics, we assess the effectiveness of each serve on the real robot in dedicated sessions with a coach. Serves deemed sufficiently challenging for matches undergo repeated open-loop execution (at least 20 times) to verify their reliability. If the failure probability of a serve is less than 5%, it is added to the library for open-loop use during matches. If the probability exceeds 5%, we attempt closed-loop MPC execution, where the parameters $\boldsymbol{\xi}_s^*$ are updated online with actual hitting states output by a ball flight predictor. If this procedure successfully decreases the failure rates to 5% or less, the serve is added to the library for closed-loop use. Details on how the serves were selected from the library can be found in Supplementary Information section 1.8.1.

Experimental protocol

Players warm up with another player for up to 15 min before the match. The player practices with the robot immediately before the start of the match for 2 min, as directed by the rules of ITTF (https://www.ittf.com). During this practice period, Ace uses a policy that returns balls to a fixed position with moderate top spin, as is common practice. Players were informed that if they enter the court side of the robot, a safety light curtain triggers an emergency system stop. However, crossing the centre line is rare in high-level games, and such a trigger never occurred in our experiments. They were instructed to wear goggles to protect their eyes. They choose goggles from a variety of sizes and colours to minimize the impact on their performance. The player decides whether to serve or receive first. The player is eligible to call a 1-min time-out during the match. Elite D used this option during the third game. Following the ITTF rules, the robot racket is shown to the player and umpire before the match. All equipment, including table (SAN-EI), net (Butterfly), balls (Nittaku), racket (VICTAS and Butterfly) and floor mat, is approved by the ITTF. We use diffused lights to ensure a uniform light intensity over the whole playing area with around 1,400 lux as directed in the rules of ITTF.

Data availability

Supplementary videos, including the full games, are available at https://sonyresearch.github.io/ace_public/. The data underlying the graphical representations used in the figures will be made available in spreadsheet form.

Code availability

Pseudocode detailing the training process and key algorithms is available in Supplementary Information, as well as online at https://github.com/SonyResearch/ace_public.

44. Gomez-Gonzalez, S., Nemmour, Y., Schölkopf, B. & Peters, J. Reliable real-time ball tracking for robot table tennis. *Robotics* **8**, 90 (2019).
45. Hansen, N. in *Towards a New Evolutionary Computation. Studies in Fuzziness and Soft Computing* Vol. 192 (eds Lozano, J. A. et al.) 75–102 (Springer, 2006).
46. Li, C. et al. Yolov6: a single-stage object detection framework for industrial applications. Preprint at https://arxiv.org/abs/2209.02976 (2022).
47. Nix, D. A. & Weigend, A. S. Estimating the mean and variance of the target probability distribution. In *Proc. 1994 IEEE International Conference on Neural Networks (ICNN'94)*, 55–60 (IEEE, 1994).
48. Benosman, R., Clercq, C., Lagorce, X., Ieng, S.-H. & Bartolozzi, C. Event-based visual flow. *IEEE Trans. Neural Netw. Learn. Syst.* **25**, 407–417 (2014).
49. Nakashima, A., Ogawa, Y., Liu, C. & Hayakawa, Y. Robotic table tennis based on physical models of aerodynamics and rebounds. In *Proc. 2011 IEEE International Conference on Robotics and Biomimetics*, 2348–2354 (IEEE, 2011).
50. Kompella, V. R., Walsh, T., Barrett, S., Wurman, P. R. & Stone, P. Event tables for efficient experience replay. In *2023 Transactions on Machine Learning Research (TMLR)* (2023).
51. Andrychowicz, M. et al. Hindsight experience replay. In *Proc. 31st International Conference on Neural Information Processing Systems*, 5055–5065 (ACM, 2017).
52. Kerrigan, E. C. & Maciejowski, J. M. Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control. In *Proc. 39th IEEE Conference on Decision and Control*, Vol. 5, 4951–4956 (IEEE, 2000).
53. Arnström, D., Bemporad, A. & Axehill, D. A dual active-set solver for embedded quadratic programming using recursive LDL^T updates. *IEEE Trans. Automat. Contr.* **67**, 4362–4369 (2022).
54. Maeda, G., Ewerton, M., Koert, D. & Peters, J. Acquiring and generalizing the embodiment mapping from human observations to robot skills. *IEEE Robot. Automat. Lett.* **1**, 784–791 (2016).

Acknowledgements We thank K. Matsushita, K. Nakamura, K. Mizutani, Y. Kawamata and K. Arisawa, as well as R. Abe, M. Akagawa, M. Ando, N. Hagii, M. Igarashi, Y. Itabashi, H. Maehara, T. Nagao, H. Nozaki, A. Saeki, M. Sasaki, H. Sato, S. Sone, M. Taira, R. Takenaka, K. Yoshimura, U. Stamm, F. Karin and L. Posch, as well as S. Maeda Y. Sakagishi, M. Seki and M. Yamaguchi for their advice and help in testing our robot; S. Suzuki, Y. Nakajima, R. Garcia and E. Kato Marcus for their support with partnerships and communication; P. Khandelwal and his team for platform support; A. Stoimenov, B. Balaceanu and R. Meza for IT support; Z. Blattmann, S. Ohwada, R. Sekulic, T. Shinbo, A. Yamazaki and J. Zanardi for administrative support; S. Baba, M. Fujita, Y. Kajio, Y. Kamikawa, M. Kamm, Y. Kawanami, S. Kurihara, H. Owaki, K. Shibata, K. Suzuki and T. Tsuboi for their technical advice and support, D. Brescianini, S. Chopra, C. Fanconi, C. Fang, H. Firooz, R. Frauenfelder, T. Gossard, P. Haas, W. Huober, O. Koç, D. Krasheninnikov, A. Latorre, M. Lievense, L. Liu, L. Mariacourt, M. McAuliffe, S. Mirrazavi, M. Mohammedalamen, R. Müller, A. Patane, M. Patel, E. Saquand, K. Scheper, T. van de Wiel and A. Ziegler for their contributions, input and support with earlier iterations of the system; as well as A. Zell and J. Tebbe for their valuable advice. We also thank M. Tsukamoto for his outstanding support for the final set of experiments.

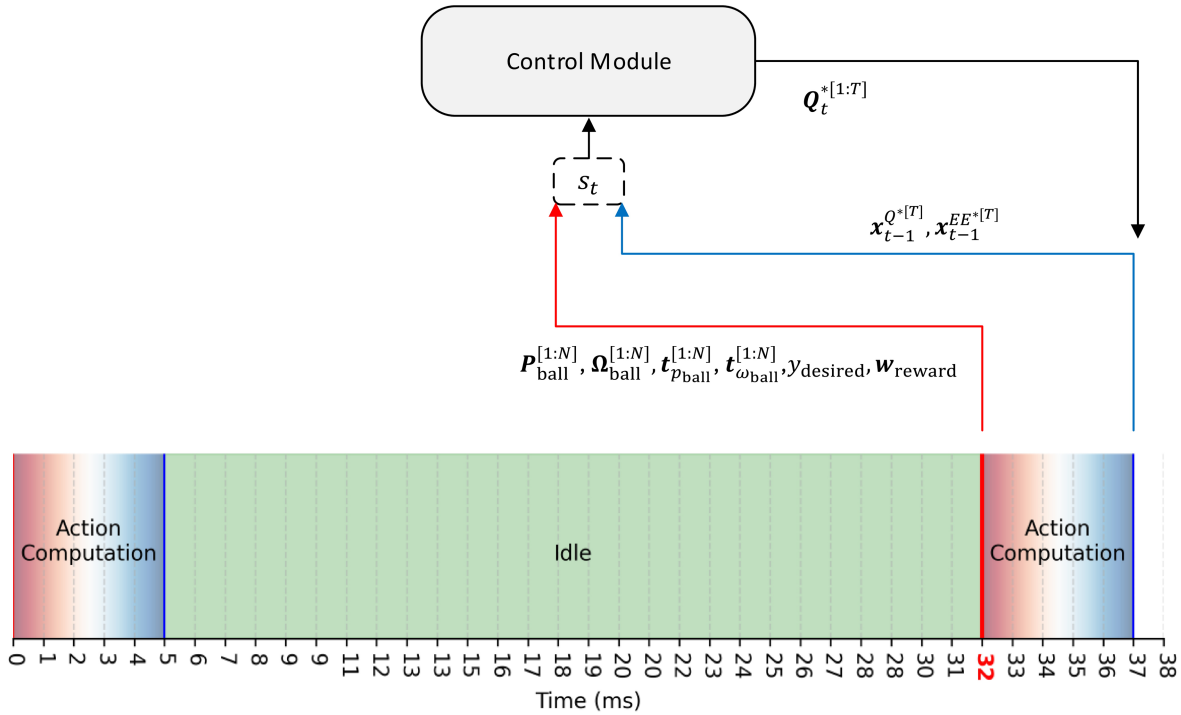
Author contributions P.D. managed the project. M.E.G., G.J.M., N.M. and N.T. led the research and development efforts. B.Y., C.C., J.W., T.S. and L. Miele contributed to the development of the simulation. S.B., M.Y.C., A.G., D.F.H., F.K., G.T., L.A., A. Scotti, Y.N., Y.-T.H., E.W., S.R., S.C., S.G., M.A.K.-J. and M.Scardecchia contributed to the simulation implementation and to the design and evaluation of the control system. A.A., F.S., H.S., R.K., R.T.M., Y.S., Y.B., Y.H., V.C. and A.P. contributed to developing the perception system. A. Sigrist, D.G., P.A., S.H. and T. Kakinuma were involved in the design and development of the robotic hardware. A. Saha, L. Martinez, V.M. and T. Kunori. contributed to the performance evaluation of the system. H.K., P.R.W., P.S. and M.Spranger provided technical advice, strategic oversight and institutional support for the project.

Competing interests P.D. and other team members have submitted multiple patent applications, including ref. 30, covering aspects of the robot hardware and gaze control system described in this paper.

Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41586-026-10338-5.

Correspondence and requests for materials should be addressed to Peter Dürr.

Peer review information *Nature* thanks Qinyu Chen, Carlos H. C. Ribeiro and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available.



Extended Data Fig. 1 | Temporal overview of the state components used by the rally control module. Every 32 ms the control module observes the current state (s_t) and computes a joint trajectory $Q_t^{*[1:T]}$ to be executed after 5 ms. s_t is constructed using both current and future information. The ball history ($P_{\text{ball}}^{[1:N]}, \Omega_{\text{ball}}^{[1:N]}, t_{p_{\text{ball}}}^{[1:N]}, t_{\omega_{\text{ball}}}^{[1:N]}$), desired y-landing position (y_{desired}) and reward weights (w_{reward}) are taken from the current time-step. While the robot joint state ($\mathbf{x}_{t-1}^{Q^*[T]}$) and end effector state ($\mathbf{x}_{t-1}^{EE^*[T]}$) represent states 5 ms into the future (i.e. when $Q_t^{*[1:T]}$ would be executed) based on the previously computed joint position commands that are currently being sent to the robot.