Whole-Body Model Predictive Control for Spin-Aware Quadrupedal Table Tennis

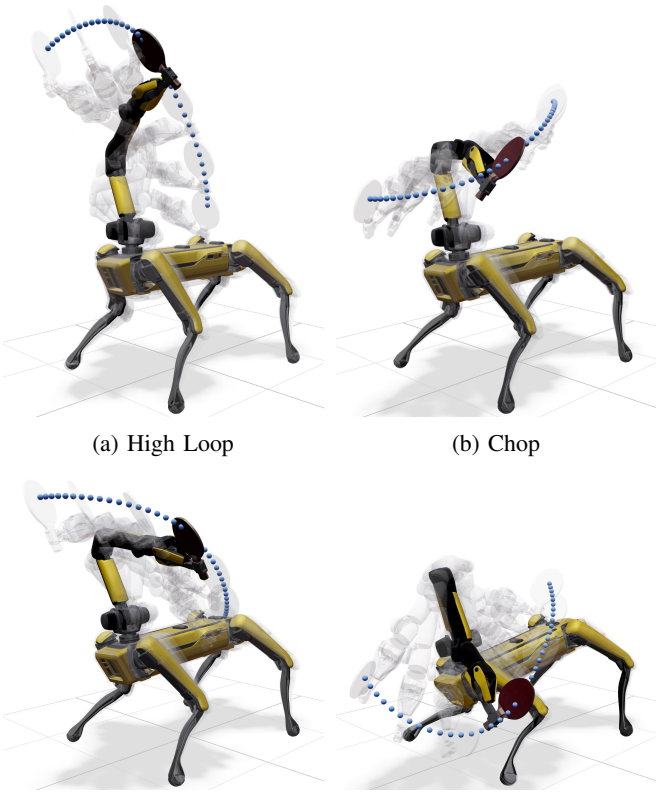
David Nguyen[1,3], Zulfiqar Zaidi[2,3], Kevin Karol[3], Jessica Hodgins[3], Zhaoming Xie[3]

***Abstract*— Developing table tennis robots that mirror human speed, accuracy, and ability to predict and respond to the full range of ball spins remains a significant challenge for legged robots. To demonstrate these capabilities, we present a novel continuous-time model predictive controller (MPC) for agile full-body control of a quadrupedal robot equipped with an arm. This formulation enables the flexibility to place time-critical constraints, such as those required to strike a ball, anywhere along the trajectory. We further develop a hybrid model-based and learned spin estimator that can accurately predict ball spin from its observed trajectory and an aiming planner that dictates how the ball must be struck. Notably, a continuous set of stroke strategies emerge automatically from different ball return objectives after combining the aiming planner and whole-body MPC. We demonstrate the system on hardware with a Spot quadruped, evaluate the accuracy of each system component, and exhibit coordination through the ability to aim and return balls with varying spin types. As a further demonstration, the system is able to rally with human players.**

I. Introduction

For generalized legged robots to become ubiquitous, we require them to perform manipulation tasks with comparable levels of speed and precision as humans. To do so, they should have agility that exceeds that of everyday tasks, which we can demonstrate through sports. Table tennis offers a unique test-bed for these kinds of problems in dynamic manipulation where players must predict, decide and respond on split-second time intervals. In a competitive table tennis game, a player must move to intercept and accurately hit a spinning $4\,\mathrm{cm}$ diameter ball with a racket, all while making strategic decisions in a fraction of a second [2].

For a robot to accurately control the trajectory and spin of a ping pong ball like this, it must solve four problems: ball perception, trajectory prediction, aiming, and swinging. First, the robot must accurately localize the ball which can travel at up to $10\,\mathrm{m\,s^{-1}}$ with $600\,\mathrm{rad\,s^{-1}}$ of spin [3], [4]. Next, to know where and when to strike the ball, the robot needs to anticipate its motion and estimate spin. To hit a ball on this predicted trajectory at a target location with specific spin, a planner must reason through contact and flight dynamics to choose the appropriate speed and orientation of the paddle. Finally, the robot must control its joints to reach the planned strike. Solving this series of problems at the speed of table tennis makes precise ball control an excellent case study for dynamic robotic control.



(a) High Loop (b) Chop

(c) Loop (d) Low Chop

Fig. 1: Variety of swing types including loop (top spin) and chop (back spin). Figures were generated using Viser [1].

Existing table tennis robot systems generally consist of a robotic arm either attached to a fixed base, e.g., [5]–[7], limiting their range of movement, or include a customized fast-moving gantry, e.g., [8], which reduces the challenge of agile motion control at the expense of generalizability across robot platforms. In contrast, quadrupedal robots must trade off high speed motion and active balance because of a floating base.

In this paper, we use a Boston Dynamics Spot equipped with a six degree of freedom (DoF) arm to play table tennis. We develop a continuous-time whole-body model predictive controller (MPC) that utilizes a Bezier curve parametrization. Unlike conventional MPC which uses a discrete-time trajectory, our formulation allows us to place constraints anywhere in time within the planning horizon which is advantageous

[1]Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2]Department of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[3]RAI Institute, Cambridge, MA 02142, USA

for time-critical constraints like striking the ball. We also address the challenge of handling ball spin in table tennis using a learned estimator that can accurately predict ball spin from an observed ball trajectory. A strike planner then optimizes for the desired paddle state given desired landing location and spin. All together, our system demonstrates the generality of our continuous-time whole-body MPC by autonomously discovering diverse stroke strategies in response to different spins. These behaviors mirror human techniques as seen in Figure 1, effectively neutralizing high incoming spin while simultaneously imparting desired counters without compromising balance.

To summarize, we make the following contributions:

- We introduce a continuous-time whole-body model predictive controller where time-critical constraints can be placed flexibly along its trajectory.
- We analyze how this new formulation enables superior computational efficiency and controller performance than a typical discrete multiple-shooting approach.
- We present the encompassing system which enables the proposed MPC to accurately handle and produce table tennis ball spin on a quadrupedal robot platform.

Our hardware system is capable of returning incoming spin of up to $280\,\mathrm{rad\,s^{-1}}$ and imparting outgoing spin of up to $200\,\mathrm{rad\,s^{-1}}$, surpassing prior work e.g. [9], [10], while also demonstrating consistent rallying with a human player.

II. Related Work

A. Robot Racket Sports

Robotic sports is a common challenge that fosters robotics research with the goal of matching the dynamic motion of humans and animals. Racket sports especially have become a popular test bed because of the need for agility, planning, and manipulation. Prior work includes robot systems that play tennis [11]–[13] and badminton [14], [15], but because of the pace and scale of the game, table tennis stands out within racket sports as a preferred robotics research challenge.

B. Perception in Table Tennis Robot

Previous works in robotic table tennis perception address ball localization [8], [16]–[19], trajectory prediction [16], [18], [20], [21], and spin estimation [18], [21], [22]. Like some existing systems [8], [17], [21], we employ a pair of high-speed RGB cameras for ball localization. However, many simpler trajectory prediction methods disregard ball spin [11], [20]. This omission is problematic because ball flight and contact dynamics are significantly influenced by spin [23], making it a critical component of the game. Some prior works estimate spin based on ball trajectory [16], [21] while Gossard et al. [24] track non-regulation markings on the ball. Other methods infer spin from human player poses [22], but this approach relies on noisy human pose measurements. Our system builds upon the methods of Tebbe et al. [21] by adding a learned neural network to the model-based estimate from trajectory curvature. This approach improves state prediction accuracy and handles incoming spin without the need for ball markings or inferring from human motions.

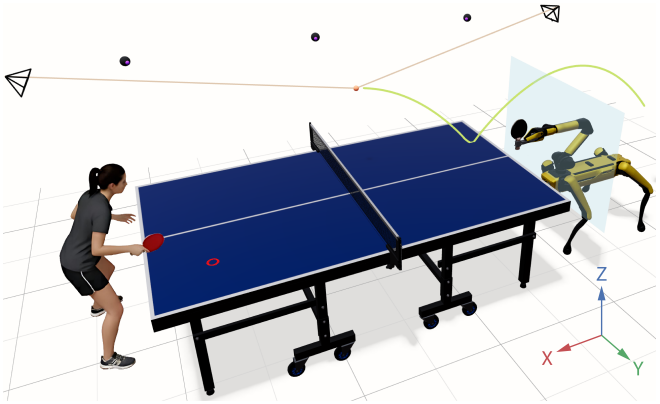


Fig. 2: System diagram with RGB cameras shown by pyramids that detect the ball in orange. Its predicted trajectory is shown in green with the strike plane in blue. The black Vicon cameras in the background observe the position of the robot, while the target ball landing location is in red on the table.

C. Control in Table Tennis Robot

To play table tennis effectively, the robot needs to perform accurate swing motions with high agility. Most prior works focus on control systems for a fully actuated robot arm that is either fixed or attached to a fast moving gantry. These systems often use human demonstrations to construct motion primitives [25] or as training data for imitation learning [26]. Reinforcement learning algorithms, e.g., [7], [15], [27] are often employed to synthesize table tennis skills, but require large quantities of simulated or real world data to perform effectively. These techniques also require separate controllers for categorized shots [27] because of the motion diversity of table tennis swings. On the other hand, model-based methods demonstrate effective control synthesis, e.g., [5], [28], without the need for data and can achieve greater shot diversity with a single controller. Our work extends model-based methods for robot table tennis to improve accuracy and handle the challenges presented by legged robots including balance.

Concurrent to our work, [20] uses reinforcement learning to train a humanoid robot to play table tennis. In contrast to their reliance on using human motion capture data to bootstrap the stroke strategies, we demonstrate a wide range of motions common in human players that automatically emerges from solving MPC. We also demonstrate that our control system can both handle incoming spin and generate it using swings such as loops and chops, which add top and back spin to the ball respectively.

III. System

In this section, we describe our system (shown in Fig. 2), including the perception, prediction, aiming, and control subsystems.

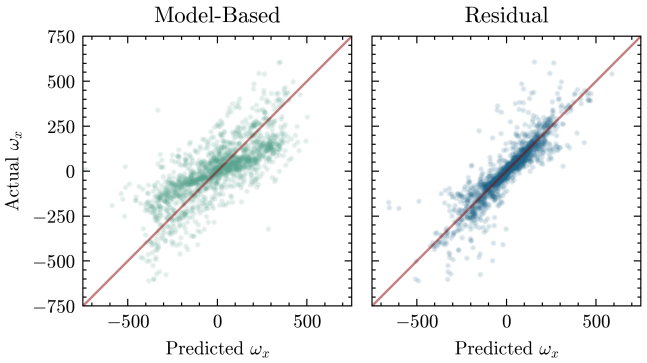


Fig. 3: Spin prediction performance using the model-based estimate on the left and the residual network on the right.

A. Ball Perception

The perception system is responsible for detecting and localizing the table tennis ball in 3D space. This task is accomplished through a stereo camera setup, and a high-speed detection pipeline.

1) System and Calibration: The system utilizes two Power over Ethernet (PoE) RGB cameras (Lucid Arena), each with a resolution of 1400x1080 pixels, capturing images at 165 frames per second (fps). The cameras are mounted on the ceiling at opposite ends of the table and angled downwards to view the entire playing surface seen in Figure 2. To calibrate the cameras, extrinsic parameters are fitted through key points at the table's corners by solving the Perspective-n-Point (PnP) problem [29].

2) Detection and Localization: Our ball detection pipeline is a two-stage process designed for high-speed performance, similar to the approach in [11]. First, a background subtraction algorithm identifies dynamic regions of the image which isolates moving objects from the static background. Next, the segmented moving regions are then composed into a smaller 480x480 pixel image and a fine-tuned YOLO convolutional neural network (CNN) [30], [31] is applied to identify the ball's 2D location in each frame. This approach significantly reduces computational load by avoiding the need to run the CNN on the full-resolution image. The YOLO network was refined on a custom dataset of these composite patches to handle the domain shift from standard datasets.

Once the ball is detected in both camera streams, the calibrated parameters are used to calculate the 3D position in the world coordinate frame through stereo triangulation.

3) Performance: With cameras capturing images at 165 fps (an interval of 6 ms per frame), the entire detection process is completed in approximately 4 ms (2.5 ms for background subtraction and 1.5 ms for CNN inference). This low latency ensures that a detection result is available well before the next frame is captured.

The system position accuracy was evaluated against a Vicon motion capture system resulting in a mean error of less than $1\,\mathrm{cm}$ in all three dimensions, far less than the $4\,\mathrm{cm}$ diameter of the ball.

B. Ball Trajectory Prediction and Spin Estimation

To predict when and where the robot must strike, we estimate the ball's linear ($\mathbf{v}$) and angular (ω) velocity from position measurements using methods proposed by Tebbe et al. [21]. This process includes fitting second-order polynomial coefficients $\mathbf{a}^{(k)} = [a_{x,k}, a_{y,k}, a_{z,k}]^\intercal$ to a history of ball positions where k is the coefficient order. We sample the derivatives of the polynomials to estimate $\mathbf{v}$ while performing a least-squares fit for ω using a discrete approximation of the ball dynamics in (1).

$$\frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{\Delta t_i} \approx -C_D \|\mathbf{v}_i\| \mathbf{v}_i + C_M(\boldsymbol{\omega} \times \mathbf{v}_i) - \mathbf{g} \tag{1}$$

where C_D and C_M represent the lumped coefficients of the drag and Magnus effects respectively. These parameters were fit using ground truth data from a Spinsight Elite ball tracking system alongside custom marked balls.

This technique provides a spin estimate, but with insufficient accuracy for competitive striking due to unmodeled effects. We extend this estimation system's performance using a learned residual network with inputs $[\omega^\intercal, \mathbf{a}^{(2)\intercal}, \mathbf{a}^{(1)\intercal}]^\intercal$ which include the original estimate and polynomial coefficients indicating curvature. The corrected spin prediction comes from a multilayer perceptron (MLP) with three hidden layers of 128 each and ReLU activations, followed by a linear output layer. This MLP is trained using 650 unique ball trajectories with varying spin along with Z-axis rotations for added data augmentation. Using R^2 as a performance metric, our learned residual improved the purely model-based approach from a 0.42 to 0.70 as seen in Fig. 3.

Using our state estimate, we integrate the ball trajectory using the same dynamics from (1) along with the table rebound model from Nonomura et al. [23]. The integration stops when the ball reaches a fixed strike plane located $0.5\,\mathrm{m}$ in front of the quadruped base. The terminal ball state is then reported to the aiming controller to choose a paddle state.

C. Strike Aiming

Using the anticipated ball state provided from the prediction system, we must find a contact paddle state that returns the ball to the other side of the table. This paddle state is described using $\mathbf{p}_{\text{des}}$, $\mathbf{v}_{\text{des}}$, and $\mathbf{n}_{\text{des}}$, the paddle position, velocity, and face normal vector respectively.

To choose these parameters, we designed a high-level aiming controller that takes in the desired landing position $\mathbf{p}_{\text{land}}$, spin ω^+, and landing time t_{land} and produces the desired paddle state. This problem is formulated as a constrained optimization in (2a)-(2e).

$$\min_{\mathbf{r}_b, \dot{\mathbf{r}}_b, \mathbf{n}_{\text{des}}, \mathbf{v}_{\text{des}}} \quad W_v \|\mathbf{v}_{\text{des}}\|_2^2 \tag{2a}$$

$$\text{s.t.} \quad \mathbf{r}_b[0] = \mathbf{p}_{\text{des}} \tag{2b}$$

$$\begin{bmatrix} \dot{\mathbf{r}}_b[0] \\ \boldsymbol{\omega}^+ \end{bmatrix} = f_{\text{contact}}(\dot{\mathbf{r}}_b^-, \mathbf{v}_{\text{des}}, \mathbf{n}_{\text{des}}, \boldsymbol{\omega}^-) \tag{2c}$$

$$\mathbf{x}_b[n+1] = \mathbf{x}_b[n] + f_{\text{aero}}(\mathbf{x}_b[n])\Delta t \ \forall n \tag{2d}$$

$$\mathbf{r}_b[N-1] = \mathbf{p}_{\text{land}} \tag{2e}$$

where $\mathbf{r}_\text{b}$ and $\dot{\mathbf{r}}_\text{b} \in \mathbb{R}^{3\times N}$ are the ball positions and velocities post collision and N is the number of trajectory nodes. f_contact represents the paddle and ball contact dynamics implemented from [23] while f_aero corresponds to the discrete aerodynamics in (1). Finally, $\mathbf{x}_\text{b} = [\mathbf{r}_\text{b}^\intercal, \dot{\mathbf{r}}_\text{b}^\intercal, \boldsymbol{\omega}^{+\intercal}]^\intercal$ signifies the full ball state for simplified notation.

We solve this optimization problem in real time using a custom Sequential Quadratic Programming (SQP) solver. We utilized OSQP [32] for the local QP solver and perform four SQP iterations before using the solution. All optimization formulations in this paper were generated using CasADi for fast function evaluation [33].

D. Whole Body Model Predictive Control

To achieve the desired paddle state from the aiming controller and prediction system, we designed a model-based kinematic planner coupled with a whole-body controller that generates dynamically feasible swings.

Generating a swinging motion that can adapt to strike changes and plan for a return requires constraints that start at the end of the planning horizon and move to the beginning as the motion progresses. Typical discrete-time formulations can handle constraints like these in locomotion controllers [34], [35], but struggle with our more rigid end-effector constraints. We propose a continuous-time alternative to address the limitations of a traditional discrete-time MPC.

1) Continuous-Time Trajectory Optimization: We present the following continuous-time formulation that represents each joint trajectory as a Bezier curve with only a single strike constraint that can be placed anywhere along the swing plan. Equations (3a)-(3h) showcase this optimization formulation with $\mathbf{q}_\text{c} \in \mathbb{R}^{24\times 8}$ representing evenly spaced Bezier curve control points for each of the 24 DoFs. These are evaluated with Bezier function $\mathcal{B}$ at a given time.

$$\min_{\mathbf{q}_\text{c},\ddot{\mathbf{q}}_\text{c},\mathbf{q}_\text{s},\dot{\mathbf{q}}_\text{s}} \quad W_a f_a(\ddot{\mathbf{q}}_\text{c}) + W_r f_r(\mathbf{q}_\text{rest}, \mathbf{q}_\text{c}) \tag{3a}$$

$$\text{s.t.} \quad \mathcal{B}(\mathbf{q}_\text{c}, 0) = \mathbf{q}_0 \tag{3b}$$

$$\mathcal{B}'(\mathbf{q}_\text{c}, 0) = \dot{\mathbf{q}}_0 \tag{3c}$$

$$\mathbf{q}_\text{min} \leq \mathcal{B}(\mathbf{q}_\text{c}, t_i) \leq \mathbf{q}_\text{max}\ \forall t_i \in (0, t_f] \tag{3d}$$

$$\mathcal{K}_f(\mathcal{B}(\mathbf{q}_\text{c}, t_i)) = \mathcal{K}_f(\mathbf{q}_0)\ \forall t_i \in (0, t_f] \tag{3e}$$

$$(\mathcal{K}_p(\mathbf{q}_\text{s}) - \mathbf{p}_\text{des}) \cdot \theta = 0 \tag{3f}$$

$$(\mathbf{J}_p(\mathbf{q}_\text{s})\dot{\mathbf{q}}_\text{s} - \mathbf{v}_\text{des}) \cdot \theta = 0 \tag{3g}$$

$$\|\mathcal{K}_n(\mathbf{q}_\text{s}) - \mathbf{n}_\text{des}\|_2^2 \cdot \theta \leq \epsilon_n \tag{3h}$$

Equations (3b) and (3c) ensure the planned trajectory starts at the current state of the robot while the joint limits and foot constraints in (3d) and (3e) are enforced by sampling points along the Bezier curves. Finally (3f)-(3h) drive the end effector to match the desired paddle strike conditions at the strike time t_s. Here, $\mathcal{K}_p$ and $\mathcal{K}_n$ are the forward kinematics for the center of the paddle and a point above the paddle face respectively. Phase parameter θ enables and disables these strike constraints to switch between swinging and return/rest modes of the controller.

To keep the end effector and other kinematic constraints decoupled, we add slack decision variables $\mathbf{q}_\text{s}$ and $\dot{\mathbf{q}}_\text{s} \in \mathbb{R}^{24}$ which are the joint and body state of the robot at strike. Equations (4a) and (4b) constrain these variables to lie along the planned Bezier curves at t_s.

$$\mathcal{B}(\mathbf{q}_\text{c}, t_s) = \mathbf{q}_\text{s} \tag{4a}$$

$$\mathcal{B}'(\mathbf{q}_\text{c}, t_s) = \dot{\mathbf{q}}_\text{s} \tag{4b}$$

To shape the swing, cost function f_a limits the robot's acceleration while f_r keeps its position close $\mathbf{q}_\text{rest}$, the rest stance. Because of the low distortion between the Bezier control points and the curve they parametrize, we can formulate these functions simply using the decision variables to keep the problem quadratic like in equation (5).

$$f_r(\mathbf{q}_\text{rest}, \mathbf{q}_\text{c}) = \sum_{n=0}^{N_c} (\mathbf{q}_\text{c}[n] - \mathbf{q}_\text{rest})^\intercal \mathbf{W}_\text{j} (\mathbf{q}_\text{c}[n] - \mathbf{q}_\text{rest}) \tag{5}$$

where $\mathbf{W}_\text{j} \in \mathbb{R}^{24\times 24}$ is a diagonal weighting matrix on each joint. Although indirect, this cost function keeps all control points close to the rest position $\mathbf{q}_\text{rest}$ which regularizes $\mathbf{q}_\text{c}$ without the need of computing any curve points. Similarly, we can minimize an acceleration proxy through f_a in equation (6a) alongside slack variable constraint (6b).

$$f_a(\ddot{\mathbf{q}}_\text{c}) = \sum_{n=0}^{N_c-2} \ddot{\mathbf{q}}_\text{c}[n]^\intercal \mathbf{W}_\text{j} \ddot{\mathbf{q}}_\text{c}[n] \tag{6a}$$

$$\ddot{\mathbf{q}}_\text{c}[n] = \mathbf{q}_\text{c}[n+2] - 2\mathbf{q}_\text{c}[n+1] + \mathbf{q}_\text{c}[n] \tag{6b}$$

Like the strike constraints, slack variables $\ddot{\mathbf{q}}_\text{c}$ are introduced here to keep the cost quadratic and remove any coupling terms between control points.

Together, both f_a and f_r create a simple convex cost function that regularizes the swinging motion. This cost has the added benefit of shaping the trajectory back to the rest position once the strike constraints are disabled, which allows this one problem to plan for swing and return motions.

We use the same SQP solver as the aiming system to solve this swing optimization problem. Because our cost is already quadratic, this solver only handles the constraint nonlinearity through multiple solves. During execution, we solve five SQP iterations on a warm-started solution which allows the controller to run at up to 200 Hz.

2) Whole Body Controller: The formulation presented above provides kinematically feasible trajectories from the current state to the strike conditions, but still require torques to realize them on hardware. Using a whole-body controller, we can solve for these torques $\mathbf{u}$ given dynamics and contact constraints with the goal of achieving a desired acceleration $\ddot{\mathbf{q}}_\text{des}$. This optimization is formulated in (7a)-(7c).

$$\min_{\ddot{\mathbf{q}},\mathbf{u},\boldsymbol{\lambda}} \quad \|\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_\text{des}\|^2 \tag{7a}$$

$$\text{s.t.} \quad \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau}_\text{g} + \mathbf{S}\mathbf{u} + \mathbf{J}^\intercal(\mathbf{q})\boldsymbol{\lambda} \tag{7b}$$

$$\boldsymbol{\lambda} \in \mathcal{FC}(\mu) \tag{7c}$$

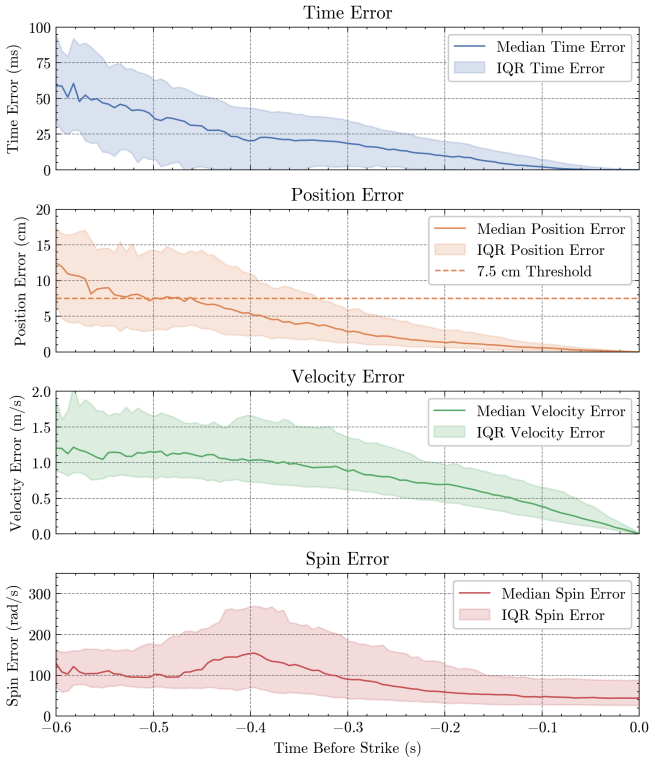


Fig. 4: Prediction output errors through ball flight with median and inter-quartile range (IQR). The $7.5\,\mathrm{cm}$ position threshold corresponds to the paddle radius, which is the minimum accuracy required to strike the ball.

where $\mathbf{M}$, $\mathbf{C}$, and $\boldsymbol{\tau}_g$ are the mass matrix, Coriolis terms, and gravity terms respectively of the general robotic manipulator equations and the $\mathbf{S}$ matrix maps control inputs to the DoFs. The ground reaction forces are solved for and denoted by $\boldsymbol{\lambda} \in \mathbb{R}^{12}$ while $\mathbf{J}$ is the Jacobian of the robot feet with respect to $\mathbf{q}$. To keep the constraints linear, the friction cone $\mathcal{FC}$ is approximated using a pyramidal approach. Together, this problem is a simple Quadratic Program and easily solved with off-the-shelf solvers, in this case OSQP [32].

With this combination of kinematic MPC planning and the dynamic whole-body controller, we are able to execute dynamic swings on hardware. Four swings are shown in Fig. 1, each with the strike state shown along with the swing motion.

IV. Evaluations

In this section, we evaluate the performance of our spin estimator and the MPC formulation. We also evaluate the performance of the entire system on a physical Spot robot.

A. Spin Estimation and Prediction

Using 650 recorded ball trajectories, we evaluated the performance of the prediction module by comparing the estimated and true final ball state as it approached the strike plane. Fig. 4 shows these errors in timing, position, velocity, and spin.

The data indicates that our estimation error reduces as more measured positions become available. Notably, the spin estimation converges to under $55\,\mathrm{rad\,s^{-1}}$ within $150\,\mathrm{ms}$ of the strike, providing ample time for the swing controller to adapt.

B. Swing Controller

In this section, we evaluate the performance of our continuous-time MPC against a standard discrete-time formulation in simulation.

1) Discrete-Time Model Predictive Control: A more typical MPC formulation using discrete-time is defined in (8a)-(8k). In this case, the decision variables are $\mathbf{q}_d$, $\dot{\mathbf{q}}_d \in \mathbb{R}^{24\times N_n}$, and $\ddot{\mathbf{q}}_d \in \mathbb{R}^{24\times(N_n-1)}$ corresponding to discrete nodes for joint positions, velocities, and accelerations respectively with N_n being the number of these nodes. The strike constraints are handled by duplicating them at every state and enabling/disabling them based on which node is closest to the strike time t_s using phase parameter ϕ_n.

$$\min_{\mathbf{q}_d,\dot{\mathbf{q}}_d,\ddot{\mathbf{q}}_d} \quad W_a f_a(\ddot{\mathbf{q}}_d) + W_r f_r(\mathbf{q}_{\text{rest}}, \mathbf{q}_d) \tag{8a}$$

$$\text{s.t.} \quad \forall n = 0, \ldots, N_n - 1 \tag{8b}$$

$$\mathbf{q}_d[0] = \mathbf{q}_0 \tag{8c}$$

$$\dot{\mathbf{q}}_d[0] = \dot{\mathbf{q}}_0 \tag{8d}$$

$$\dot{\mathbf{q}}_d[n+1] = \dot{\mathbf{q}}_d[n] + \ddot{\mathbf{q}}_d[n]\Delta t \tag{8e}$$

$$\mathbf{q}_d[n+1] = \mathbf{q}_d[n] + \dot{\mathbf{q}}_d[n+1]\Delta t \tag{8f}$$

$$\mathbf{q}_{\text{min}} \leq \mathbf{q}_d[n] \leq \mathbf{q}_{\text{max}} \tag{8g}$$

$$\mathcal{K}_f(\mathbf{q}_d[n]) = \mathcal{K}_f(\mathbf{q}_0) \tag{8h}$$

$$(\mathcal{K}_p(\mathbf{q}_d[n]) - \mathbf{p}_{\text{des}}) \cdot \phi_n = 0 \tag{8i}$$

$$(\mathbf{J}_p(\mathbf{q}_d[n])\dot{\mathbf{q}}_d[n] - \mathbf{v}_{\text{des}}) \cdot \phi_n = 0 \tag{8j}$$

$$\|\mathcal{K}_n(\mathbf{q}_d[n]) - \mathbf{n}_{\text{des}}\|_2^2 \cdot \phi_n \leq \epsilon_n \tag{8k}$$

Like the continuous-time formulation, (8c) and (8d) are the initial state constraints while the nodes are connected through semi-implicit Euler integration in (8e) and (8f). The joint limit and foot constraints are identical to our proposed formulation but evaluated at all nodes in (8g) and (8h). Similarly, the strike constraints match but with the new phase parameter scheme in (8i)-(8k). The same cost functions f_a and f_r are used as well but with their explicit node accelerations and positions rather than Bezier control point approximations. This problem is solved using the same number of SQP iterations as the continuous-time problem with cost weights tuned for similar motions.

2) Simulation Sweeps: To demonstrate the improvements of our proposed continuous-time MPC over the typical discrete-time version, we performed a sweep of three swing types across a dense grid of positions in the strike plane summing to 30,000 samples. Each point was generated by simulating one of the two MPC planners and the whole-body controller at $100\,\mathrm{Hz}$ while recording various metrics. Throughout each swing, Gaussian noise was added to $\mathbf{p}_{\text{des}}$, $\mathbf{v}_{\text{des}}$, and $\mathbf{n}_{\text{des}}$ to simulate how the changing prediction

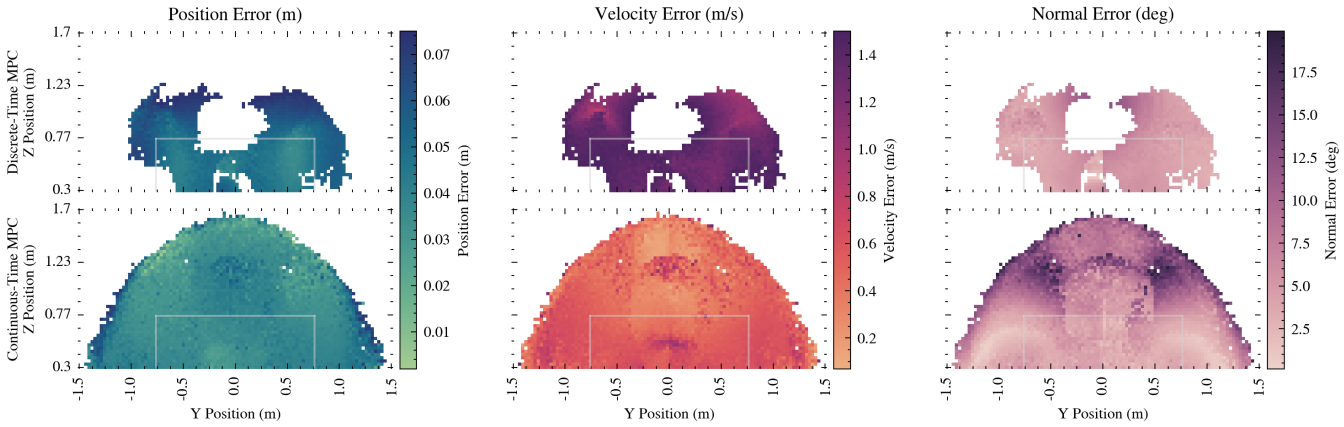


Fig. 5: Position, velocity and orientation error of the paddle at strike of chop swings at an array of positions for both discrete-time and continuous-time controllers. The Y and Z positions displayed are in the strike plane and the gray lines represent the table location.

updates the strike conditions. At the strike time, we then evaluated how closely the controller reaches the desired paddle state, logging position, velocity, and orientation errors.

For analysis, we define successful strike criteria as being within $7.5\,\mathrm{cm}$, $1.5\,\mathrm{m\,s^{-1}}$, and $20°$ of the desired targets. Using these thresholds, Table I displays the compared tracking performance of each controller for all three swing types with the continuous-time MPC outperforming the discrete version in all metrics other than orientation tracking for loops and chops. We plot the chop swing strike errors across the workspace for both controllers as well in Fig. 5 to show the increased workspace of the continuous-time controller along with its more consistent striking accuracy.

We further investigate this performance discrepancy by plotting the strike constraint violation for each formulation's solutions over simulated time in Fig. 6. The discrete controller's errors show more volatility that explodes near the end of the trajectory. This instability is a byproduct of the discrete jumps of the strike constraint selection parameter ϕ_n. Because a new constraint is enabled when ϕ_n changes, the solver requires more iterations before these strike constraints converge. This rapid increase in constraint violation near the time of strike requires termination of the MPC approximately $100\,\mathrm{ms}$ early to maintain decent tracking at the cost of late adaptability. Conversely, the continuous-time controller has a more consistent set of strike errors which do not increase at the tail end of the trajectory. This attribute leads to superior tracking of the desired paddle state and more resilience to late perturbations than the traditional discrete-time formulation.

When executing these controllers on hardware, faster solve times are advantageous to limit delay and improve responsiveness. Therefore, the continuous-time formulation wins again with the ability to solve at up to $200\,\mathrm{Hz}$ versus only $50\,\mathrm{Hz}$ for the discrete MPC due to a significant difference in problem size. This is because the discrete-time formulation needed $N_n = 20$ to reach minimum performance. These solve times were measured on an Intel i9-14900K CPU.

		Continuous-Time	Discrete-Time
Position Error (cm)	Loop	**3.90 ± 0.61**	5.50 ± 0.65
	Drive	**4.90 ± 0.67**	6.76 ± 0.67
	Chop	**3.64 ± 0.84**	5.53 ± 1.03
Velocity Error (m/s)	Loop	**0.47 ± 0.19**	0.99 ± 0.18
	Drive	**0.42 ± 0.14**	1.28 ± 0.14
	Chop	**0.50 ± 0.15**	1.33 ± 0.15
Orientation Error (°)	Loop	8.16 ± 2.67	**7.04 ± 2.55**
	Drive	**7.53 ± 2.47**	9.33 ± 1.56
	Chop	7.11 ± 4.19	**5.02 ± 1.53**

TABLE I: Simulated mean striking error and standard deviation of each swing type for both MPC formulations.

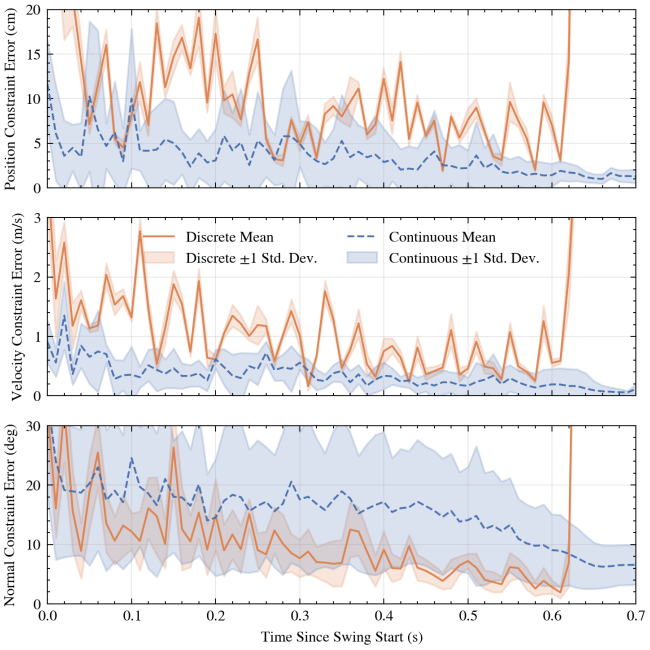


Fig. 6: Constraint error over time for both the continuous-time and discrete-time controllers.

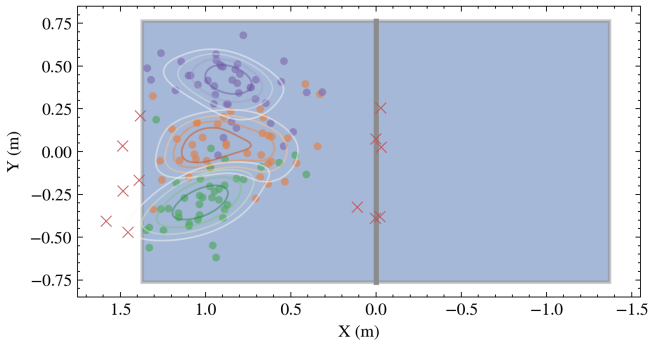


Fig. 7: Landing locations when aiming at three different table locations with misses indicated by an "x". Shots originated from Spot on the right of the plot.

	Spin Detection		
ω_y^-	None	Least Squares	Residual
220 rad/s	0%	28%	**44%**
100 rad/s	12%	52%	**72%**
0 rad/s	72%	68%	**84%**
-125 rad/s	12%	44%	**88%**
-280 rad/s	40%	68%	**88%**
Mean	27.2%	52.0%	**75.2%**

TABLE II: Return rate for different incoming ball spin ω_y^- with different degrees of spin detection. Negative ω_y^- correspond to top spin while positive values indicate back spin.

C. Hardware Evaluation

To demonstrate the continuous-time MPC performance on hardware, we struck 132 balls and recorded their landing locations with three different $\mathbf{p}_{\text{land}}$ values. Figure 7 includes the recorded landing locations for each target marked as a return or miss. Over the 132 trials, 90.1% were returned with a clear aiming pattern given $\mathbf{p}_{\text{land}}$.

The state estimation and prediction system utilizes a residual network on top of a model-based estimator for $\boldsymbol{\omega}^-$. We tested the necessity for these components with a small ablation, which included no spin estimation, only model-based, and the full residual estimator. To choose the outgoing spin $\boldsymbol{\omega}^+$, we implemented a simple heuristic strategy used by players to return shots with the same spin as they are received. For each setup, we tested five different spin values shown in Table II and recorded the return rate over 25 trials.

This shows the need for the spin estimation and the improvement of return performance when the residual network is included. Overall, the full system was capable of handling a wide variety of ball spin with a mean return rate of 75.2%.

Since the aiming system is capable of solving for the exiting ball spin $\boldsymbol{\omega}^+$, we tested the system's accuracy in generating this spin. Table III includes the mean and standard deviation over 25 trials for four different target spin values.

Over the four targets, the system is capable of getting within 20 rad/s of the desired $\boldsymbol{\omega}^+$ showing its ability to generate diverse spin. Examples of back spin and top spin shots are shown in Fig. 8.

ω_y^+	Mean	Std. Dev.
200 rad/s	191.9 rad/s	16.5 rad/s
125 rad/s	118.1 rad/s	15.0 rad/s
-125 rad/s	-133.1 rad/s	11.5 rad/s
-200 rad/s	-182.6 rad/s	24.0 rad/s

TABLE III: Mean and standard deviation of ball spin added by Spot for different desired spin speeds. Here, positive values correspond to top spin and negative to back spin.

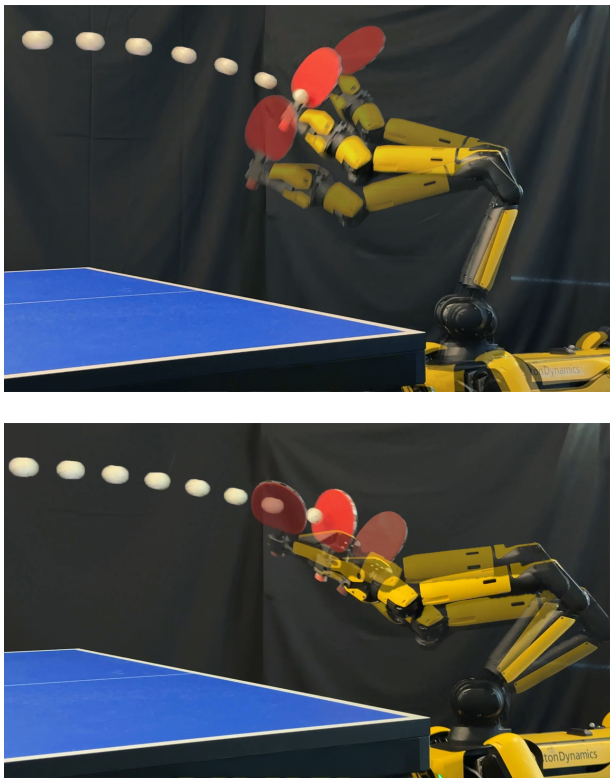


Fig. 8: Back spin (top) and top spin (bottom) hardware swing examples with the exiting ball trajectory shown.

D. Table Tennis Game Play

The quantitative evaluations above show how each sub-component operates well and that the integrated system is capable of handling spin as well as aiming. Qualitatively, our system also supports playing table tennis with a person and is able to sustain a rally of over 10 hits per side all while handling spin from the opponent.

V. Conclusions

We introduce a continuous-time whole-body model predictive controller which allows an arm-equipped quadrupedal robot to play table tennis. In simulation, we demonstrate how our new continuous-time MPC outperforms typical discrete-time formulations in strike performance and solve speed. In conjunction with an extended learning-based spin estimator, aiming planner, and prediction system, we showcase this new controller on hardware with a Spot quadruped robot.

This includes a hardware ablation indicating the effectiveness of our learned spin estimation strategy. Finally, the MPC demonstrates the ability to return arbitrary spin with a diverse set of emerging swing types.

Future work aims to address some of the limitations of the current controller framework. Most importantly, our system is incapable of taking steps, a key aspect of high-level table tennis game play. This is a core limitation of the continuous-time formulation due to the non-smooth nature of stepping trajectories. To overcome this, a hybrid approach combining a reinforcement learning-based method [14], [20] with our proposed MPC could enable the robot to perform the agile footwork essential in human table tennis, while still leveraging the MPC for efficient online swing planning. Other system-level extensions can be made including on-board perception, strike strategy, and extending the work to humanoids much like [20].

REFERENCES

[1] B. Yi, C. M. Kim, J. Kerr, G. Wu, R. Feng, A. Zhang, J. Kulhanek, H. Choi, Y. Ma, M. Tancik, and A. Kanazawa, "Viser: Imperative, web-based 3d visualization in python," 2025. [Online]. Available: https://arxiv.org/abs/2507.22885

[2] J. Łapszo, "The speed of sequential movements in table tennis studied under simulated conditions with respect to range, body involvement and direction," *International Journal of Table Tennis Sciences, 4*, vol. 5, pp. 19–28, 2002.

[3] R. Fullen. (2004) Mechanics of table tennis. [Online]. Available: https://protabletennis.net/book/export/html/210

[4] Spinsight. [Online]. Available: https://spinsight.com/insights/

[5] D. Nguyen, K. D. Cancio, and S. Kim, "High speed robotic table tennis swinging using lightweight hardware with model predictive control," *arXiv preprint arXiv:2505.01617*, 2025.

[6] J. Tebbe, L. Krauch, Y. Gao, and A. Zell, "Sample-efficient reinforcement learning in robotic table tennis," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 4171–4178.

[7] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters, "Learning to play table tennis from scratch using muscular robots," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3850–3860, 2022.

[8] D. B. D'Ambrosio, J. Abelian, S. Abeyruwan, M. Ahn, A. Bewley, J. Boyd, K. Choromanski, O. Cortes, E. Coumans, T. Ding *et al.*, "Robotic table tennis: A case study into a high speed learning system," *arXiv preprint arXiv:2309.03315*, 2023.

[9] C. Liu, Y. Hayakawa, and A. Nakashima, "Racket control and its experiments for robot playing table tennis," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2012, pp. 241–246.

[10] Y. Wang, Y. Luo, H. Zhang, W. Zhang, K. Dong, Q. He, Q. Zhang, E. Cheng, Z. Sun, and B. Song, "A table-tennis robot control strategy for returning high-speed spinning ball," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 3, pp. 2115–2124, 2023.

[11] Z. Zaidi, D. Martin, N. Belles, V. Zakharov, A. Krishna, K. M. Lee, P. Wagstaff, S. Naik, M. Sklar, S. Choi *et al.*, "Athletic mobile manipulator system for robotic wheelchair tennis," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2245–2252, 2023.

[12] F. Yang, Z. Shi, S. Ye, J. Qian, W. Wang, and D. Xuan, "Varsm: Versatile autonomous racquet sports machine," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, 2022, pp. 203–214.

[13] A. Krishna, Z. Zaidi, L. Chen, R. Paleja, E. Seraj, and M. Gombolay, "Utilizing human feedback for primitive optimization in wheelchair tennis," 2022. [Online]. Available: https://arxiv.org/abs/2212.14403

[14] Y. Ma, A. Cramariuc, F. Farshidian, and M. Hutter, "Learning coordinated badminton skills for legged manipulators," *Science Robotics*, vol. 10, no. 102, p. eadu3922, 2025.

[15] H. Wang, Z. Shi, C. Zhu, Y. Qiao, C. Zhang, F. Yang, P. Ren, L. Lu, and D. Xuan, "Integrating learning-based manipulation and physics-based locomotion for whole-body badminton robot control," *arXiv preprint arXiv:2504.17771*, 2025.

[16] J. Tebbe, Y. Gao, M. Sastre-Rienietz, and A. Zell, "A table tennis robot system using an industrial kuka robot arm," in *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40*. Springer, 2019, pp. 33–45.

[17] S. Gomez-Gonzalez, Y. Nemmour, B. Schölkopf, and J. Peters, "Reliable real-time ball tracking for robot table tennis," *Robotics*, vol. 8, no. 4, p. 90, 2019.

[18] Q. Xiao, Z. Wu, and M. Gombolay, "Learning dynamics of a ball with differentiable factor graph and roto-translational invariant representations," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 8826–8832.

[19] C. H. Lampert and J. Peters, "Real-time detection of colored objects in multiple camera streams with off-the-shelf hardware components," *Journal of Real-Time Image Processing*, vol. 7, no. 1, pp. 31–41, 2012.

[20] Z. Su, B. Zhang, N. Rahmanian, Y. Gao, Q. Liao, C. Regan, K. Sreenath, and S. S. Sastry, "Hitter: A humanoid table tennis robot via hierarchical planning and learning," *arXiv preprint arXiv:2508.21043*, 2025.

[21] J. Tebbe, L. Klamt, Y. Gao, and A. Zell, "Spin detection in robotic table tennis," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 9694–9700.

[22] Q. Xiao, Z. Zaidi, and M. Gombolay, "Multi-camera asynchronous ball localization and trajectory prediction with factor graphs and human poses," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 13 695–13 702.

[23] J. Nonomura, A. Nakashima, and Y. Hayakawa, "Analysis of effects of rebounds and aerodynamics for trajectory of table tennis ball," in *Proceedings of SICE Annual Conference 2010*, 2010, pp. 1567–1572.

[24] T. Gossard, J. Tebbe, A. Ziegler, and A. Zell, "Spindoe: A ball spin estimation method for table tennis robot," 2023.

[25] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[26] K. M. Lee, S. Ye, Q. Xiao, Z. Wu, Z. Zaidi, D. B. D'Ambrosio, P. R. Sanketi, and M. C. Gombolay, "Learning diverse robot striking motions with diffusion models and kinematically constrained gradient guidance," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 12 017–12 024.

[27] D. B. D'Ambrosio, S. W. Abeyruwan, L. Graesser, A. Iscen, H. B. Amor, A. Bewley, B. Reed, K. Reymann, L. Takayama, Y. Tassa *et al.*, "Achieving human level competitive robot table tennis," in *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2024.

[28] O. Koç, G. Maeda, and J. Peters, "Online optimal trajectory generation for robot table tennis," *Robotics and Autonomous Systems*, vol. 105, pp. 121–137, 2018.

[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, Jun. 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[30] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[31] A. B. et al., "hank-ai/darknet," https://github.com/hank-ai/darknet, may 26 2025. [Online]. Available: https://github.com/hank-ai/darknet

[32] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[33] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[34] S. H. Jeon, H. J. Lee, S. Hong, and S. Kim, "Residual mpc: Blending reinforcement learning with gpu-parallelized model predictive control," 2025. [Online]. Available: https://arxiv.org/abs/2510.12717

[35] C. Khazoom, S. Hong, M. Chignoli, E. Stanger-Jones, and S. Kim, "Tailoring solution accuracy for fast whole-body model predictive control of legged robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 074–11 081, 2024.