Let the Golf Fly

Songyuan Zhang
Department of Aeronautics and Astronautics
MIT
Cambridge, MA, USA
szhang21@mit.edu

Mingxin Yu
Department of Aeronautics and Astronautics
MIT
Cambridge, MA, USA
yumx35@mit.edu

***Abstract*—We consider the problem of controlling a robot to play golf, aiming to land the ball in the hole with only one hit. This is a hard task because of the precision required and the discontinuous dynamics of hitting. To solve the problem, we establish a simulator with physical engines, and divide the problem into three stages: pre-hitting, hitting, and post-hitting. In the pre-hitting stage, we use kinematic trajectory optimization to control the robot to the desired pre-hitting configuration. In the hitting stage, based on the desired initial velocity of the ball calculated via direct shooting in the post-hitting stage, we calculate the desired joint velocities of the arm via system identification and do velocity control to hit the ball. The experimental results show that our framework can reach about 19% success rate.**

***Index Terms*—direct-shooting, kinematic trajectory optimization, system identification**

I. Introduction

Golf is a sport in which the players use a set of clubs to hit the balls into holes with as few strokes as possible. It is a typically difficult game, requiring the players to be trained for years before they can finally play the game. There are a great number of things for a well-trained player to consider while playing, including the terrain of the golf terrain, the angle and speed of hitting, the speed and the direction of the wind, etc. Before hitting, the player needs to have knowledge of the course, getting to know where a hill may exist that can block the ball, where some long grasses can slow down the ball, and where a pool can swallow the ball. After that, the player is supposed to come up with the trajectory of the ball, taking the wind and the rotation of the ball into consideration, which determines the desired hitting angle and the force used. Finally, the player strikes the ball with the desired angle and force, observes the landing position, and does the steps more times until the ball lands in the hole.

Playing golf is a complex and difficult problem for humans, but is it the same for robots? Can we control a robot arm and let it play golf? To begin with, we simplify the multi-stroke golf problem to single stroke and only consider striking action, as shown in Figure 1. In this project, we present a framework to control a robot to play golf. Because of the discontinuous dynamics of hitting, we divide the problem into three parts: pre-hitting, hitting, and post-hitting. First, in the post-hitting stage, we use trajectory optimization to calculate the desired initial velocity of the golf ball such that the ball can land in the hole. Then, in the pre-hitting stage, we control the robot to the desired configuration for hitting the ball. Finally, in the hitting stage, we control the robot to hit the ball so that the ball flies with the desired initial velocity. The key ideas of the project are summarized as follows:

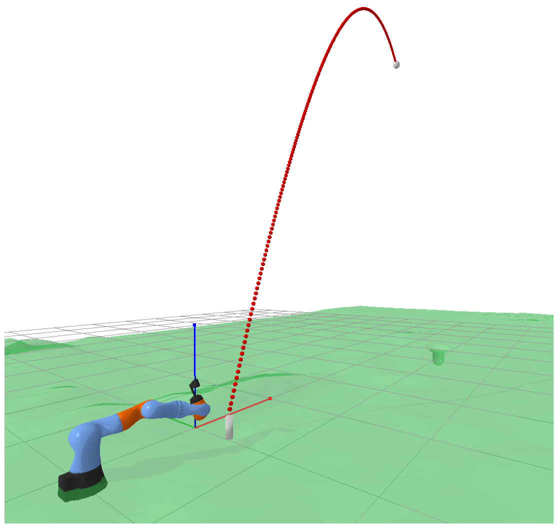
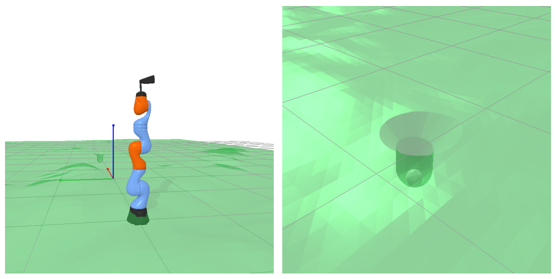


Fig. 1. Demonstration of the simplified single-stroke problem.

- We establish a full-stack system to simulate the robot playing golf using physical engines.
- We apply trajectory optimization to calculate the desired initial velocity of the golf ball.
- We use frame transformation to calculate the desired position and orientation of the club, and use kinematic trajectory optimization to control the robot to the desired pre-hitting configuration.
- We use linear regression and supervised learning to do system identification to establish the mapping from the initial velocity of the golf ball to the joint velocities of the robot arm, and predict the desired joint velocities based on the desired initial velocity of the ball.
- Experimental results show that our algorithm can reach 19% success rate. We also discuss the possible failure reasons and future directions.

II. Related works

A few works [1]–[5] have been working on playing ping-pong with robot arms. These methods vary a lot regarding



(a) (b)

Fig. 2. (a) Kuka-iiwa robot with golf club (b) Terrain with hole

the exploitation of physics models. [2] utilizes model-free reinforcement learning to return a ping-pong ball, while [1] utilizes inverse reinforcement learning to train a Rethink Robotics Sawyer for a particular table tennis strike (up-spin) in the real world. [3] propose a hybrid policy gradient approach by designing a white-box simulation environment and combine with reinforcement learning to strike a ping-pong ball. On the other end of the spectrum, [4] uses pure physics modeling with mirror law and PD controller to hit a ping-pong ball. However, playing golf with one strike is quite different. It requires accurate control of the ball's initial velocity and involves intractable contact dynamics. We must combine the robustness of physics modeling with the effectiveness of data-driven methods.

III. Method

A. Problem Definition

We consider the task for a robot to strike a golf ball into a hole. We assume that the robot has complete knowledge of the ball's rest pose, the location of hole and the dynamics when the ball is flying in air. However, the dynamics of hitting process and of the ball rolling on terrain remain unknown. The task is successful if the ball can finally rest in the hole with only one strike.

B. Simulation Environment

We set up our environment illustrated in Figure 2 using the PyBullet simulator [6]. The simulation environment has a Kuka-iiwa robot, whose gripper is substituted with a fixed golf club, as shown in Figure 2(a). A mobile base is attached on the Kuka robot, enabling the robot to move freely along x and y directions. The collision shape of the golf ball is simplified as a sphere with radius $r = 6\,\text{cm}$. The terrain is described by a concave mesh, whose upper surface is uneven. The shape of the hole is a cone stacked on a cylinder, as shown in Figure 2(b). In our system, air resistance is taken into account and modeled to be proportional to ball velocity, based on Stokes' Law [7] - a spherical object with rather small Reynolds numbers in a viscous fluid.

C. Pipeline

There are three stages of motion for one-stroke golf: before hitting, hitting, and after hitting, based on dynamics discontinuity. For the one-stroke golf task we designed, no control input can be applied to the golf ball after hitting. Therefore, it is essential to enable the ball to acquire a certain velocity accurately after hitting. By assuming perfect knowledge of the ball's rest pose, the location of the hole, and the ball's dynamics of flying in the air, we can solve the ball's desired initial velocity via forward simulation and trajectory optimization.

The major challenge for the task is the non-differentiable gray-box dynamics of the hitting process. Though the process of collision can be accurately modeled by Newton's law of motion, discrete collision detection and numerical integration for a high-speed object in the simulation environment will introduce deviation and instability. It also makes no sense to enforce the robot to solve the exact contact model used in the simulator. To address the challenge, we propose a hybrid method exploiting both the knowledge of collision and the effectiveness of data-driven methods to control the robot arm.

1) Post-hitting - Trajectory Optimization: The only thing that we can control in the post-hitting stage is the initial velocity of the golf ball, which is determined by the hitting velocity of the golf club. After the hitting, the ball is only forced by the gravity $\mathbf{g}$ and the air resistance $-\alpha\mathbf{v}$, where α is the coefficient of the Stokes' Law, so the trajectory of the ball will be uniquely determined by the initial velocity. Therefore, we consider the initial velocity v of the golf ball as the decision variable, and formalize this problem as a direct-shooting problem:

$$\begin{aligned}
\min_{v[0],dt} \quad & dt && \text{(1a)}\\
\text{subject to} \quad & v_{\min} \le v[0] \le v_{\max} && \text{(1b)}\\
& x_{\text{hole}} - \delta r \le x[N] \le x_{\text{hole}} + \delta r && \text{(1c)}\\
& y_{\text{hole}} - \delta r \le y[N] \le y_{\text{hole}} + \delta r && \text{(1d)}\\
& z_{\text{hole}} - \delta z \le z[N] \le z_{\text{hole}} + \delta z && \text{(1e)}\\
& v_z[N] \le 0 && \text{(1f)}\\
& \sqrt{\frac{v_x[N]^2 + v_y[N]^2}{v_z[N]^2}} \le \tan\theta_{\max} && \text{(1g)}\\
& dt \ge 0 && \text{(1h)}\\
& \mathbf{x}[n+1] = f(\mathbf{x}[n], dt), \quad \forall n \in [0, N-1] && \text{(1i)}
\end{aligned}$$

where N is the pre-defined simulation steps and dt is the simulation time interval, and objective (1a) means that we want the ball to reach the hole position as fast as possible. Equation (1b) constrains the initial velocity of the golf ball. x_{hole}, y_{hole}, and z_{hole} are the x, y, z position of the hole. δr is the radius difference of the ball and the hole, and δz is the distance tolerance in z-axis. Therefore, the constraints (1c), (1d), (1e) mean that the final position of the ball should land in the hole. Equation (1f) constrains the final velocity of the ball. Equation (1g) constrains the landing angle of the ball (the angle between the velocity and the vertical direction) because a large landing angle can cause the ball to bump out the hole. Finally, Equation (1i) is the aerodynamics of the ball, with

$\mathbf{x} = [x, y, z, v_x, v_y, v_z]^\top$ as the state vector. The aerodynamics is given by:

$$f(\mathbf{x}[n], dt) = \mathbf{x}[n] + \begin{bmatrix} v_x[n] \\ v_y[n] \\ v_z[n] \\ -\alpha v_x[n] \\ -\alpha v_y[n] \\ -\alpha v_z[n] \end{bmatrix} \quad (2)$$

Note that we can add more constraints to the problem, such as avoiding obstacles, but we omit that part because of the time limitation.

This is a standard direct-shooting problem and we can solve it with many solvers including CasADi [8]. If the ball is endowed with the solved initial velocity, it is 100% guaranteed to land in the hole in our experiments.

2) Pre-hitting - Velocity-controlled Preparation: We want the ball to be hit such that its initial velocity is the desired one. However, this is non-trivial since the hitting is not continuous or differentiable (e.g., compared with the throwing problem). Therefore, we separate the hitting problem into two stages to solve: preparation and hitting. In the preparation stage, we control the robot to the desired configuration such that the club contacts the ball and the normal of contact point aligns with the desired initial velocity of the ball. In this way, we can simplify the hitting stage.

In the pre-hitting stage, our goal is to find the configuration ${}^{W}X^{L}$, i.e. the transformation from the world frame to the link frame (the 'club-link' of the robot arm). To find this, we first establish the transformation from the club frame to the link frame ${}^{C}X^{L} = {}^{L}X^{C^{-1}}$, where the club frame is the frame attached to the club object. The club is transformed and scaled with some given value when attaching to the arm, so ${}^{L}X^{C}$ is known. Next, we construct the hitting frame by finding a desired hitting point on the club, aligning $+z$ direction of the hitting frame with the normal of the point(as shown in Figure 3(a)) , and construct the transformation from the club frame to the hitting frame ${}^{C}X^{H}$. Then, we define the desired transformation from the world frame to the hitting frame ${}^{W}X^{H}$, by aligning the z-axis of the hitting with the initial velocity of the ball and the y-axis aligns with the y-axis of the world frame. Finally, we find the desired configuration

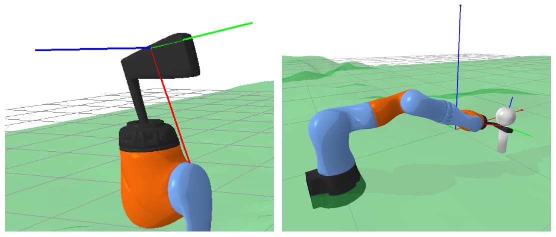

(a) (b)

Fig. 3. (a) Definition of hitting frame (b) The pre-hitting configuration with hitting frame and world frame

of the link frame (shown in Figure 3(b)) in the world frame by

$${}^{W}X^{L} = {}^{W}X^{H} \left({}^{C}X^{H}\right)^{-1} \left({}^{L}X^{C}\right)^{-1} \quad (3)$$

After finding the desired configuration ${}^{W}X^{L}$, we can do optimization to get the desired joint positions q_{goal}. We solve the problem

$$\min_{q} \quad \|J(q_{\text{goal}})q_{\text{goal}} - x_{\text{club}}\| \quad (4)$$

where $J(q)$ is the forward Jacobian of the arm, and x_{club} is the position and orientation of the link frame corresponding with ${}^{W}X^{L}$.

The next step is to control the robot to reach the desired joint position q. We apply kinematic trajectory optimization to solve this problem, which is formulated as

$$\begin{aligned} \min_{\alpha, T} \quad & T, \\ \text{subject to} \quad & q_\alpha(0) = q_{\text{start}} \\ & q_\alpha(T) = q_{\text{goal}} \\ & |\dot{q}_\alpha(t)| \le v_{max}, \quad \forall t \\ & \text{no collision}, \quad \forall t \end{aligned} \quad (5)$$

where T is the time taken from the starting joint positions q_{start} to the desired joint positions q_{goal}, and 'no collision' means that the robot should avoid self-collision when planning a trajectory. By solving the kinematic trajectory optimization problem, we can get a series of intermediate joint positions $q_\alpha(t)$. We then do position control to control the robot to follow the intermediate joint positions and reach the goal configuration.

3) Hitting: System Identification and Velocity Control: After setting the robot to the pre-hitting configuration, we need to control the robot to hit the ball such that the ball will fly with the desired initial velocity. However, this is non-trivial because of the difficulty of modeling the collision of club and the ball. To solve the problem, we propose to first do system identification to establish the mapping from the initial velocity of the golf ball to the joint velocities of the robot arm, and then do velocity control on the joints to hit the ball with the mapped joint velocities. We observe that the rotation of the last joint of the arm can provide normal velocity to the ball, while the rotation of the second last joint can provide both normal and tangential velocity. Therefore, to simplify the problem, instead of controlling all the joints, we only control the last two joints while fixing the others. We believe that the remaining two joints are enough for hitting the ball to the hole.

We generate data for system identification by randomly sampling velocities $\dot{q}_1$ and $\dot{q}_2$ of the last and the second last joint with Gaussian distribution and record the initial velocities of the golf ball. We determine the mean and the variance of the Gaussian distribution by manually control the joints to hit the ball such that the ball lands close to the hole. With the data, we first filter out the outliers with RANSAC [9] algorithm, and then adopt two different system identification methods: linear regression and supervised learning. For linear regression, we fit

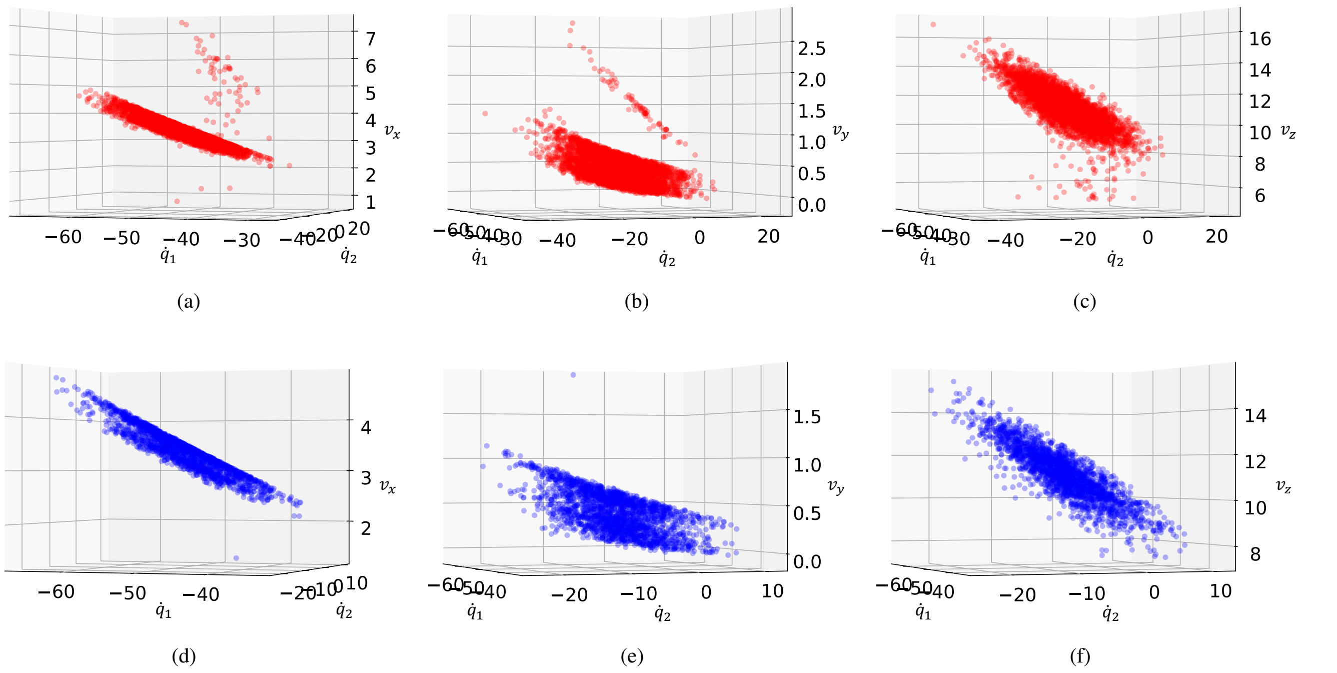


Fig. 4. Data visualization. (a)-(c) are all data points and (d)-(f) are inliers. We can observe that the major data points follow a strong linear correlation. $\dot{q}_1$ and $\dot{q}_2$ are in $\mathrm{rad/s}$, $v_{x,y,z}$ are in $\mathrm{m/s}$.

two linear models to map the initial velocity of the ball to the joint velocities $\dot{q}_1$ and $\dot{q}_2$ respectively, and for the supervised learning, we train a neural network to learn the mapping. We will compare the efficacy of the two algorithms in Section IV.

Finally, with the fitted system identification model, we can predict the desired velocities $\dot{q}_1$ and $\dot{q}_2$ of the last two joints. Then we do velocity control and input the desired the velocity, and the arm will hit the ball to make it fly.

IV. Result

A. Experiment Setup

The hole position is randomly sampled from a $3\,\mathrm{m} \times 3\,\mathrm{m}$ court. The center of the court is approximately $8\,\mathrm{m}$ away from the ball's initial position. α, the coefficient of air resistence is set as 0.1.

To make it easier for the ball to acquire high speed after striking, we set the ball's restitution to be 0.98 and contact stiffness as 100. To avoid the ball bouncing out of the hole, the restitution of the terrain is set to 0.02.

B. Post-hitting Stage

The trajectory optimization problem defined in Equation (1a) is solved with CasADi [8]. We set $\theta_{\text{max}} = 13.5°$, $\delta r = 4\,\mathrm{cm}$ and $\delta z = 0.1\,\mathrm{cm}$. CasADi is able to find a solution within 50 iterations and the residual error is less than 10^{-8}.

As discussed in Section III-C, the task's success relies heavily on accurately calculating the ball's desired initial velocity. We test the robustness of success with respect to the ball's initial velocity. We observe that velocity with a deviation larger than $0.5\,\mathrm{m/s}$ will never hit the hole in our setting. As a result, we have to control the initial velocity of the ball very precisely in order to hit the hole. This confirms that one-strike golf is indeed a difficult control problem and requires comprehensive controller design.

C. Hitting

For the hitting process, we need to recover a mapping from (v_x, v_y, v_z) back to $(\dot{q}_1, \dot{q}_2)$. First, we collect $6.5\,\mathrm{k}$ data points of $(\dot{q}_1, \dot{q}_2)$ and (v_x, v_y, v_z). Shown in Figure 4, data points show a strong linear correlation. This is because the mapped joint velocity points only occupy a small portion of the entire velocity space. In this case, a linear approximation is indeed a natural option for local approximation. We also noticed the existence of outliers. Digging deeper, we find that the major source for outliers is that the golf club may be unable to strike the ball under some scenarios. We will discuss this in detail in the next section.

This strong correlation inspires us to use RANSAC [9] as an outlier detector and as a linear regressor. We randomly generate $10\,\mathrm{k}$ trials for RANSAC to get a linear model. Attempting to model the mapping more accurately, we also try training a two-layer MLP with 8 hidden neurons each for 40960 iterations. Since our goal is to map the velocities that can succeed, we remove the outliers using RANSAC from the dataset in advance.

D. Whole system

We evaluate two different system identification methods in the simulation environment by randomly sampling the hole position on the court. We repeat this experiment 100 times

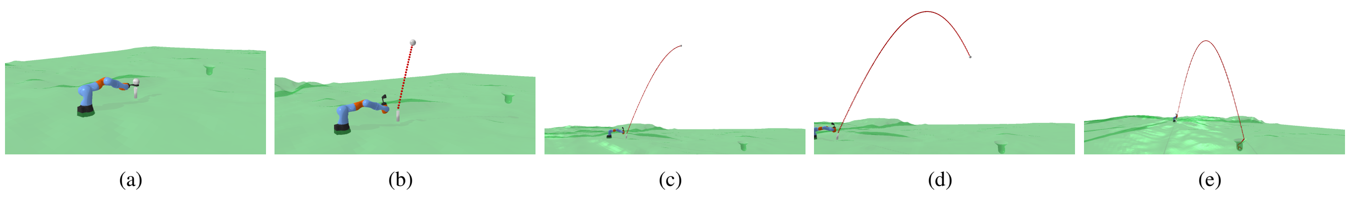


Fig. 5. Video clips of a successful video. (a) for pre-hitting, (b)-(e) for post-hitting.

respectively and the success rates for the two methods are $19/100$ for RANSAC and $17/100$ for RANSAC+MLP. This suggests that when linear approximation is good enough, it might be worse to do further fitting and that don't be superstitious about neural network.

A video clip of a successful trial is shown in Figure 5. And the corresponding video can be found at https://youtu.be/C_4rVFgeUFA.

V. Discussion

Qualitatively, our results show that there are two main failure scenarios:

- As shown in Figure 6(a), the golf club is not striking the ball as expected, and the ball's initial velocity is markedly smaller than successful ones.
- As shown in Figure 6(b), the ball's initial velocity is within a reasonable range, but not accurately enough to hit the hole.

The first scenario suggests that our separation of the pre-hitting and hitting process still requires improvement. It is not robust to suddenly give the joints a very large velocity to hit ball. A better way is to interpolate the joint velocities and add the velocities to the kinematic optimization process during the pre-hitting stage. In addition, only using the last two joints to hit the ball may not be enough. With larger amount of data, we can fit a larger model considering more joints.

The second scenario implies that our modeling of hitting dynamics is not satisfying enough. Non-differentiable simulators like PyBullet [6] are not able to simulate the fast-evolving hitting process reliably when the timestep is relatively large ($> 1/60\,\mathrm{s}$). Relying purely on discrete collision detection and integration will inevitably bring numerical instability and non-physical behavior. What's more, we also found that the dynamics parameter such as contact stiffness and restitution will affect the hitting process notably. It will be interesting to investigate the modeling performance using differentiable simulators [10], [11] in such high-speed scenarios.

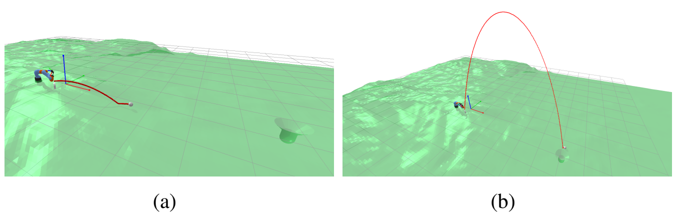


Fig. 6. Failure cases: (a) The strike is not successful. (b) The landing point is not accurate enough.

VI. Conclusion

In conclusion, we present a framework to play the golf with a robot arm, which has success rate 19% in our experiments. The framework consists of direct shooting, frame transformation, kinematic trajectory optimization, and system identification. The failure reasons of the framework include (1) suddenly give the joints very large velocities is not robust; (2) Controlling two joints to hit the ball may not be enough; (3) The simulator is not good at simulating the fast-evolving hitting process. These failure reasons are all interesting future topics.

Acknowledgement

The two authors contribute equally to the project and the final report. Detailed contribution divisions are as follows:

- Simulation environment was set up by Mingxin.
- Post-hitting stage was accomplished by Songyuan.
- Pre-hitting and hitting stages were done collaboratively.

References

[1] Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conference on robot learning*, pages 1262–1277. PMLR, 2021.

[2] Wenbo Gao, Laura Graesser, Krzysztof Choromanski, Xingyou Song, Nevena Lazic, Pannag Sanketi, Vikas Sindhwani, and Navdeep Jaitly. Robotic table tennis with model-free reinforcement learning. *IEEE International Conference on Intelligent Robots and Systems*, pages 5556–5563, 2020.

[3] Yapeng Gao, Jonas Tebbe, and Andreas Zell. Optimal stroke learning with policy gradient approach for robotic table tennis. *Applied Intelligence*, 2022.

[4] Dylan Zhou and Chaitanya Ravuri. Rallybot: Exploring physics-based approaches to robotic table tennis, 2021.

[5] Ashwin Vangipuram, Wen Hao Li, and William Loo. Ping pong bois, 2021.

[6] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2022.

[7] Keith J Laidler and John H Meiser. *Physical Chemistry*. Benjamin-Cummings Publishing Co., Subs. of Addison Wesley Longman, Reading, PA, January 1982.

[8] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[9] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981.

[10] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C. Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact, 2021.

[11] Taylor A. Howell, Simon Le Cleac'h, J. Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A Differentiable Simulator for Robotics. pages 1–19, 2022.