

# Learning to Play Table Tennis From Scratch using Muscular Robots

Dieter Büchler, Simon Guist, Roberto Calandra, Vincent Berenz, Bernhard Schölkopf and Jan Peters

**Abstract**—Dynamic tasks like table tennis are relatively easy to learn for humans, but pose significant challenges to robots. Such tasks require accurate control of fast movements and precise timing in the presence of imprecise state estimation of the flying ball and the robot. Reinforcement learning (RL) has shown promise in learning complex control tasks from data. However, applying step-based RL to dynamic tasks on real systems is safety-critical as RL requires exploring and failing safely for millions of time steps in high-speed and high-acceleration regimes. This paper demonstrates that using robot arms driven by pneumatic artificial muscles (PAMs) enables safe end-to-end learning of table tennis using model-free RL. In particular, we learn *from scratch* for thousands of trials while a stochastic policy acts on the low-level controls of the real system. The robot returns and *smashes* real balls with  $5\text{ ms}^{-1}$  and  $12\text{ ms}^{-1}$  on average respectively to a desired landing point. Additionally, we present HYSR, a practical hybrid sim and real training procedure that avoids training with real balls by virtually replaying recorded ball trajectories and applying actions to the real robot. To the best of our knowledge, this work pioneers (a) fail-safe learning of a safety-critical dynamic task using anthropomorphic robot arms, (b) learning a precision-demanding problem with a PAM-driven system that is inherently hard to control as well as (c) train a robot to play table tennis without real balls. Videos, code and datasets can be found on [muscularTT.embodied.ml](https://muscularTT.embodied.ml).

**Index Terms**—Reinforcement learning, sim-to-real, pneumatic muscles, dynamic task, real world robotics, robot table tennis.

## I. INTRODUCTION

**A**CQUIRING the skill to play table tennis appears easy even for young children but continues to be challenging for robots. Table tennis and other ball games can be considered dynamic tasks that require precise control of highly accelerated and fast motions. An aspect making dynamic tasks particularly intriguing is that slower motions are easier to control, but more accelerated and faster movements can improve the performance. High accelerations enable the robot to reach faster balls and balls farther away from the current end-effector position in less time. In addition, the robot can smash balls and thereby increase the variety of balls it can return. Studying the generation, precise control, and safe learning of highly accelerated real robot motions, thus, seems to be a promising direction to shed more light on what distinguishes today’s robots from humans.

Dieter Büchler, Simon Guist, Vincent Berenz, Bernhard Schölkopf and Jan Peters are with the Department of Empirical Inference, MPI for Intelligent Systems, Tübingen (Germany) ([dbuechler](mailto:dbuechler@tuebingen.mpg.de), [sguist](mailto:sguist@tuebingen.mpg.de), [vberenz](mailto:vberenz@tuebingen.mpg.de), [bs@tuebingen.mpg.de](mailto:bs@tuebingen.mpg.de), [mail@jan-peters.net](mailto:mail@jan-peters.net)). Jan Peters is also with the Department of Computer Science, Technische Universität Darmstadt (Germany). Roberto Calandra is with Meta AI Research (USA) ([rcalandra@fb.com](mailto:rcalandra@fb.com))

Manuscript received on February 6, 2021; revised on August 12, 2021; accepted on April 5, 2022

To optimally solve the trade-off between fast motions and precise control, many real-world robot table tennis approaches focus on adapting non-anthropomorphic systems such as delta robots. Such systems are inherently simpler to control and can reach high velocities and accelerations due to low inertia. In this manner, approaches using this type of body reach the state-of-the-art performance in real-world robot table tennis [1], [2].

Another stream of research in robot table tennis

focuses on using learning with more anthropomorphic human-sized robot arms. These works use techniques such as imitation learning [3], choosing or optimizing from safe demonstrations [4]–[6], minimizing acceleration for optimized trajectories [7], distributing torques over all degrees of freedom (DoF) [8] as well as cautious learning control approaches [9]. These approaches use primitives that are optimized before execution with little or no online adaptation. More recent works focus on leveraging deep reinforcement learning (RL) to learn to play table tennis [2]. For instance, [10] benchmarks the most recent RL algorithms. This work casts table tennis into a problem of lower complexity with a single step per episode, focusing on the most important aspects of the problem, such as the state of the ball when crossing the table plane and the ball state just before the hit. This approach uses a Kuka industrial robot since it relies on controllers that bring the robot with significant certainty to the desired hitting state and transfer well from simulation to reality due to the virtual pretraining. All of these approaches have in common to 1) reduce the control complexity by either using hardware that is easier to control or on desired joint or end-effector trajectories that a low-level controller tracks, and 2) incorporate prior knowledge to simplify the problem.

In this work, we take a more general perspective on table tennis and, for the first time, learn a real robot dynamic



Fig. 1. Our setup consists of a four DoF robot arm actuated by pneumatic muscles and a ballgun that launches balls towards the robot arm. The robot successfully learns to return the ball to the desired landing point on the table. A color-based camera detection system measures the Cartesian positions of the ball.

task in an end-to-end fashion. The goal is to learn such a task from scratch without any prior knowledge. Being able to learn dynamic tasks from scratch enables researchers to benchmark approaches on a real robot in the same way as in simulation. Multiple recent papers point out that overcoming the challenges to employ RL reliably on real robots [11]–[13] is crucial to develop better methods.

So far, learning from scratch on real anthropomorphic systems is only possible when constraining robots to slower motion skills due to safety concerns and hence limited to manipulation problems. For such tasks, engineered checks sufficiently assure robot safety. These checks detect, for instance, collisions with objects (usually based on heuristics) and the robot itself. Limiting joint accelerations and velocities, stopping the robot before it reaches its joint limits, and filtering noisy actions generated by stochastic policies [14], [15] may further ensure safety. However, as pointed out above, limiting the accelerations also caps the performance in dynamic tasks. In contrast, our goal is to let the agent freely explore fast motions.

At least two possible paths exist to allow robots to try out fast motions safely: The first is to define innocuous state transitions given the specifics of the dynamic task and dynamics of the robot, which can be explored for high reward. Unfortunately, finding safe areas in the state-action space requires exploration or prior knowledge about the task and system, which defies the objective of this work. Often approaches resort to safe but potentially suboptimal demonstrations, regularizers to punish harmful motions, or, as mentioned above, simple thresholds on velocities and accelerations. All these mechanisms usually sacrifice performance for safety.

The second way is to use inherently safe hardware, which is the subject of this work. It has been shown that systems actuated by pneumatic artificial muscles (PAM) are suitable to execute explosive hitting motions safely [16], [17]. A fast motion generated by the high forces of the PAMs can be decelerated by adjusting the pressure range for each PAM before a DoF exceeds its allowed joint angle range. Besides, PAM-driven robots are inherently backdrivable, which makes them exceptionally robust. However, such actuators are substantially more challenging to control than traditional motor-driven systems [18], [19]. For this reason, the predominant use of PAM-driven systems is to handle heavy objects slowly, for disturbance rejection, or as a testbed for control approaches.

In this paper, we show that the soft actuation of PAM-driven systems enables RL to be safely applied to a dynamic task directly on real hardware and, simultaneously, RL overcomes the difficulties of a dynamic task and the control issues of soft robots. By enabling RL to explore fast motions directly on the real system, we leverage the benefits of such complex systems for dynamic tasks (such as high power-to-weight ratio and storage of energy and release) despite their control issues. Rather than avoiding fast motions, we leverage the robot’s inherent safety to favor highly accelerated strikes by maximizing the velocity of the returned ball directly in the reward function without algorithmically handling safety. Also, we 1) apply noisy actions sampled from a Gaussian multi-layer perceptron (MLP) policy directly on the low-level

controls (desired pressures), 2) while running the RL algorithm for millions of time steps on the real hardware, 3) randomly initialize the policy at the start and 4) introduce a hybrid sim and real training procedure (HYSR) to circumvent practical issues of long-duration training such as collecting, removing, and supplying table tennis balls from and to the system. Using this training procedure, the agent learns to return and smash balls without touching any real ball during training.

It is worth mentioning that, with our setup, it is possible to learn robot table tennis while adding as minimal priors to the solution as possible. Neither do we have to add any constraints or regularizers, such as minimal accelerations or energy, nor do we have to use a higher abstraction level like task or joint space but instead learn directly on the low-level controls. As such, we do not rely on the underlying control framework of the particular system. Also, we avoid potentially suboptimal models or demonstrations. Models, constraints, and regularizers steer the policy’s optimization to a solution with possibly degenerated performance. Therefore, it is desirable to specify only task-related entities in the reward function and avoid such priors. In this manner, the solution mainly emerges from the hardware and the desired behavior defined in the reward function.

The goal of this paper is not to reach state-of-the-art performance in robot table tennis but is novel – to the best of our knowledge – in many other aspects: First, it demonstrates the first results in learning the challenging task of playing table tennis with muscular robots. In particular, this paper can be seen as a proof-of-concept that hard-to-control soft robots can be used for a precision-demanding task like table tennis. Second, this paper is the first to learn smash hitting motions on real robots by directly maximizing the ball’s velocity in the reward function while not being required to add safety measures. Third, this work pioneers applying model-free RL on safety-critical dynamic tasks directly on real hardware in the same way as in simulation. Fourth, we present the first approach to robot table tennis, where the training does not involve real balls. All of these contributions are only possible due to our careful choice of combination of techniques.

The remaining of the paper is divided as follows: In Section II, we introduce the task and reward functions used to learn to return and smash, as well as the hybrid sim and real training procedure (HYSR). The return and smash experiment are described in detail in Section III. We summarize the contributions and discuss the results in Section IV.

## II. TRAINING OF MUSCULAR ROBOT TABLE TENNIS

Learning dynamic tasks using muscular robots allows the RL algorithm to run on real systems similarly to simulated robots. In return, RL helps to overcome the inherent control difficulties of PAM-driven systems and leverage its beneficial properties, such as the powerful actuation to learn to return and smash real table tennis balls to desired landing points. To illustrate the details of this symbiosis, we discuss how robot types influence the performance in table tennis, introduce the task setup and the dense rewards used for the experiments, as well as the hybrid sim and real training (HYSR) that enables practical learning of these tasks.

## Variability in Recorded Ball Trajectories

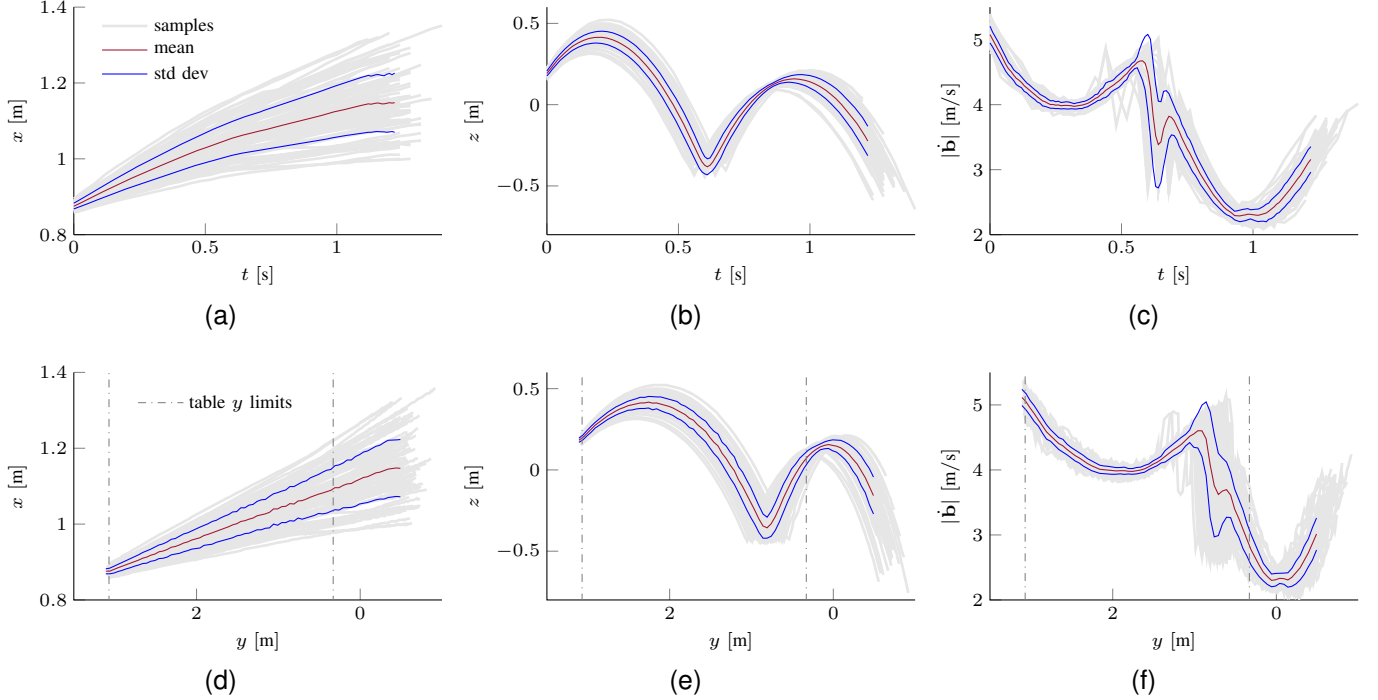


Fig. 2. Variability of the recorded ball trajectory dataset  $\mathcal{D}^{\text{rec}}$ . A ball launcher with fixed settings launches table tennis balls towards the robot. A color-based camera vision system [20] provided the position of the ball at 180 Hz. Variability is quantified by the sample mean and sample variance with respect to time a) to c) and along the long side of the table ( $y$  coordinate d) to f). The  $x$  coordinate aligns with the table’s shorter edge, and  $z$  coordinate links to the table plane’s normal. Although the settings of the ball launcher are constant, the ball varies substantially: 1) variability increases throughout the trajectory, especially after the bounce, 2) the first bounce on the table (incoming ball) varies around 50 cm along  $y$ -coordinate (subfigure e)), 3) the agent has to handle deviation of around 40 cm along the  $x$ -axis and  $\sim 50$  cm along the  $z$ -axis when the ball is in reach for the hit. 4) the variability with respect to time and the  $y$ -axis increases over time, which means that the agent cannot simply learn to start the hit after a particular fixed duration. Additionally, the agent needs to learn to adjust the amount of energy it transfers to each ball as the ball velocity also varies at the end of the table.

### A. Impact of Robot Type on Table Tennis Performance

The performance in robot table tennis significantly depends on the choice of robot. For instance, it seems natural that a robot similar to the human body in terms of DoFs and size has the potential to perform well with sufficient training time. A human can move along the table and has redundant ways to hit the ball due to the high number of DoFs. Parallel robots can reach similar dexterity as such robots can accelerate quickly due to low inertia. Additionally, such robots can be placed above the table to expand its reach. Such robots currently reach state-of-the-art performance in robot table tennis [1], [2]. Traditionally, motor-driven robots with a similar amount of DoFs as the human arm have been incorporated [4], [21]. Generally, the longer the links are, the higher the racket velocity can be reached with the same joint angular velocity. It appears to be slightly easier to reach good performance if such a robot is even placed on a linear axis [22], [23]. Lastly, motor-driven systems are generally easier to control precisely compared to soft actuation such as PAMs. For these reasons, we do not expect our system to beat the state-of-the-art in robot table tennis.

### B. Muscular Robot Table Tennis Task Setup

The considered table tennis task consists of returning an incoming ball with the racket attached to the robot arm to a desired landing point on the table  $\mathbf{b}^{\text{des}} \in \mathbb{R}^2$ . We denote the

ball trajectory  $\tau^{\text{b}} = [\mathbf{s}_t^{\text{b}}]_{t=0}^T$  consisting of a series of ball states  $\mathbf{s}_t^{\text{b}} = [\mathbf{b}_t, \dot{\mathbf{b}}_t]$  that themselves contain the current ball position  $\mathbf{b}_t \in \mathbb{R}^3$  and velocity  $\dot{\mathbf{b}}_t \in \mathbb{R}^3$ . Figure 2 depicts the ball trajectories considered in this paper. In a successful stroke, the robot hits the ball at time  $t_{\text{h}}$  and position  $\mathbf{b}^{\text{h}} \in \mathbb{R}^3$  such that the ball lands on the table plane at position  $\mathbf{b}^{\text{land}} \in \mathbb{R}^2$  at time  $t_{\text{land}}$ .

Table tennis falls into the general class of dynamic tasks such as baseball [24], tennis or hockey [25]. Dynamic tasks represent a class of problems that are relatively easy to solve for humans but hard for robots. The features of dynamic tasks are 1) quick reaction times as adapting to changes in the environment (such as moving balls) must happen fast, 2) precise motions because objects are supposed to reach some goal state (e.g., desired landing position on the table) and 3) fast and highly accelerated motions. The latter point is essential for two reasons: First, a successful strategy can incorporate fast strikes such as a table tennis smash. Second, if the ball’s desired hitting position is far from the current racket position, highly accelerated motions help reach this point in time. Thus, the maximum acceleration the system can generate represents the upper limit to the dexterity the agent can develop at such tasks. The class of dynamic problems differs from manipulation, where the task itself can be richer than a dynamic task in the sense that the objects and setting can vary vastly. Unlike in dynamic tasks, however, slow motions and

small accelerations are usually sufficient.

Safely generating high accelerations is harder with anthropomorphic robots than with parallel [1] or Cartesian systems [26]. Low inertia and force transmission without cables ease these systems' control, making high acceleration and estimating potentially dangerous motions feasible. Our work, in contrast, investigates table tennis with anthropomorphic robots. Damages on such systems can occur by breaking cables due to fast-changing control commands or exceeding joint limits when the system cannot stop in time. Learning the solution to the task while assuring robot safety makes anthropomorphic table tennis especially challenging.

Pneumatic artificial muscles (PAMs) are a particularly useful actuation system for anthropomorphic robots when applied to dynamic tasks. These actuators contract if the air pressure inside increases. Hence at least two PAMs act antagonistically on one degree of freedom (DoF) as a single PAM can only pull and not push. In this paper, we leverage the PAM-driven robot arm developed in [16], [17] which has four DoFs actuated by eight PAMs. Such robots are capable of generating high accelerations due to a high power-to-weight ratio. At the same time, adjusting the allowed pressure ranges prevents exceeding joint limits despite fast motions. We use this property in III-A and III-B to let the RL agent freely explore fast motions without any further safety considerations. Another benefit of PAM-driven systems is the inherent robustness resulting from passive compliance. This property helps reducing damage at impact due to shock absorption [27] as well as the adjustable stiffness via co-contracting the PAMs in an antagonistic pair. In this paper, we leverage this robustness to apply stochastic policies directly on the desired pressures, which are the low-level actions in this system (see Figure 3).

These numerous beneficial properties come at the cost of control challenges. PAMs are highly non-linear systems that change their dynamics with temperature as well as wear and are prone to hysteresis effects. Thus, modeling such systems for better control is challenging [18], [19]. PAMs are, for this reason, predominately used as a testbed for control algorithms rather than dynamic tasks. In this work, we show that it is possible to satisfy the precision demands of the table tennis task despite the control difficulties of PAM-driven systems by using RL (see Sections III-A and III-B).

### C. Dense Reward Functions for Returning and Smashing

We formulate the learning problem as an episodic Markov Decision Process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \mathcal{P}, p_0, \gamma) \quad (1)$$

where  $\mathcal{S}$  is the state space,  $\mathcal{A}$  is the action space,  $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$  is the immediate reward function,  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$  is the transition probability or dynamics,  $p_0$  is the initial state distribution and  $\gamma \in [0, 1]$  the discount factor. The goal in RL is to find a policy  $\pi : \mathcal{S} \mapsto p(\mathcal{A})$  that maximizes the expected return  $J = \mathbb{E}_{\tau} \sum_{t=0}^T \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)$  where  $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_T)$  is the state action trajectory,  $\mathbf{s}_0 \sim p_0$ ,  $\mathbf{a}_t \sim \pi(\mathbf{s}_t)$  and  $\mathbf{s}_{t+1} = \mathcal{P}(\mathbf{s}_t, \mathbf{a}_t)$ .

The state  $\mathbf{s} = [\mathbf{s}^b, \mathbf{s}^r]$  we use here is composed of the ball state  $\mathbf{s}^b$  and the robot state  $\mathbf{s}^r$ . The robot state  $\mathbf{s}^r = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{p}]$

### Noisy Actions of Stochastic Policy Applied Directly to Low Level Controls

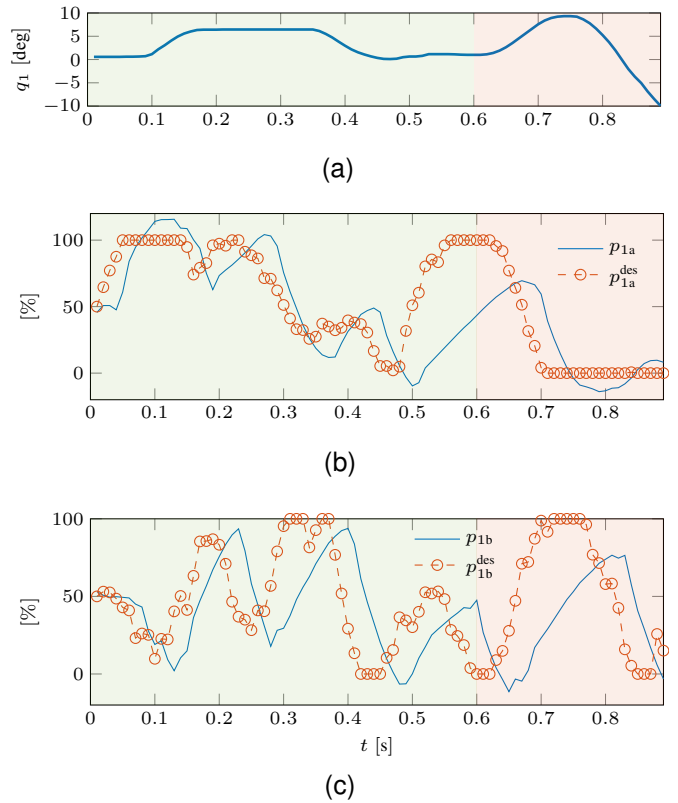


Fig. 3. Noisy actions due to sampling of a stochastic policy in every time step exemplified on the first DoF with both antagonistic PAMs  $p_{1a}$  and  $p_{1b}$  of one episode. The pressures are mapped from their particular minimal and maximal value to  $[0\% \dots 100\%]$ . The system's low-level controls are the desired pressures  $p^{des}$  which are set by the RL policy.  $p_{1a}$  and  $p_{1b}$  switch along the whole pressure range multiple times during the episode. The impact with the ball happens at  $t = 0.9$  s. At  $t = 0.6$  s the agent switches the pressures from minimum to maximum and vice versa to hit the ball. The green and red background color indicate the two distinct phases of preparation and hit from Figure 4. Applying such action sequences to the low-level controls of traditional robots of similar dimensions (e.g., link lengths) likely breaks the system. Figure 8 visualizes this return and a smash motion in more detail.

consists of the joint angles  $\mathbf{q} \in \mathbb{R}^4$ , joint angle velocities  $\dot{\mathbf{q}} \in \mathbb{R}^4$  and air pressures in each PAM  $\mathbf{p} \in \mathbb{R}^8$ . The ball state  $\mathbf{s}^b = [\mathbf{b}, \dot{\mathbf{b}}]$  has been already defined in II-B. The system we utilize here actuates each DoF with two PAMs. The actions  $\mathbf{a}$  are the change in desired pressures in each PAM  $\Delta \mathbf{p}^{des} \in \mathbb{R}^8$ .

In practice, the true Markovian state  $\mathbf{s}$  is not accessible in real robots experiments. Especially for PAM driven systems, the Markov state composition is unclear [19] leading to a partially observable MDP (POMDP) which assumes to receive observations  $\mathbf{o}$  of the true state  $\mathbf{s}$ . For the sake of clarity, we continue using  $\mathbf{s}$  instead of  $\mathbf{o}$  notation. The immediate reward function  $r(\mathbf{s}_t, \mathbf{a}_t)$  defines the goal of the task. The task of returning an incoming ball to a desired landing point can be divided into two stages: 1) manage to hit or touch the ball and 2) fine-tune the ball's impact with the racket such that it flies in a desired manner. As the landing location of the ball changes only in case the robot manages to hit the ball, we

introduce a conditional reward function

$$r = \begin{cases} r^{\text{tt}} & \text{racket touches the ball} \\ r^{\text{hit}} & \text{otherwise,} \end{cases} \quad (2)$$

where  $r^{\text{tt}}$  is the table tennis reward

$$r^{\text{tt}} = \begin{cases} 1 - c \|\mathbf{b}^{\text{land}} - \mathbf{b}^{\text{des}}\|^{\frac{3}{4}} & \text{return task} \\ (1 - c \|\mathbf{b}^{\text{land}} - \mathbf{b}^{\text{des}}\|^{\frac{3}{4}}) \max_{t > t_h} \|\dot{\mathbf{b}}_t\| & \text{smash task} \end{cases} \quad (3)$$

that evaluates the stroke depending on the ball trajectory after the impact of the ball and racket. In particular, it considers the distance of the actual landing point  $\mathbf{b}^{\text{land}}$  to the desired landing point  $\mathbf{b}^{\text{des}}$  for the return task (see III-A). The normalization constant  $c = \|\boldsymbol{\tau}_0^{\text{r}} - \mathbf{b}^{\text{des}}\|^{-1}$  is chosen such that  $r^{\text{tt}}$  is usually within the range  $[0, 1]$  where  $\boldsymbol{\tau}_0^{\text{r}}$  is the initial racket position. We also cap the table tennis reward  $r^{\text{tt}} = \max(r^{\text{tt}}, -0.2)$  in order to avoid too negative rewards in case the ball flies into a random direction with high velocity as happens if hit by an edge of the racket. We also introduce an exponent to the components of  $r^{\text{tt}}$  to cause the rewards' values to be more different closer to the optimal value. We found the exponent  $3/4$  to empirically work well. For the smashing task, the agent is supposed to maximize the ball velocity  $\dot{\mathbf{b}}$  simultaneously. The product between these two goals forces the agent to be precise *and* play fast balls as  $r^{\text{tt}}$  is small overall if a single component has a low value. The hitting reward

$$r^{\text{hit}} = - \min_t \|\boldsymbol{\tau}_t^{\text{b}} - \boldsymbol{\tau}_t^{\text{r}}\| \quad (4)$$

is a dense reward function representing the minimal Euclidean distance in time between the ball trajectory  $\boldsymbol{\tau}^{\text{b}}$  and Cartesian racket trajectory  $\boldsymbol{\tau}^{\text{r}}$  where  $\boldsymbol{\tau}_t^{\text{r}} = \mathbf{x}_t^{\text{r}} = \mathcal{T}(\mathbf{q}_t) \in \mathbb{R}^3$  using the forward kinematics function  $\mathcal{T}(\cdot)$ , the Cartesian racket position  $\mathbf{x}_t^{\text{r}}$  and ignoring the racket orientation. This reward function encourages the agent to get closer to the ball and finally hit it by providing feedback about how close it missed the racket.

Note that we do not incorporate *any* safety precautions such as state constraints like joint ranges, minimal accelerations  $\|\ddot{\mathbf{q}}_t\|$  or change in actions  $\|\mathbf{a}_t - \mathbf{a}_{t-1}\|$  into the reward function. On the contrary, we add a term that favors faster hitting motions. By doing so, we let the solution emerge purely from the hardware and the reward function that - in our case - incorporates only task-related terms.

#### D. HYSR: Hybrid Sim and Real Training

Running experiments with real robots and real objects for millions of time steps is a tedious practical effort. For instance, learning robot table tennis using model-free RL involves launching balls automatically, removing them from the scene after the stroke, and returning them to the ball reservoir. Automating this pipeline takes a substantial amount of work. Hence, we considered training with simulated balls. Simulation training, however, can be problematic: Simulated balls might differ too much from the real ball so that the learned policy might not be useful when playing with real balls. Additionally, the lack of suitable models of PAM-driven systems [18], [19] renders simulating the real robot infeasible.

---

#### Algorithm 1 HYSR: Hybrid Sim and Real Training

---

```

1: while max_timesteps not reached do
2:    $i \sim \text{uniform}(0, N^{\text{rec}})$ 
3:    $\mathcal{D}_i^{\text{rec}} \leftarrow \mathcal{D}^{\text{rec}}$ 
4:    $t \leftarrow 0$ 
5:   racketTouched  $\leftarrow$  false
6:   while episode end not reached do
7:      $\mathbf{s}_t^{\text{r}} \leftarrow \text{readSensors}()$ 
8:     if racketTouched then
9:        $\mathbf{s}_t^{\text{sim}} \leftarrow \text{sim}(\mathbf{s}_{t-1}^{\text{sim}})$ 
10:       $\mathbf{s}_t = [\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{sim}}]$ 
11:    else
12:       $\mathbf{s}_t^{\text{rec}} = \boldsymbol{\tau}_t^{\text{rec}}$ 
13:       $\mathbf{s}_t = [\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{rec}}]$ 
14:    end if
15:    if racketTouchesBall( $\mathbf{s}_t$ ) then
16:      racketTouched  $\leftarrow$  true
17:    end if
18:     $\mathbf{a}_t \sim \pi(\mathbf{s}_t)$ 
19:     $t = t + 1$ 
20:  end while
21:  update  $\pi$  using PPO
22: end while

```

---

For this reason, we introduce a hybrid sim and real training (HYSR) where the key idea is to use real data as much as possible and simulations wherever necessary. Specifically, actions  $\mathbf{a}$  sampled from the policy  $\pi$  are applied to the real robot, but the ball exists only in simulation during training. In simulation, we replay a single ball trajectory per episode sampled uniformly from a prerecorded data set  $\mathcal{D}^{\text{rec}} = [\boldsymbol{\tau}^{\text{rec}, i}]_{i=0}^{N^{\text{rec}}}$ . Within the recorded data set  $\mathcal{D}^{\text{rec}}$  the  $i$ -th trajectory consist of a sequence of ball states  $\boldsymbol{\tau}_t^{\text{rec}, i} = \mathbf{s}_t^{\text{rec}}$ .

The real robot is copied to the simulation by overwriting the simulated with the real robot state  $\mathbf{s}^{\text{r}}$  in every time step. In this manner, the real robot moves in simulation in the same way as in reality. In this manner, the simulation and the real scenery stay identical (the only difference being that the ball has been launched before training) until contact of the ball with the racket. At this point, it is impossible to predict the subsequent ball trajectory from  $\mathcal{D}^{\text{rec}}$ . For this reason, we start simulating the ball after the impact, which allows us to calculate the landing point of the ball  $\mathbf{b}^{\text{land}}$  more reliably as we simulate at the latest possible point. Essential for sufficiently accurate transfer from simulated to the real ball within HYSR is the rebound model of the ball with the racket. We found the rebound model from [28] to work well after optimizing its parameters empirically. The model

$$\dot{\mathbf{b}}_{\text{out}} - \dot{\mathbf{x}}_{t_h}^{\text{r}} = \epsilon_{\text{R}}(-\dot{\mathbf{b}}_{\text{in}} + \dot{\mathbf{x}}_{t_h}^{\text{r}}) \quad (5)$$

calculates the outgoing velocity of the ball  $\dot{\mathbf{b}}_{\text{out}}$  from the ball velocity  $\dot{\mathbf{b}}_{\text{in}}$  before impact, the racket speed  $\dot{\mathbf{x}}_{t_h}^{\text{r}}$  at impact (all measured along the racket normal) and the restitution coefficient of the racket  $\epsilon_{\text{R}}$ . Note that this model assumes no spin. Algorithm 1 summarizes a single episode of the training procedure.

### Visualization of Learned Two Stage Hitting Motion

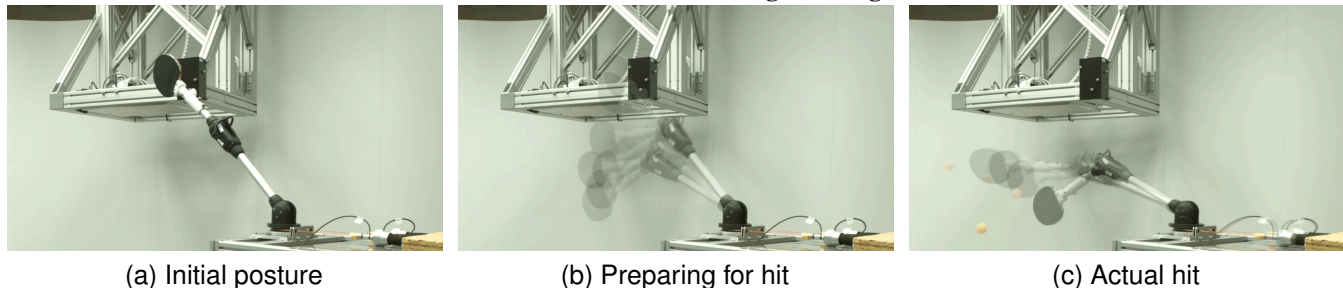


Fig. 4. Images extracted from a video showing the learned hitting motion of the return experiment. After an episode, the robot initializes to an initial position shown in a), which takes 2 s to 4 s. The agent automatically learned two distinctive hitting stages consisting of b) preparing for a hit and c) the actual hit.

It is important to mention that, although the settings of the ball launcher have been kept fixed, the  $N^{\text{rec}} = 100$  recorded ball trajectories in  $\mathcal{D}^{\text{rec}}$  broadly vary as can be seen in Figure 2. Variability arises through the noise in the color-based vision system that automatically detects balls and returns their Cartesian position. Another reason is that the ball launcher adds little noise to each ball, accumulating more significant deviations the farther the ball flies.

HYSR allows us to leverage the simulation conveniently to estimate the landing position  $\mathbf{b}^{\text{land}}$  and other context entities such as distances or boolean contact indicators that are important for the reward (see Equation (4) and Equation (3)). Besides, we avoid collecting and launching real balls. Reliably and practically collecting real data is important when training with RL. HYSR enabled us to learn to return real balls well without experiencing how the ball rebounds from the racket and how it flies to the opponent’s side (see Sections III-A and III-B). This fact seems counterintuitive and was only possible due to the way we train. In this manner we were able to renounce from fine-tuning with real balls when testing with real balls. Note that this training method can serve as an entry point for sim2real techniques such as domain randomization or curriculum learning. For instance, the training’s progress could be used to update the ball’s initial state  $\mathbf{s}_0^{\text{b}}$ , or the ball trajectory  $\tau^{\text{b}}$  can be perturbed.

### III. EXPERIMENTS AND EVALUATIONS

This paper’s key contributions are to 1) enable RL to explore fast motions of soft robots without safety precautions and by doing so, 2) learn a difficult dynamic task using RL with a complex and real system. To show 2), we learn to return and smash a table tennis ball with a PAM-driven robot arm using the HYSR training described in Section II-D and the reward functions discussed in Section II-C. We highlight 1) by quantifying the robustness of the system during the training. In particular, we illustrate the returned ball’s speed and depict the noisy actions on the low-level controls of the real system due to the application of a stochastic policy. Results are best seen in the supplemental videos at [muscularTT.embodied.ml](https://muscularTT.embodied.ml).

#### A. Learning to Return

Returning table tennis balls with PAM-driven systems is a challenging problem due to PAMs being hard to model and

control [18], [19] as well as table tennis requiring precise control of the racket at impact with the ball. In this task, the robot’s task is to return balls to a desired landing position  $\mathbf{b}^{\text{des}}$  (see Figure 7) on the opposite side of the table shot by a ball launcher as shown in Figure 1 and described in Section II-B. We demonstrate that by enabling RL to explore freely at fast paces, the agent can learn this task well.

We let the robot train for 1.5 million time-steps using a stochastic policy. The stochasticity is vital to guide the exploration, which decreases with the progress in learning. At test time, when playing with real balls, we use the mean of the learned stochastic policy. The policy has been randomly initialized and the actions are changes in target pressures  $\Delta \mathbf{p}^{\text{des}}$ . One strike corresponds to one episode. At the end of each episode, the agent receives a reward according to the dense return reward function from Equation (3). We use PPO [29] as a backbone RL algorithm. In particular, we leverage the ppo2 implementation of PPO from OpenAI baseline [30]. Table I lists the hyperparameters used for this experiment.

After training, the final policy has been evaluated with real balls. The agent hits 96% and returns 77% of the 107 real balls shot by the ball launcher, as indicated in Table II. Figure 7 illustrates that the landing points spread in a circle around the desired landing point. This circle overlaps with the opponent’s side but is not fully contained by it. Thus, the return rate to the table would have been higher if  $\mathbf{b}^{\text{des}}$  would be moved towards the center of the table half.

Interestingly, the agent did not only learn to intercept the ball but also to prepare for the hit, as can be seen in Figure 4. These two stages are part of the four stages of a table tennis game introduced in [31] which have been validated from human play [28]. This behavior emerged, although the goal was only to return the ball to a desired landing point. Specifying the same behavior within the classical pipeline of 1) planning a reference trajectory and 2) tracking with an existing (model-based) controller appears to be more difficult: Such approaches have the disadvantage of relying on human expert knowledge, which is likely to lead to non-optimal solutions considering the challenges of the control problem being solved. Hence, this work represents a type of end-to-end approach to dynamic tasks where we learn a mapping from sensor information to low-level controls directly. This kind of approaches are only possible if the hardware is sufficiently

**Histogram of Maximal Speeds of Returned Balls**

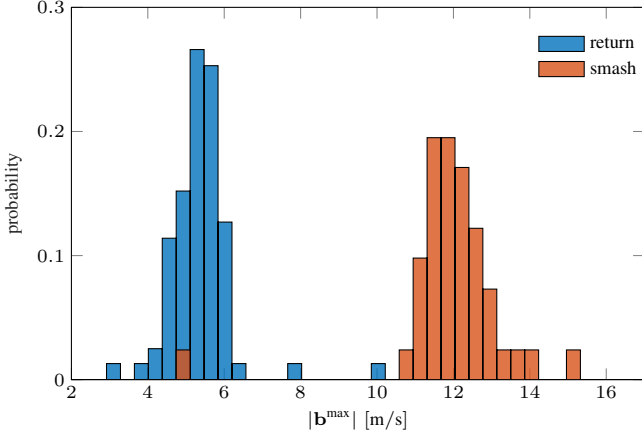


Fig. 5. Histograms depicting the speeds of the balls for the return and smash experiment. Considered are the maximum velocities after impact with the racket. The final policy of each experiment generated the data for the particular histogram. One can see that the speeds significantly increase for the smashing experiment.

robust.

**B. Learning to Smash**

In table tennis, smashing is a means of maximizing the ball’s velocity, such that the opponent has a hard time returning it. The motion needs to be very fast and, at the same time, precise enough to return the ball to the opponent’s side. Smashing is more challenging to learn than just returning balls, as imprecise motions might lead to the ball not being returned on the table entirely. Here we show that, by using soft robots, we can learn this skill using RL only by defining a reward function that maximizes the ball speed and minimizes the distance to the desired landing point (Equation (3)) concurrently. Rather than taking safety into account algorithmically, we - on the contrary - favor aggressive and explosive motions.

We chose to repeat the experiment from Section III-A with the same hyperparameters as in Section III-A but updated the reward function with the speed bonus from Equation (3). The

**TABLE I**  
HYPERPARAMETERS USED FOR RL EXPERIMENTS

hyperparameter	value
algorithm	ppo2
network	mlp
num_layers	1
num_hidden	512
activation	tanh
nsteps	4096
ent_coef	0.001
learning_rate	lambda f:1e-3*f
vf_coefs	0.66023
max_grad_norm	0.05
gamma	0.9999
lam	0.98438
nminibatches	8
noptepochs	32
cliprange	0.4

**Learning Curves of Return and Smash Experiments**

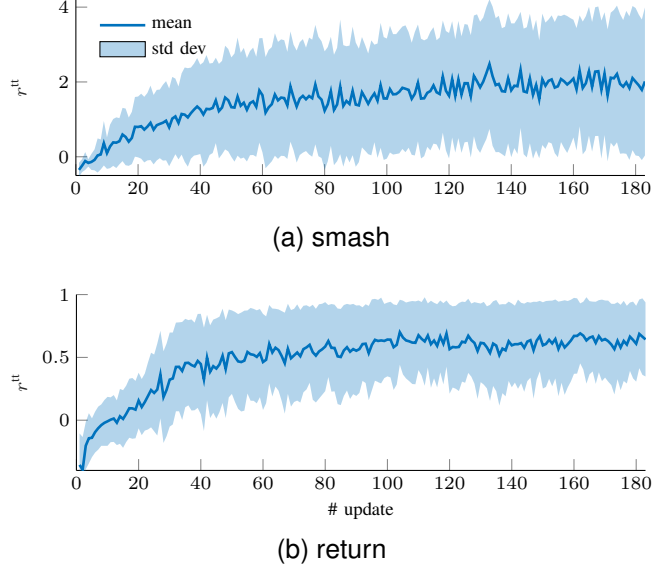


Fig. 6. Sample mean and standard deviation of the rewards of each episode for the a) smash and b) return task from Section III-A and Section III-B. The policies for both experiments were updated 183 times. The number of episodes differs for each experiment as we define the end of the training by the number of time steps (1.5 million for both experiments at 10 ms), but the actual episode length varies. The return task required 15676 and the smash task 15161 strokes or episodes. Each episode takes approximately 1 s with additional initialization of the robot (2 to 4 s, see Figure 4a) The total moving time is 14 h and 10 min for the return task and 14 h and 18 min for the smash experiment.

learning curve is depicted in Figure 6. Compared with the learning curve from Section III-A, the smashing task is harder to learn for the agent. The standard deviation of the reward for the smash experiment is higher than in the return task. Also, the precision of the returned balls is lower in the smash experiment, as shown in Figure 7.

Figure 5 shows histograms of the maximum ball velocities after the hit for reward function with and without speed bonus. The ball speed increases when the reward contains a speed bonus. For the return task, the robot returns balls at  $5 \text{ m s}^{-1}$  on average, whereas in the smash experiment, this number increases to  $12 \text{ m s}^{-1}$ . In related work, balls faster than  $10 \text{ m s}^{-1}$  have been considered to be smashes in human play [32].

The higher return speed comes at a cost: The return and hitting rates indicated in Table II show that the faster the hit, the robot returns the ball less precisely. Hence, the more energy is transferred to the ball, the higher the chance of failing.

**TABLE II**  
RETURN AND HITTING RATES OF RETURN (107 TRAILS) AND SMASH (128 TRAILS) EXPERIMENTS

task	hitting rate	rate of returning to opponents side
return	0.96	0.75
smash	0.77	0.29

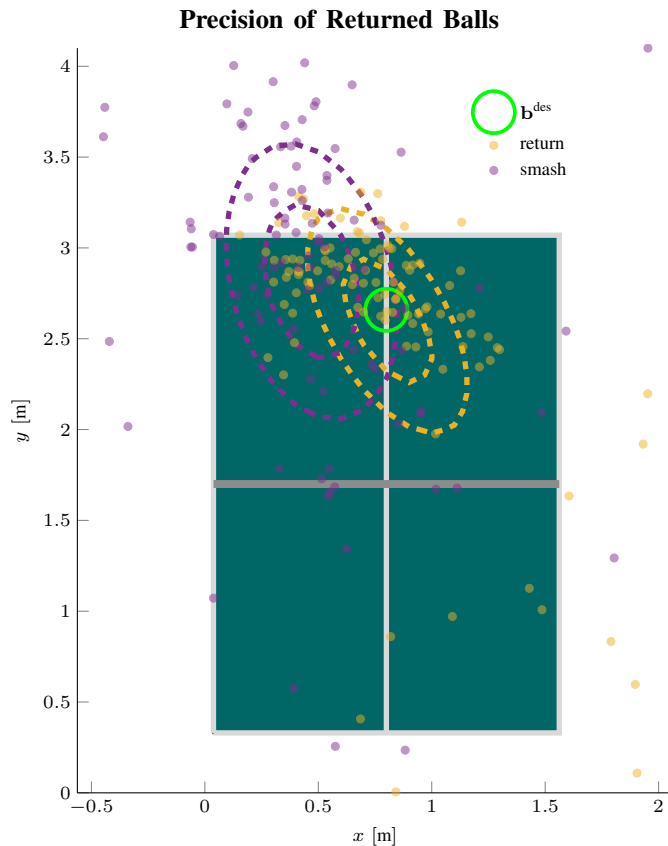


Fig. 7. Landing positions of the returned balls for the return as well as for the smash experiment from Section III-A and Section III-B. The landing points of the smash experiment are further apart from the desired landing point compared to the return experiment as can be seen by the fitted Gaussian distributions. The mean of this distribution for the return experiment coincides with the desired landing point  $\mathbf{b}^{\text{des}}$  but is shifted for the smash experiment. Simultaneously, the smashed balls are faster, hence, corresponding to a more aggressive but risky gameplay strategy. At such speeds, a small error in the racket orientation leads to more significant deviations in landing point.

### C. Robustness

The robustness of the PAM-driven robot arm enables the RL algorithm to explore fast motions while executing a stochastic policy directly on the low-level controls. We quantify the robustness of this PAM-driven system in multiple manners.

First, the sheer number of running a real system for 1.5 million training time-steps stresses that soft robots are indeed robust. 1.5 million time steps at 100 Hz are equivalent to 4 h and 16 min of actual training time. Also, the robot initializes after each episode, which takes a further 2 to 4 s per episode. In total, we train the return task for 14 h and 18 min and the smash task for 14 h and 10 min. Within these durations, both experiments' policies are updated 183 times, and we perform 15676 (return) and 15161 (smash) strokes, each corresponding to one episode. Note that the training stopped because the algorithm converged and not due to hardware issues.

Second, each episode is carried out by sampling actions from a Gaussian multi-layer perceptron (MLP) at every time-step. For this reason, the signals are noisy and can vary substantially with each time step. Figure 3 depicts the actions  $\Delta p_{1a}^{\text{des}}$  and  $\Delta p_{1b}^{\text{des}}$  of the first DoF and the measured pressures

$p_{1a}$  and  $p_{1b}$  in percent of the allowed pressure ranges alongside with the corresponding joint angle  $q_1$ . The hit of the ball happens at 0.9 s. The agent learned to switch the actions from minimum to maximum pressure and vice versa around  $t = 0.6$  s right before the hit. In the preparation phase (see Figure 4a) before the hit (indicated by the green background color in 3), the agent used the whole pressure range to bring the robot into a beneficial initial state for the hit. Applying such signals to the low-level controls of a traditional motor-driven system with the same dimensions as our robot arm presumably causes damage or at least severe wear.

Third, in both experiments, we learn from scratch, where the initial policy receives random weights. Nevertheless, the motions during training did not exceed the joint limits because the allowed pressure ranges have been set such that one of the muscles in the antagonistic pair is stretched close to the respective joint limit [16], [17]. In this manner, the robot can train *without* human supervision. To achieve safety for dynamic motions on traditional robotic systems, a filter on the actions have been used [14]. However, there are multiple downsides to this approach: 1) Adding a filter makes the state non-Markovian if the internal filter state is not part of the RL state, which would, in turn, increase the RL state's dimensions. 2) Tuning the filter can be tedious: On the one hand, setting the parameters too conservatively leads to slower motions, although the task requires fast motions. On the other hand, too optimistic parameters might damage the robot for some configurations. 3) Filtering the stochastic output of the policy counteracts its intended use.

## IV. CONCLUSION

Accurately returning and smashing table tennis balls to a desired landing point with anthropomorphic robots requires the execution of precise *and* fast motions. Exploration at fast speeds is highly desirable for improving accuracy but might damage the robot, e.g., exceeding joint limits. The synergy between soft robots and Reinforcement learning for dynamic tasks resolves this dilemma. The robustness of soft actuation allows RL to act like in simulation and, vice versa, RL helps to overcome the difficulties of the table tennis tasks and the control problems of soft robots. An exciting finding is that in our experiments, it was neither necessary to take safety into account in the algorithm nor was any model or demonstration needed. On the contrary: We even encouraged returning the ball with high speed by an additional term in the reward function resulting in explosive hitting motions while learning from scratch for millions of time steps. To make training more practical, we introduce HYSR, a hybrid sim and real training procedure that allows the robot to learn from thousands of strokes without touching any real balls during training. With these choices, this paper is the first 1) to achieve sufficiently accurate motions for the precision demanding task of table tennis with PAM-driven soft robots, 2) to enable fail-safe learning of the safety-critical task of smashing real balls directly on a real robot using model-free RL *from scratch* and 3) to learn robot table tennis without using real balls during training.

In this paper, we proofed our contributions on a single task. Still, our approach is applicable beyond table tennis. In particular, learning on muscular robots can be safely performed in all scenarios where the focus is on robot safety rather than the safety of the surroundings. This assumption is true for many ball or non-contact sports, such as baseball, golf or parts of a football game, but might be limited for manipulation tasks. Similarly, HYSR does not only relief from handling table tennis balls during long duration training but is generally advantageous if the robot is more challenging to model than the environment such as for soft robots.

In future work, we aim at improving the sample-efficiency of our training. HYSR, for instance, can be extended with data-augmentation techniques or by prioritizing the replayed ball trajectory using curriculum learning. Although not the focus of this paper, it is worth noting that our approach to learning table tennis is less sample-efficient than previous robot table tennis approaches. More efficient training would enable us to improve table tennis performance. Here, the most crucial objective will be to perfect the precision of returned balls. Subsequently, we aim at extending the task itself, such as serving balls, playing fore- and backhand strikes, or two robot play. It is important to mention that the current version of the real PAM-driven system still suffers from severe non-linear friction and stiction effects due to the cable drive. Improving in this aspect will additionally lead to better performance.

This paper is a step towards highlighting the beneficial synergy of soft robots and learning approaches, which is not widely explored yet. For this purpose, we open-source the data collected throughout the experiments. These rich data sets are suitable for benchmarking dynamics models as they contain a variety of motions at different speeds. Also, we present the videos of the full training and a summarizing video as well as the code at `muscularTT.embodied.ml`.

## APPENDIX A

### VISUALIZATION OF A RETURN AND SMASH

Using the training procedure from Section II, the RL agent automatically learned two distinct phases of a strike motion (prepare and hit). This two distinct phases are depicted in Figure 4 and indicated by the green (prepare) and red (hit) background color in Figure 3 and, in more detail, in Figure 8. Figure 8 shows a smash motion from Section III-B in addition to a return strike from Section III-A. In contrast to the return motion, the agent learned to gain momentum in Dof 1 from the beginning of the episode and uses the other Dofs for finetuning within the hitting phase. The agent also learned two different strategies for the return and smash motion: The return uses the third DoF and the first DoF, whereas DoF 1 and 4 are used for the smash. The low-level actions  $p^{\text{des}}$  switch multiple times within the allowed range without damaging the system. In this manner, the system’s robustness enables the RL agent to try and fail in a safety-critical task to find the optimal policy.

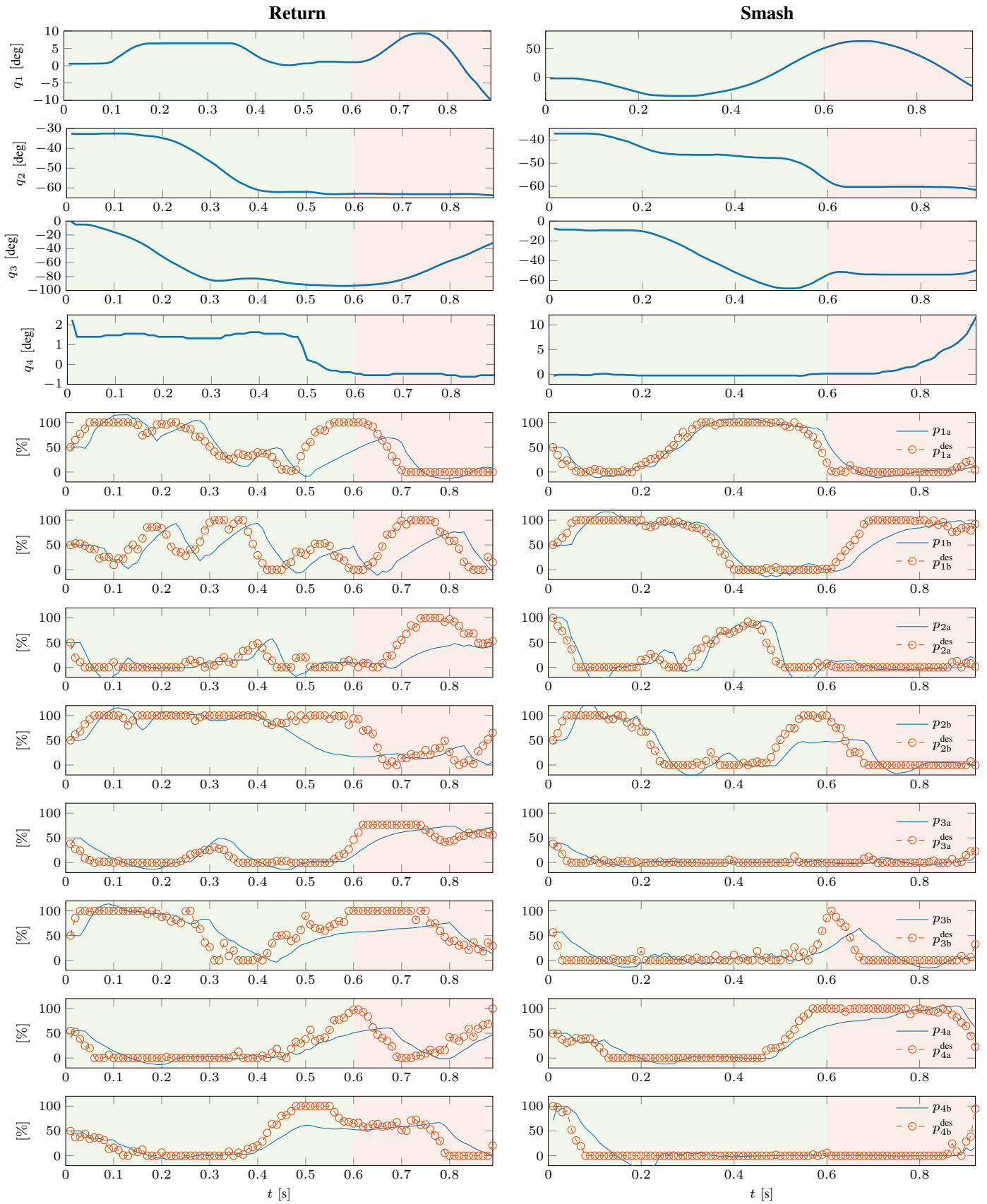


Fig. 8. Joint angle and action trajectories of a single return (left) and smash (right) motion from Section III-A and Section III-B. The pressures are mapped from their particular minimal and maximal pressures to  $[0\% \dots 100\%]$ . The ball's impact happens at  $t \approx 0.9$ s for the return and the smash. Green and red background color indicate the prepare and hit phase from Figure 4.

REFERENCES

- [1] S. Kawakami, M. Ikumo, and T. Oya, "Omron table tennis robot forpheus," Tech. Rep., 2016. [Online]. Available: <https://www.omron.com/innovation/forpheus.html>
- [2] L. Yang, H. Zhang, X. Zhu, and X. Sheng, "Ball Motion Control in the Table Tennis Robot System Using Time-Series Deep Reinforcement Learning," *IEEE Access*, vol. 9, pp. 99 816–99 827, 2021, conference Name: IEEE Access.
- [3] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters, "Adaptation and Robust Learning of Probabilistic Movement Primitives," *arXiv:1808.10648 [cs, stat]*, Aug. 2018, arXiv: 1808.10648. [Online]. Available: <http://arxiv.org/abs/1808.10648>
- [4] K. Muelling, J. Kober, and J. Peters, "Learning table tennis with a Mixture of Motor Primitives," in *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Dec. 2010, pp. 411–416.
- [5] 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, Mar. 2013. [Online]. Available: <http://ijr.sagepub.com/content/32/3/263>
- [6] Y. Huang, D. Büchler, O. Koç, B. Schölkopf, and J. Peters, "Jointly learning trajectory generation and hitting point prediction in robot table tennis," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov. 2016, pp. 650–655.
- [7] O. Koç, G. Maeda, and J. Peters, "Online optimal trajectory generation for robot table tennis," *Robotics and Autonomous Systems*, vol. 105, pp. 121–137, Jul. 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921889017306164>
- [8] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 853–858.
- [9] O. Koç, G. Maeda, and J. Peters, "Optimizing the Execution of Dynamic Robot Movements With Learning Control," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 909–924, Aug. 2019.
- [10] J. Tebbe, L. Krauch, Y. Gao, and A. Zell, "Sample-efficient Reinforcement Learning in Robotic Table Tennis," *arXiv preprint arXiv:2011.03275*, 2020.
- [11] C. Bodnar, A. Li, K. Hausman, P. Pastor, and M. Kalakrishnan, "Quantile QT-Opt for Risk-Aware Vision-Based Robotic Grasping," *arXiv:1910.02787 [cs, stat]*, Oct. 2019, arXiv: 1910.02787. [Online]. Available: <http://arxiv.org/abs/1910.02787>
- [12] A. Majumdar, S. Singh, A. Mandekar, and M. Pavone, "Risk-sensitive Inverse Reinforcement Learning via Coherent Risk Models," in *Robotics: Science and Systems*, 2017.
- [13] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of Real-World Reinforcement Learning," *arXiv:1904.12901 [cs, stat]*, Apr. 2019, arXiv: 1904.12901. [Online]. Available: <http://arxiv.org/abs/1904.12901>
- [14] D. Schwab, T. Springenberg, M. F. Martins, T. Lampe, M. Neunert, A. Abdolmaleki, T. Hertweck, R. Hafner, F. Nori, and M. Riedmiller, "Simultaneously Learning Vision and Feature-based Control Policies for Real-world Ball-in-a-Cup," *arXiv:1902.04706 [cs, stat]*, Feb. 2019, arXiv: 1902.04706. [Online]. Available: <http://arxiv.org/abs/1902.04706>
- [15] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates," *arXiv:1610.00633 [cs]*, Oct. 2016, arXiv: 1610.00633. [Online]. Available: <http://arxiv.org/abs/1610.00633>
- [16] D. Büchler, H. Ott, and J. Peters, "A Lightweight Robotic Arm with Pneumatic Muscles for Robot Learning," in *International Conference on Robotics and Automation (ICRA)*, Stockholm, May 2016.
- [17] D. Büchler, R. Calandra, and J. Peters, "Learning to Control Highly Accelerated Ballistic Movements on Muscular Robots," *arXiv:1904.03665 [cs]*, Apr. 2019, arXiv: 1904.03665. [Online]. Available: <http://arxiv.org/abs/1904.03665>
- [18] B. Tondou, "Modelling of the McKibben artificial muscle: A review," *Journal of Intelligent Material Systems and Structures*, vol. 23, no. 3, pp. 225–253, Feb. 2012. [Online]. Available: <http://jim.sagepub.com/content/23/3/225>
- [19] D. Büchler, R. Calandra, B. Schölkopf, and J. Peters, "Control of Musculoskeletal Systems using Learned Dynamics Models," *IEEE Robotics and Automation Letters*, 2018.
- [20] S. Gomez-Gonzalez, Y. Nemmour, B. Schölkopf, and J. Peters, "Reliable Real Time Ball Tracking for Robot Table Tennis," *arXiv:1908.07332 [cs]*, Aug. 2019, arXiv: 1908.07332. [Online]. Available: <http://arxiv.org/abs/1908.07332>
- [21] J. Tebbe, Y. Gao, M. Sastre-Rienietz, and A. Zell, "A table tennis robot system using an industrial kuka robot arm," in *German Conference on Pattern Recognition*. Springer, 2018, pp. 33–45.
- [22] W. Gao, L. Graesser, K. Choromanski, X. Song, N. Lazic, P. Sanketi, V. Sindhwani, and N. Jaitly, "Robotic Table Tennis with Model-Free Reinforcement Learning," in *2020 IEEE/RISJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 5556–5563, ISSN: 2153-0866.
- [23] SIASUN, "Siasun table tennis robot pongbot," 2017. [Online]. Available: [https://www.youtube.com/watch?v=Ov8jwAKucmk&ab\\_channel=TableTennisFan\\_ThePenholder](https://www.youtube.com/watch?v=Ov8jwAKucmk&ab_channel=TableTennisFan_ThePenholder)
- [24] T. Senoo, A. Namiki, and M. Ishikawa, "Ball control in high-speed batting motion using hybrid trajectory generator," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 1762–1767.
- [25] G. Neumann, C. Daniel, A. Paraschos, A. Kupcsik, and J. Peters, "Learning Modular Policies for Robotics," *Frontiers in Computational Neuroscience*, vol. 8, p. 62, 2014. [Online]. Available: <http://journal.frontiersin.org/Journal/10.3389/fncom.2014.00062/pdf>
- [26] Y. Huang, D. Xu, M. Tan, and H. Su, "Adding Active Learning to LWR for Ping-Pong Playing Robot," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1489–1494, Jul. 2013, conference Name: IEEE Transactions on Control Systems Technology.
- [27] K. Narioka, T. Homma, and K. Hosoda, "Humanlike ankle-foot complex for a biped robot," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, Nov. 2012, pp. 15–20.
- [28] K. Mülling, J. Kober, and J. Peters, "A biomimetic approach to robot table tennis," *Adaptive Behavior*, vol. 19, no. 5, pp. 359–376, Oct. 2011. [Online]. Available: <http://adb.sagepub.com/content/19/5/359>
- [29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs]*, Jul. 2017, arXiv: 1707.06347. [Online]. Available: <http://arxiv.org/abs/1707.06347>
- [30] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Openai baselines," *GitHub, GitHub repository*, 2017.
- [31] Ramanantsoa and Durey, "Towards a stroke Construction Model," *ITTF Education*, Jan. 1994. [Online]. Available: <https://www.ittfeducation.com/towards-stroke-construction-model/>
- [32] K. Muelling, A. Boularias, B. Mohler, B. Schölkopf, and J. Peters, "Learning strategies in table tennis using inverse reinforcement learning," *Biological cybernetics*, 2014. [Online]. Available: [http://www.ias.tu-darmstadt.de/uploads/Site/EditPublication/Muelling\\_BICY\\_2014.pdf](http://www.ias.tu-darmstadt.de/uploads/Site/EditPublication/Muelling_BICY_2014.pdf)

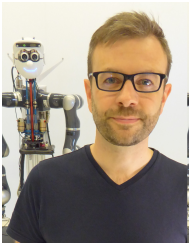


Machine Learning and Robotics.

**Dieter Büchler** Dieter Büchler is a research group leader at the Max Planck Institute for Intelligent Systems within the Empirical Inference department in Tübingen (Germany) led by Bernhard Schölkopf. Previously, Dieter received his PhD from the Technische Universität Darmstadt under the supervision of Jan Peters, a M.Sc. in Biomedical Engineering from the Imperial College London (United Kingdom), and a B.Eng. in Information and Electrical Engineering from the HAW Hamburg (Germany). His research interests are in the areas of Control,

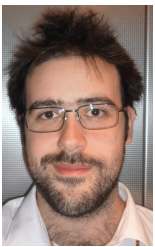


**Simon Guist** Simon Guist is a PhD student at the Max Planck Institute for Intelligent Systems in the Empirical Inference department in Tübingen (Germany) led by Bernhard Schölkopf. Before, Simon pursued a M.Sc. degree in Computer Science from the University Heidelberg (Germany). Simon's research interests are in the areas of Reinforcement Learning approaches that help real robots to learn increasingly complex and general skills.



**Vincent Berenz** Vincent Berenz is scientific software engineer in the Autonomous Motion Department at the Max Planck Institute for Intelligent Systems (Tübingen, Germany) since January 2015. Vincent received a Chemical Engineering degree from Ecole Supérieure de Chimie Organique et Minérale (2001, France) and a Master of Business Engineering Bio-informatics from Ecole de Biologie Industrielle (2002, France). He worked as a software engineer at CEREP (USA and France) and Pharmedesign (Japan) from 2002 to 2006 and from 2006

to 2007, respectively. In 2012, he received his Ph.D. in Intelligent Interactions Technologies from the University of Tsukuba (Japan) under the supervision of Kenji Suzuki. In 2013 and 2014 he was a research scientist at RIKEN Brain Science Institute/Toyota Collaborative Center (Japan).



**Roberto Calandra** Roberto Calandra is a Research Scientist at Facebook AI Research (FAIR). Previously, he was a postdoctoral scholar at UC Berkeley in the Berkeley Artificial Intelligence Research Laboratory (BAIR). Roberto received a Ph.D. from TU Darmstadt (Germany), a M.Sc. in Machine Learning and Data Mining from the Aalto university (Finland), and a B.Sc. in Computer Science from the Università degli studi di Palermo (Italy). His scientific interests focus at the conjunction of Machine Learning and Robotics, in what is known as Robot

Learning.



**Bernhard Schölkopf** Bernhard's scientific interests are in machine learning and causal inference. He has applied his methods to a number of different application areas, ranging from biomedical problems to computational photography and astronomy. Bernhard has researched at AT&T Bell Labs, at GMD FIRST, Berlin, and at Microsoft Research Cambridge, UK, before becoming a Max Planck director in 2001. He is a member of the German Academy of Sciences (Leopoldina), and has received the J.K. Aggarwal Prize of the International Association for

Pattern Recognition, the Max Planck Research Award (shared with S. Thrun), the Academy Prize of the Berlin-Brandenburg Academy of Sciences and Humanities, the Royal Society Milner Award and the Körber Prize.



**Jan Peters** Jan Peters is a full professor (W3) for Intelligent Autonomous Systems at the Computer Science Department of the Technische Universität Darmstadt and at the same time an adjunct senior research scientist at the Max Planck Institute for Intelligent Systems, where he heads the Robot Learning Group of the Empirical Inference department. Jan Peters has received a few awards, most notably, he has received the Dick Volz Best 2007 US Ph.D. Thesis Runner Up Award, the 2012 Robotics: Science & Systems— Early Career Spotlight, the

2013 IEEE Robotics & Automation Society's Early Career Award, and the 2013 INNS Young Investigator Award. In 2019, he was appointed as IEEE Fellow.