High Acceleration Reinforcement Learning for Real-World Juggling with Binary Rewards

Jonglage mit einem Highspeed-Roboterarm

Master thesis by Kai Ploeger

Date of submission: May 28, 2020

1. Review: M.Sc. Michael Lutter
2. Review: Prof. Dr. Jan Peters

Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Kai Ploeger, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 28. Mai 2020

Kai Ploeger

Abstract

Robots that can learn in the physical world will be important to enable robots to overcome their stiff and pre-programmed movements. For dynamic high-acceleration tasks, such as juggling, learning in the real-world is particularly challenging as one must push the limits of the robot and its actuation without harming the system. Therefore, learning these tasks on physical system amplifies the necessity of sample efficiency and safety for robot learning algorithms, making a high-speed task an ideal benchmark to highlight robot learning systems. To achieve learning on the physical system, we propose a learning system that directly incorporates the safety and sample efficiency requirements in the design of the policy representation, initialization and optimization. This approach is in contrast to prior work which mainly focuses on the learning algorithm details, but neglect the engineering details. We demonstrate that our system enables the high-speed Barrett WAM to learn juggling of two balls from 56 minutes of experience. The robot learns to juggle consistently solely based on a binary reward signal. The optimal policy is able to juggle for up to 33 minutes or about 4500 repeated catches. The videos documenting the learning process and the evaluation can be found at **https://sites.google.com/view/jugglingbot**

Zusammenfassung

Roboter, die in der physischen Welt lernen können, werden wichtig sein, um steifen und vorprogrammierten Bewegungen entkommen zu können. Bei dynamischen Aufgaben mit hohen Beschleunigungen, wie z.B. Jonglieren, ist das Lernen in der realen Welt besonders anspruchsvoll, da man die Grenzen des Roboters und seiner Aktuatoren ausreizen muss, ohne das System zu beschädigen. Daher verstärkt das Lernen dieser Aufgaben auf dem physischen Roboter die Notwendigkeit der Sample-Effizienz und Sicherheit für Roboter-Lernalgorithmen und macht eine Hochgeschwindigkeitsaufgabe zu einem idealen Maßstab, um Roboter-Lernsysteme hervorzuheben. Um das Lernen auf dem physikalischen System zu erreichen, schlagen wir ein Lernsystem vor, das die Anforderungen an Sicherheit und Sample-Effizienz direkt in den Entwurf der Darstellung, der Initialisierung und der Optimierung der Policy einbezieht. Dieser Ansatz steht im Gegensatz zu früheren Arbeiten, die sich hauptsächlich auf die Details der Lernalgorithmen konzentrieren, aber die technischen Details vernachlässigen. Wir zeigen, dass dieses System es dem Hochgeschwindigkeits-Barrett WAM ermöglicht, das Jonglieren mit zwei Bällen aus 56 Minuten Erfahrung zu lernen. Der Roboter lernt das kontinuierliche Jonglieren allein auf der Grundlage eines binären Belohnungssignals. Die optimale Policy ist in der Lage, bis zu 33 Minuten oder für etwa 4500 wiederholte Fänge zu jonglieren. Die Videos, die den Lernprozess und die Auswertung dokumentieren, finden Sie unter **https://sites.google.com/view/jugglingbot**

Acknowledgments

On IAS side I would like to thank my supervisor Michael Lutter. Especially towards the end of this project he invested many ours to help me set up and record experiments. Also I would like to thank Dorothea Koert for co-supervising earlier projects leading up to this thesis, helping to break down the task and set realistic goals.

I would also like to thank all Students working together with me in the IAS student room "D-Lab" for many interesting discussions and foremost Quentin Delfosse, who also helped 3D-printing necessary hardware components.

I am very grateful towards the Rhein-Main West Cost Swing dance community, above all towards Christine Kruse and Carina Klaas. While supporting me emotionally, both made sure I would take occasional breaks and I have experienced some of the most joyful moments with them.

Finally I would like to thank Lutz Marquardt for sparking my interest in mathematics and constantly challenging me with new problems, thereby setting my path towards scientific research and robotics.

Contents

1. Introduction

Learning robots are one promising approach to overcome the stiff and pre-programmed movements of current robots. When learning a task, the robot autonomously explores different movements and improves its own behavior using scalar rewards. In recent years, research has focused a lot on improving task-agnostic deep reinforcement learning algorithms by either changing the optimization [1, 2, 3], randomizing the simulation's physics parameters [4, 5, 6] or gradually increasing the task complexity [7, 8]. While these approaches have propelled learning robots to very complex domains in simulation ranging from full-body control of humanoids [9] to control of dexterous hands [10, 11], most of these approaches are not applicable to learn on the physical system as they neglect the intricate complexities of the real world.

Consider the high-acceleration task of juggling two balls with a single anthropomorphic manipulator. The manipulator is required to throw a ball up, move to the right, catch and throw the second ball and return in time to catch the first ball. To achieve the cyclic juggling pattern repeatedly, the robot must always throw the ball sufficiently vertical and, maintain the precise timing of motions. Therefore, this task pushes the limits of the robot to achieve the required high accelerations (of up to $8g$), while maintaining precise control of the end-effector and the safety of the physical system. The task is not only inherently difficult to master[1] but also requires learning on the physical system. Real world experience is required, as the simulation models — while good at simulating contact-free rigid-bodies — cannot represent the non-linear effects encountered at the torque limits of the actuators and the ball impacts on the end-effector. For the tendon driven Barrett WAM, rigid-body-simulators also cannot model the dominating cable dynamics at high accelerations. Therefore, simulation-based solutions cannot be transferred for very dynamic tasks.

[1] For reference, the untrained human jugglers of the lab achieve 2 repeated catches and improve to about 20 repeated catches after a few hours of training.

In addition, the high accelerations also amplify the peculiarities of safety and sample efficiency required for learning on the physical system. Collisions at high velocities would have severe consequences for physical systems. The sample efficiency is important as running the system for days is undesirable, since the high accelerations cause too much wear and tear and achieving the the full environment automation is too laborious for most dynamic tasks. To automate the juggling environment, one would have to devise a mechanism to pick-up two balls, put one ball in the end-effector and one ball in the release mechanism at the ceiling. These requirements are in contrast to the other reinforcement learning benchmark of contact and perception-rich manipulation. The manipulation environments can be engineered to be safe due to low velocities and can be easily automated to enable large scale data collection. Therefore, the task of robot juggling is an ideal benchmark to highlight the challenges of learning on the physical system as it requires increased sample efficiency, safe policies and safe exploration.

To emphasize the necessary requirements for building a real world learning system for dynamic tasks, we present an approach that directly incorporates sample efficiency and safety for the physical system within the system design. We start with the application of robot juggling and design a robot learning system for this task, that is simple, efficient and applicable to the real world. To achieve these requirements, we directly incorporate engineering and task expertise within the policy representation, initialization and optimization. While these engineering decision are specific to the task, we extensively motivate our decisions and relate the chosen approach to prior works. Therefore, the reasoning transfers to many dynamic tasks. This is in contrast to many prior works, which mainly proposed new policy representations or policy optimizations and demonstrated that new algorithm can be applied to the physical system. Instead, we start with the task and focus on combining existing representations and algorithms to achieve a safe and sample efficient robot learning system. Afterwards, we validate the learning system by demonstrating the learning of juggling with two balls and a single anthropomorphic manipulator in the real world. Therefore, the contribution of this work is the application of robot learning to the challenging task of single arm juggling with two balls and highlighting the necessary engineering and task expertise to build a learning system in the physical world.

This thesis is structured as follows: First we cover the prior work on robot juggling and learning on the physical robot (Chapter 2) and provide a brief introduction to juggling in general (Chapeter 3). Afterwards, we describe our proposed learning system (Chapter A) and the design of the environment (Chapter 5) to achieve successful learning. Finally, we demonstrate learning in the physical world (Chapter 6) and summarize the results (Chapter 7) before giving an outlook on future work (Chapter 8).

2. Background and Related Work

2.1. Robot Juggling

For decades robot juggling has been used to showcase the ingenuity of mechanical system design and control engineering. Starting with Claude Shannon in the 1970s, many different juggling machines were built.[1] For example, the Shannon juggler used cups to bounce up to five balls of a surface [12], the devil-sticking machine stabilized a stick in the air [13, 14], paddle jugglers, built from designated hardware [15, 12] or by attaching tennis rackets to manipulators [16, 17, 18, 19, 20, 21], juggled multiple balls using paddling, toss jugglers were built using manipulators to juggle one [22, 23] or two balls [24] and even humanoids were used to juggle up to three balls with both arms [25, 26]. Most of these approaches proposed new engineering solutions for movements and controllers, showing that these system achieve juggling when the control parameters are manually fine-tuned. Only few approaches used supervised learning for model learning [17, 14], behavioral cloning [21] or evolutionary strategies with dense fitness functions [27] to achieve paddle juggling and devil sticking. In this work we build upon the vast experience on end-effectors and controller design for robot juggling but in contrast to prior work, we demonstrate, to the best of our knowledge, the first robot learning system that learns toss juggling with two balls and a single anthropomorphic manipulator in the real world, using only binary rewards. Using this approach, we achieve juggling of up to 33 minutes and high repeatability between trials.

[1] A historic overview of juggling robots can be found at `https://www.youtube.com/watch?v=2ZfaADDlH4w`.

Juggling Type	Approach	Papers
Devil Sticking	Engineered	[12]
Devil Sticking	Model Learning	[14]
Paddle Juggling	Engineered	[18, 19, 12, 20, 15]
Paddle Juggling	Imitation	[21]
Paddle Juggling	Model Learning	[17]
Paddle Juggling	Evolutionary Strategies	[27]
Toss Juggling	Engineered	[24, 26, 25, 23, 22]
Toss Juggling	**Reinforcement Learning**	[**Ours**]

Table 2.1.: Prior work on different types of robot juggling

2.2. Learning on the Physical Robot

Despite the recent surge of deep reinforcement learning algorithms for controlling robots, most of these approaches are constrained to learn in simulation, due to sample complexity and the high risk of catastrophic policies. Only Schwab et al. [28] and Levine et al. [29] achieved learning on the physical system by engineering fully automated environments to achieve large-scale data collection and engineering classical safety mechanisms to avoid damaging the physical system. Using these safe and automated environments, deep reinforcement learning was able to learn Ball-in-a-Cup [28] and object grasping with pinch grippers [29] from raw pixels. Most other learning approaches for physical systems use more classical robot learning techniques, that combine engineering- and task knowledge with learning to achieve sample efficient and safe learning, that does not require completely safe and fully automated environments. For example, combining model learning with trajectory optimization/model predictive control [30, 11, 14] or model-free reinforcement learning with engineered policy representation, expert policy initialization and dense rewards [31, 32, 21, 33, 34]. In our work, we extend the classical robot learning approach to a robot learning system that learns the high acceleration task of juggling with designed feature representations, expert policy initialization and binary rewards instead of dense rewards. We also focus on the necessary design decisions that incorporate engineering and task expertise to achieve safety and sample efficiency and

relate our decisions to the previously mentioned prior works. This focus is in contrast to most prior work, as these mostly highlighted the details of the learning algorithms, but not the many engineering details that enabled learning on the physical system.

Task	**Policy**	**Init**	**Reward**	**Learning**
Baoding Balls [11]	FF-NN	Random	Dense	MBRL
Ball in a Cup [28]	CNN	Random	Dense	MFRL
Object Grasping [29]	CNN	Random	Binary	MFRL
Ball in a Cup [32]	DMP	Expert	Dense	MFRL
Table Tennis [21, 33, 34]	DMP	Expert	Dense	MFRL
Pancake Flipping [31]	DMP	Expert	Dense	MFRL
Helicopter Flight [30]	PM	Expert	Dense	MBRL
Toss Juggling (Ours)	**VP**	**Expert**	**Binary**	**MFRL**

Table 2.2.: Prior work on robot learning using model-free reinforcement learning (MFRL) and model-based reinforcement learning (MBRL) in the physical world.

3. A Mathematical Perspective on Juggling

While the main contribution of this work is in the design and implementation of a basic juggling task, we would like to give a brief introduction to the mathematically well understood topic of toss juggling, to give the reader a more sophisticated understanding about juggling task and spark ideas for future projects.

Initiated in the 1970s by Claude Shannon at MIT, a surprisingly large body literature offering mathematical insights developed as many jugglers do have a mathematical background. Shannon [35] described the relationship between the number of balls b, the number of hands h, the dwell time d that a ball spends in the hand between being caught and thrown, the flight time f and the vacant time v that a hand stays empty

$$\frac{f+d}{v+d} = \frac{b}{h},$$

showing how jugglers trade off time the balls spend in the air against time they spend in the hands.[1] In the following, we introduce the nowadays widespread notation of juggling sequences.

Juggling sequences are often attributed to Tiemann et al. [36] and their properties are described in detail in the works of Polster [37], which this chapter largely builds upon. To define a juggling pattern by a juggling sequence, three assumptions about the pattern have to be made.

1. The balls are juggled to a constant equidistant beat.
2. Patterns are periodic.
3. At most one ball is caught and thrown on every beat.

[1] Ironically, given Shannon's outstanding digital innovations, this theorem is analog.

Consequently, a time discrete juggling state can be defined by the Boolean string, that encodes at which future beats a ball will be caught. Consider the case of juggling three balls, as depicted in Figure [3]. In state 1110 the three balls will be caught one by one during the next three beats and the forth beat is left open, as no ball has yet been thrown, such that it would be caught at that moment. Throwing the first ball in a manner, that it will be caught after three beats, pushes it in the first free time slot, preserving the current juggling state. A throw of height 4 moves the current ball to the second open time slot, leaving a single beat gap and therefore resulting in the state 1101. Throwing a 2 will fill the gap and bring the pattern back to the initial state 1110.

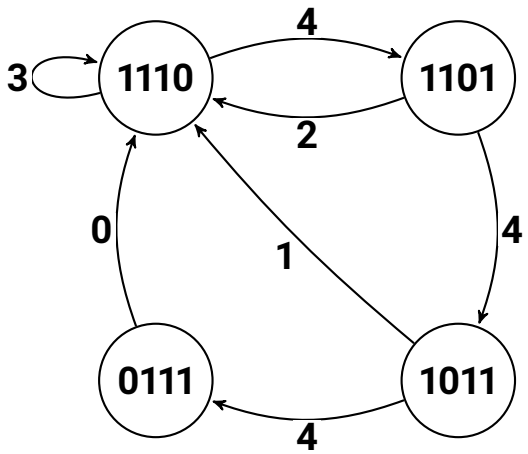


Figure 3.1.: The 3-ball juggling state graph of height 4. The states are defined by Boolean strings encoding at which future beats a ball will be caught and the transitions correspond to the height of the next throw. Every closed loop in the directed graph is a juggling sequence corresponding to a distinct juggling pattern.

Excluding transitions that would move a ball to a time-slot, which is not free and limiting the height of the throws, a juggling state graph can be constructed. Every closed loop in this directed graph is a juggling sequence corresponding to a distinct juggling pattern. For three balls the most common juggling patterns are $\{3\}$, $\{4, 2, 3\}$ and $\{4, 4, 1\}$. A juggling simulator, that visualizes any valid juggling sequence can be found at http://www.gunswap.co/. Note that juggling sequences are mostly written in simplified form, eg. 423 or 441.

Juggling sequences offer properties and operations, that render juggling an appealing task to demonstrate efficient exploration strategies. Letting $s = \{a_k\}_{k=0}^{p-1}$ be a juggling sequence of length p, the number of required balls b is the average value of the elements in the juggling sequence. If b is not an integer, the sequence is not a valid juggling sequence.

$$b = \frac{\sum_{k=0}^{p-1} a_k}{p}$$

The number S of distinct b-ball juggling patterns of length p is

$$S(p, b) = (b + 1)^p - b^p.$$

This in property can be used to easily benchmark exploration strategies against the theoretic maximum of different patterns. Any b-ball base pattern of length p can be discovered using permutations called site swap. The heights of two throws a_i and a_j are changed according to

$$a_i \to a_i - j + i$$
$$a_j \to a_j + j - i$$

such that the beats, at which the thrown balls are caught, switch. For example performing a site swap on throws $i = 1$ and $j = 2$ on the juggling sequence $\{3, 3, 3\}$ results in $\{4, 2, 3\}$. Applying the same site swap again results in $\{4, 1, 4\}$, which corresponds to the same pattern as $\{4, 4, 1\}$, since juggling sequences are invariant to cyclic shifts.

The juggling sequence, which the robot learns in this work, is $\{4, 0\}$. Assuming two existing end-effectors, it can be seen, that two balls are thrown and caught by the first end-effector, while the second end-effector is not used. If a second end-effector independently performed the same task as the first one, it would result in the juggling sequence $\{4, 4\}$, which is the four ball base pattern. Therefore, jugging two balls with a single end-effector is equally difficult as juggling four balls in two end-effectors and obviously significantly more difficult than juggling three balls.

4. Approach

In the following, we describe the learning system used to archive juggling on the real system. First, the low dimensional policy representation achieving safe movements (4.1) and the corresponding initialization strategy (4.2) is introduced. Afterwards, the optimization procedure to obtain the optimal policy parameters (4.3) is presented.

4.1. Policy Representation

The choice of the policy representation determines the sample efficiency and the safety of the executed movements. While the number of learnable policy parameters defines the dimensionality of the search space hence, directly relates to the required samples to find the optimal policy, the space of possible policies also determines the range of possible motions, which impacts the safety of the learning procedure. High capacity representations, such as deep networks, can represent arbitrary, discontinuous and unbounded functions including high-jerk trajectories, that are not necessarily feasible for the kinematic structure. In contrast, constrained representations can be designed to describe only bounded and low jerk torque sequences that guarantee stability for every possible parameter configuration. To achieve stable movements and high sample efficiency, we define the probabilistic policy to be a normal distribution over joint-space via-points and execute these via-points using a stable tracking controller. Therefore, this policy is safe for every parameter configuration as long as the via-points are feasible. This feasibility can be easily achieved by clipping the via-points to the joint limits with an additional safety margin. The chosen policy is similar to a DMP policy [32, 31, 21, 33, 34], which also achieves stable motion. However, we rather chose via-points due to the interpretability, as this interpretability enables to intuitively incorporate expert knowledge with the initialization in the absence of kinesthetic demonstrations.

Let the policy parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ define a multivariate Gaussian $\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ over the parameter space Ω. Each point in parameter space describes a set of via-points and timings of a possible juggling movement, i.e., $\theta = [\mathbf{q}_0, \dots \mathbf{q}_N, \dot{\mathbf{q}}_0, \dots, \dot{\mathbf{q}}_N, t_0, \dots, t_N]$ with the joint position $\mathbf{q}$, joint velocity $\dot{\mathbf{q}}$ and timing t. These via-points are transformed to torques $\boldsymbol{\tau}$ by interpolating between the via-points using cubic splines and applying a PD controller with gravity compensation, i.e.,

$$\mathbf{q}_{\text{ref}}(t) = \sum_{n=0}^{3} \boldsymbol{a}_n (t - t_0)^n, \;\; \dot{\mathbf{q}}_{\text{ref}}(t) = \sum_{n=1}^{3} n \boldsymbol{a}_n (t - t_0)^{n-1}$$

$$\boldsymbol{\tau} = \mathbf{K}_{\text{P}}(\mathbf{q}_{\text{ref}} - \mathbf{q}) + \mathbf{K}_{\text{D}}(\dot{\mathbf{q}}_{\text{ref}} - \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}).$$

The spline parameters are computed by

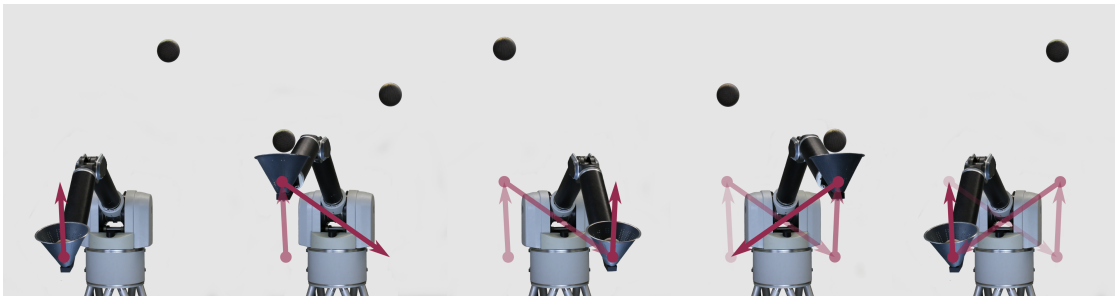
$$\begin{aligned} \boldsymbol{a}_{i,0} &= \boldsymbol{q}_i \quad \boldsymbol{a}_{i,1} = \dot{\mathbf{q}}_i \\ \boldsymbol{a}_{i,2} &= 3 \; (\boldsymbol{q}_{i+1} - \boldsymbol{q}_i)\, t_i^2 - (\dot{\boldsymbol{q}}_{i+1} + 2\dot{\boldsymbol{q}}_i)\, t_i \\ \boldsymbol{a}_{i,3} &= 2 \; (\boldsymbol{q}_i - \boldsymbol{q}_{i+1})\, t_i^3 + (\dot{\boldsymbol{q}}_{i+1} + \dot{\boldsymbol{q}}_i)\, t_i^2. \end{aligned}$$

The control gains $\mathbf{K}_{\text{P}}$ and $\mathbf{K}_{\text{D}}$ are set low compared to industrial manipulators to achieve smooth movements.

This policy is only closed-loop w.r.t. the robot state but does not depend on the full environment state, which includes the ball positions and velocities. We chose this representation as prior work showed the stability of such movements for juggling [12] and including feedback-control on the ball state would significantly increase the complexity. When incorporating the feedback on the ball, one has to develop a representation that can adapt to the ball position. However, engineering such representation is non-trivial as solely adapting the catching movement is not sufficient, as one would also need to adapt the consecutive throwing to the previous catch and prevent the system from drifting. In addition, one would need to observe the complete system state using visual trackers, which are prone to artefacts causing large jumps between time steps and hence, one would need to guarantee, that the policy or the tracker is robust to outliers to achieve safe execution.

4.2. Policy Initialization

Besides the dimensionality of the policy representation, the initialization of the policy parameters significantly affects the sample efficiency of the learning. Initializing the parameters close to the optimal solution reduces the amount of update steps to achieve the optimal parameters and hence, decreases the sample complexity. Especially, in the case of binary rewards, the reward signal would be very sparse without good initialization as large areas of the parameter space do not yield any contact with the balls and hence, would yield no information about the desired task. The policy initialization is commonly achieved by recording kinesthetic demonstrations and fitting the policy parameters to match the demonstrations, e.g., [32, 31, 21, 33, 34] used demonstrations to initialize DMP parameters. However, for the dynamic task of robot juggling, kinesthetic demonstrations are not applicable as the human demonstrator cannot achieve the necessary accelerations on the real robot. Instead, we leverage the interpretable nature of the policy representation and use task knowledge to construct an approximate juggling movement, that significantly reduces the search space. Exploring all possible joint-space via-point sequences is not feasible on the physical system.



(a) (b) (c) (d) (e)

Figure 4.1.: The juggling movement consisting of four separate movements, which are repeated to achieve juggling of two balls with a single anthropomorphic manipulator.

The juggling movement for two balls consists of four repeated movements, as shown in Figure 4.1 — i.e., (1) throwing the first ball, (2) catching the second ball (3) throwing the second ball and (4) catching the first ball (Fig. 4.1). We define the switching points between these movements as the via-points of the policy and repeat these via-points to achieve a limit-cycle. To start the cyclic pattern an initial stroke based movement is

prepended describing a slightly different first cycle. This separate movement is necessary to quickly converge to the limit cycle without dropping a ball. Furthermore, we introduce domain knowledge to enforce symmetry between throwing movements and zero velocity at the switching points to reduce the search space dimensions. These additional constraints reduce the effective dimensionality from 80 parameters to 21 parameters. All learned parameters and constraints are listed in Appendix A.

4.3. Policy Optimization

To obtain the optimal policy π^*, reinforcement learning explores the parameter space and improves the policy to achieve maximum reward. Deep reinforcement learning approaches, including [28, 29], explore the policy space by applying noisy actions at each time step and update the policy to maximize an approximated Q-function. This approach is often not directly suitable for safe and sample efficient learning on the real system as the step-based exploration in action space can cause excessive jerk and learning the correct reward attribution to each action requires many samples. Learning the Q-function for the juggling task is especially hard due to the long delays between bad actions and dropping a ball. Furthermore, the complete system state would need to be observed. Due to these shortcomings of step-based exploration and step-based reinforcement learning for this specific juggling task, we chose to frame the policy optimization as episodic exploration and episodic reinforcement learning problem. The episodic exploration in parameter space samples a policy parameter vector, i.e. $\boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and evaluates the episodic reward of this parameter vector. For the physical system this exploration strategy is favorable because it yields smooth and stable action sequences given the previous policy representation. The episodic reinforcement learning is favorable, as one does not need to learn the Q-function and only the episodic reward is required. Hence, the precise state does not need to be observed. Similarly to our approach, Kober et al. [32] and Kormushev et al. [31] use an episodic expectation-maximization based policy search for ball in a cup and pancake flipping. Instead of EM-based policy search, we use the information theoretic policy search approach of episodic relative entropy policy search (eREPS) [38, 39] as this approach adds an additional Kullback-Leibler (KL) divergence constraint between two consecutive policies to prevent pre-mature convergence and large, possibly unsafe, jumps of the policy.[1]

[1] Further information about reinforcement learning for robotics can be found in the surveys Kober et al. [40] and Deisenroth et al. [38].

Let the optimization problem of eREPS be defined as

$$\pi^{'} = \arg\max_{\pi} \int_{\Omega} \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta}, \quad \text{s.t. } d_{\mathrm{KL}}(\pi^{'} || \pi) \leq \epsilon$$

with the updated policy $\pi^{'}$, the episodic reward R and the KL-divergence d_{KL}. With the additional constraint of π being a probability distribution, this optimization problem can be solved by first optimizing the dual to obtain the optimal Lagrangian multiplier η^* and fitting the new policy using weighted maximum likelihood. The sample-based optimization of the dual is described by

$$\eta^* = \arg\min_{\eta} \eta\epsilon + \eta \log \sum_{i=0}^{N} \pi(\boldsymbol{\theta}_i) \exp\left(R(\boldsymbol{\theta}_i)/\eta\right).$$

The optimization of the likelihood $\mathcal{L}$ to obtain the updated policy is described by

$$\boldsymbol{\mu}^{'},\ \boldsymbol{\Sigma}' = \arg\max_{(\boldsymbol{\mu},\, \boldsymbol{\Sigma})} \sum_{i=0}^{N} \exp\left(R(\boldsymbol{\theta}_i)/\eta^*\right)\ \log(\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \boldsymbol{\theta}_i)).$$

For the considered multivariate Gaussian policy distribution this optimization simplifies to

$$\boldsymbol{\mu}' = \sum_{i=0}^{N} w_i\, \boldsymbol{\theta}_i, \quad \boldsymbol{\Sigma}' = .\frac{1}{z} \sum_{i=0}^{N} w_i\ \left(\boldsymbol{\theta}_i - \boldsymbol{\mu}'\right) \left(\boldsymbol{\theta}_i - \boldsymbol{\mu}'\right)^T$$

with $w_i = \exp\left(R(\mathbf{w}_i)/\eta^*\right) / \sum \exp\left(R(\mathbf{w}_i)/\eta^*\right)$ and $z = (1 - \sum w_i^2)^{-1}$.

5. Experimental Setup

5.1. Robot

For the experiments the high-speed manipulator Barrett WAM with 500Hz torque control is used. We use the 4 degree of freedom (DoF) variant, as the wrist joints of the 7 DoF version become uncontrollable at very high accelerations with the end-effector attached. When trying to throw, the wrist snaps back and these uncontrollable movements prevent precise control of the throwing movements. For the simulation studies, MuJoCo [41] (Figure 5.1) was used to simulate the rigid-body-dynamics of the robot. Therefore, the non-linear cable drives are neglected in simulation. For the contacts, only the cone shaped end-effector is collidable and high damping constants for the balls and the end-effector stabilize the balls during the rebound phase. The simulator is available at **https://github.com/kploeger/mujoco_robots**

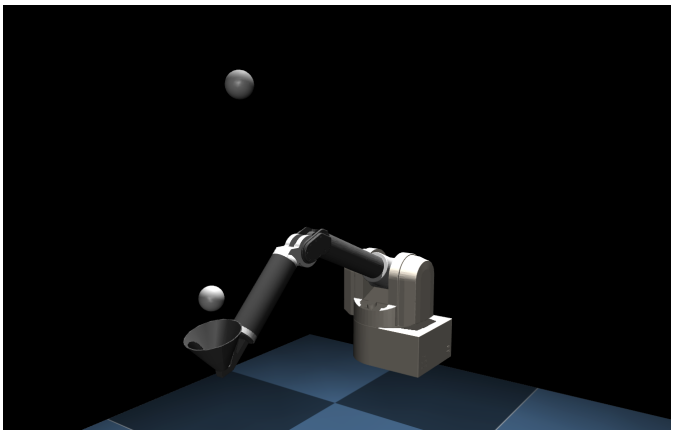


Figure 5.1.: The MuJoCo juggling environment used for simulating the rigid-body-dynamics.

5.2. End-effector Design & Juggling Balls

To achieve long juggling durations, the balls must be thrown sufficiently vertical to ensure subsequent catching. Even minor differences of the ball position and velocity relative to the end-effector can cause very different throws. To achieve high repeatability, we use hardware designed to passively dissipate the kinetic energy of the balls and thereby prevent bouncing as well as funnel-shaped end-effector, which slides the balls to the identical position on each catch, as successfully done in prior toss juggling approaches [26, 25, 23, 22]. The strong dampening is achieved by using 75mm Russian style juggling balls [42], which consist of a hollow plastic shell partially filled with 37g of sand resulting in a total weight of 50g. The sand disperses the kinetic energy of the ball via friction when hitting an obstacle and hence, prevents the ball from bouncing. A similar effect can be achieved using beanbags [23, 22], but these deform during catching and this deformation results in inconsistent throws. In contrast the elastic shells of the used juggling balls retain their shape and ensure consistent throws.

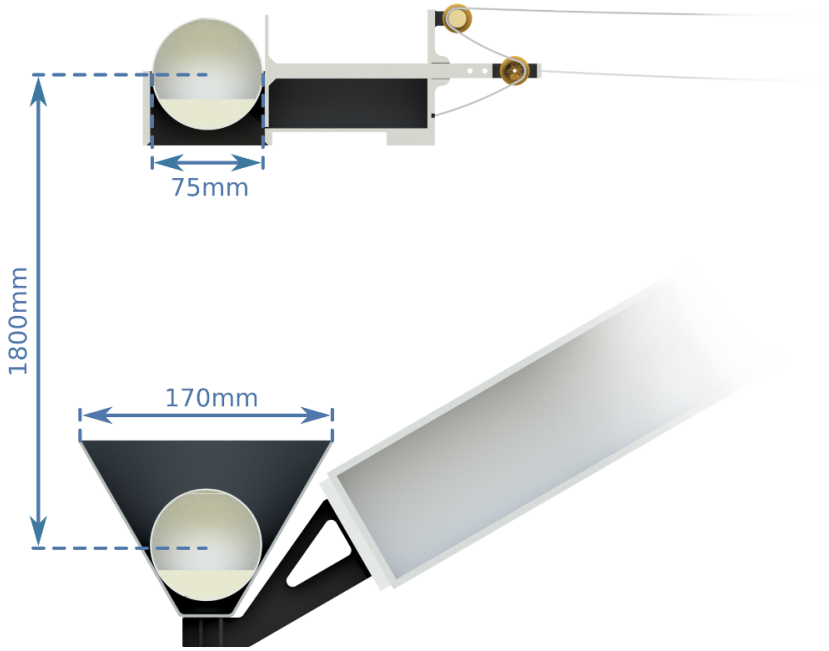


Figure 5.2.: Mechanical design of the funnel shaped end-effector, the partially filled juggling balls and the pulley driven ball launcher.

5.3. Reward Function

The reward function assigns a positive binary reward. as long as the robot is juggling. Juggling is defined as keeping both balls 60cm above the floor, which is measured using the external marker tracking system OptiTrack [43]. Therefore, the step based reward is described by

$$r(t) = \begin{cases} 1 & \text{if } b_z \geq 0.6 \\ 0 & \text{if } b_z < 0.6 \end{cases}$$

with the ball height b_z. This reward signal maximizes the juggling duration and is not engineered to incorporate any knowledge about the desired ball- or manipulator trajectory. This choice of reward function is intuitive but also uninformative, as a bad action only causes a delayed negative reward. For example, a bad action within the throwing will cause a zero reward seconds after the action. This delay between action and reward, i.e., 'credit assignment', is a challenge in many RL problems.

From a technical perspective, these binary rewards are also favorable, as they do not require precise tracking of the juggling balls, as one only requires to detect a ball beneath a threshold. To evaluate a dense reward function, the precise tracking of the balls is required. This precise tracking of the individual balls is challenging due to the required robust filtering. Furthermore, the reflective tape of the balls degrades fast due to frequent contacts, which significantly reduces the tracking precision.

The choice of the binary reward functions is in stark contrast to prior work, as most of the previously proposed approaches use dense rewards. Schwab et al. [28] combine two shaped dense rewards with five binary rewards triggered at engineered task configurations for ball-in-a-cup, Levine et al. [11] use the distance to desired state at every time-step and a binary survival bonus for Baoding balls, Kober et al. [32] use the exponential weighted Mahalanobis distance to the desired configuration at an engineered time step for ball-in-a-cup and Schaal et al. [27] use engineered characteristics of the end-effector and ball trajectory as dense reward for paddle juggling. These dense rewards significantly simplify the learning problem, but also require extensive engineering. The parameters must also be tuned to obtain the desired behavior.

5.4. Environment Standardization & Automation

The physical world is inherently noisy due to the imperfect mechanical systems and actuators, the inaccurate resetting of the environment and many unobserved environment states. In particular, for juggling, minor differences in the robot initialization and the release of the second ball can make the identical policy juggle perfectly or miss the ball on the first try. Hence, minimizing the stochasticity of the environment is essential to achieve successful reinforcement learning on the physical system. While reinforcement learning algorithms can learn using stochastic samples, the sample complexity increases as the expected reward must be approximated using many more samples. This increase in sample complexity can render learning on the physical system unfeasible. Therefore, one wants to create an environment that enables consistent repetitions of the same experiment by removing as much variance as possible. Ideally, one engineers a fully autonomous environment that can, in addition to consistent samples, reset itself to generate large sample sizes over extended durations, e.g., [29, 28] built fully automated environments to feed their deep policy networks with samples for days or weeks. However, for many tasks building a fully autonomous environment is more challenging than the original task. For juggling, a fully automated environment would need to pick up the balls from the ground, put one ball in the end-effector and release the second ball from the ceiling. Building such an environment is as challenging as engineering the juggling robot and hence, undesirable.

For the juggling task, the main cause of variance is the release of the second ball. The second ball must be released with identical height, velocity and timing to achieve consistent initialization. To achieve precise control of the initial ball position and velocity, we designed and mounted a launching mechanism 3m above the floor, shown in Figure 5.2. This release mechanism releases the ball by pushing the ball to an opening, using a piston attached to a pulley system. To achieve the correct timing, the release of the ball is detected using OptiTrack [[43]] and this detection starts the episode. The resetting of the environment, i.e., picking up the fallen balls and resetting the release mechanism, is performed manually and accounts for most of the total training time.

6. Evaluation

In the following, the learning of the optimal policy on the physical system is described and the optimal deterministic policy is evaluated w.r.t. repeatability and duration of the juggling. For comparison, the learned policy is benchmarked against a hand-tuned policy. Videos documenting the learning process and the evaluation can be found at **https://sites.google.com/view/jugglingbot**.

6.1. Simulation Studies

The initial validation of the proposed learning system is performed in simulation to evaluate the convergence of different seeds and different number of roll-outs per episode. Figure 6.1 shows the mean juggling duration distribution of the final policy over 60 different seeds and 10, 25 and 50 roll-outs per episode. For 10 roll-outs per episode, the learning system frequently converges to a sub-optimal final policy, which does not achieve consistent juggling of 10 seconds. The probability of converging to a good policy is the highest for 25 roll-outs per episode and hence, we use 25 roll-outs per episode on the physical system. It is important to point out, that learning juggling in the simulation is more challenging compared to the real world as the dampening and friction of partially filled juggling balls had to be approximated for computational efficiency.

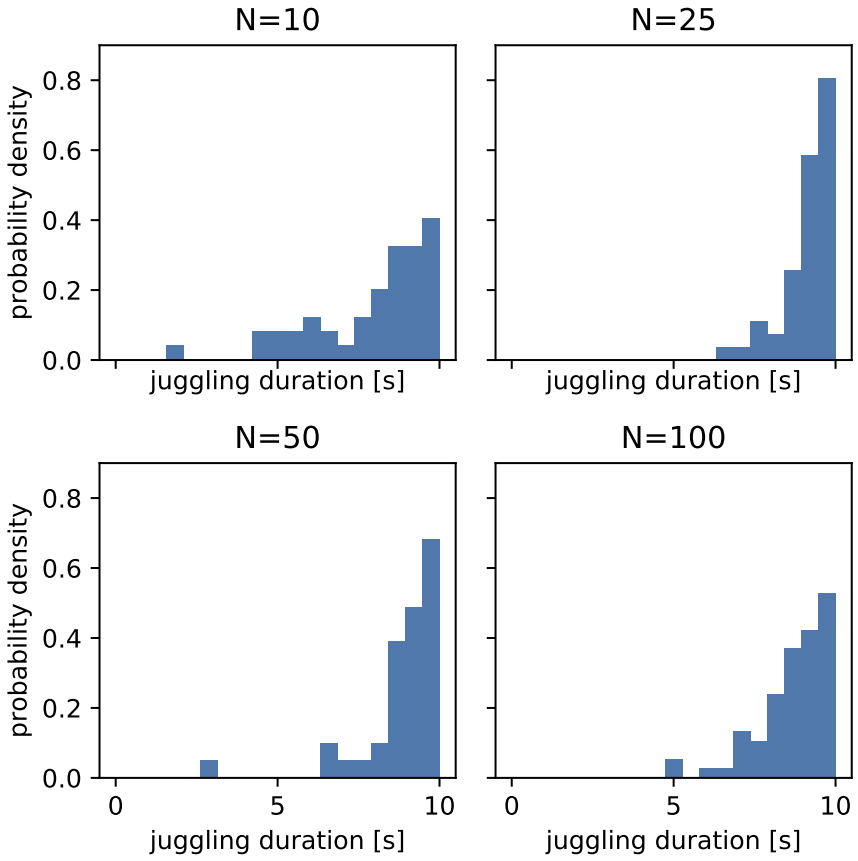


Figure 6.1.: The distribution of the mean juggling durations of final policies learned with varying batch sizes N over 60 different seeds. The maximal juggling duration is 10s. At $N = 25$ the policy converges to optimal performance most reliably, therefore we choose this batch size for experiments on the physical system.

6.2. Learning on the Barrett WAM

For the learning on the physical Barrett WAM 20 episodes were performed. During each episode 25 randomly sampled parameters were executed and the episodic reward evaluated. If the robot successfully juggles for 10s, the roll-out is stopped. Roll-outs that were corrupted due to obvious environment errors were repeated using the same parameters. Minor variations caused by the environment initialization were not repeated. After collecting the samples, the policy was updated using eREPS with a KL constraint of 2. The learning progress is shown in Figure 6.2. Initially the robot on average achieves between 1 to 3 repeated catches. These initial catches are important as the robot would otherwise not receive any information in the rewards. Therefore, the rudimentary expert initialization must yield some roll-outs with repeated catches to ensure fast learning. After the initial episodes, the robot rapidly improves the juggling duration. Starting from episode 10, the robot achieved consistent juggling of 10 seconds in the majority of the roll-outs and only very few balls dropped during the start. In the next 10 episodes, the average reward oscillates as the number of dropped balls varies, but the robot achieves successful completion for the other trials. At episode 20 the robot achieves perfect juggling of all 25 randomly sampled parameters.

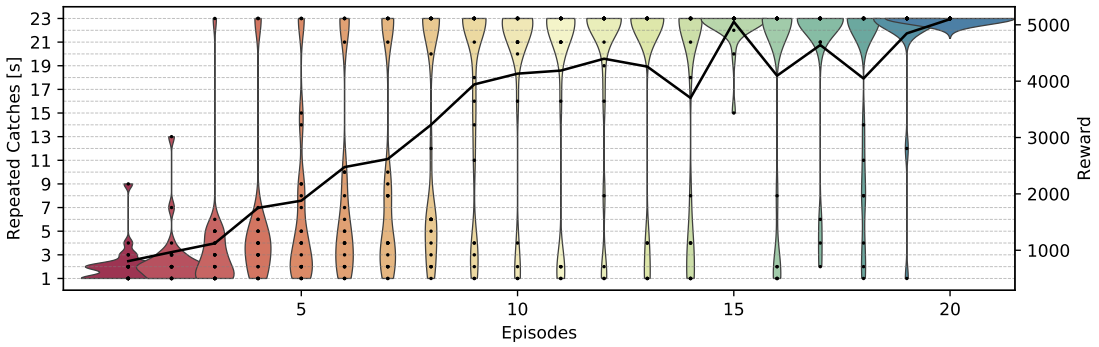


Figure 6.2.: Reward distributions for learning to juggle on the physical system. Each point corresponds to a single roll-out on the system. Starting from the expert initialization, which only achieves juggling for a few repeated catches, the robot continuously improves using only binary rewards. After 56 minutes of training, the final policy achieves consistent juggling for more than 10s.

6.3. Juggling Repeatability and Duration

To test the repeatability and stability of the learned policy, the policy mean of episode 20 is deterministically executed for 30 repeated roll-outs with a maximum duration of 2 minutes. The achieved performance is compared to a hand-tuned policy. The distributions of the juggling duration is shown in Figure 6.3. The learned policy performs significantly better compared to the hand-tuned policy. The learned policy achieves an average duration of 106.82s and juggles 27 trials for more than 90 seconds. The hand-tuned policy only achieves an average duration of 66.51s and juggles only 16 trials for more than 90 seconds. The hand-tuned policy fails more often than the learned policy to initiate the juggling movement. The weaker performance of the hand-tuned policy within the stroke based initiation movement matches our expectations, as the parameters of the stroke based movement are the hardest ones to tune. Both policies do not achieve perfect repeatability due to the stochasticity of the real world environment.

To test the stability of the learned policy, the juggling was repeatedly executed and the maximum juggling duration recorded. The hand-tuned and the learned policy achieve juggling for more than 30 minutes, which corresponds to roughly 4500 repeated catches. The hand-tuned policy required more trials to achieve the 33 minutes while the learned policy achieved this on the second trial after achieving 22 minutes on the first trial. The high number of repeated catches, highlights the precision of the Barrett WAM and the passively stabilizing end-effector design, which once the juggling is initiated successfully, allows to recover from minor variations in the ball trajectories.

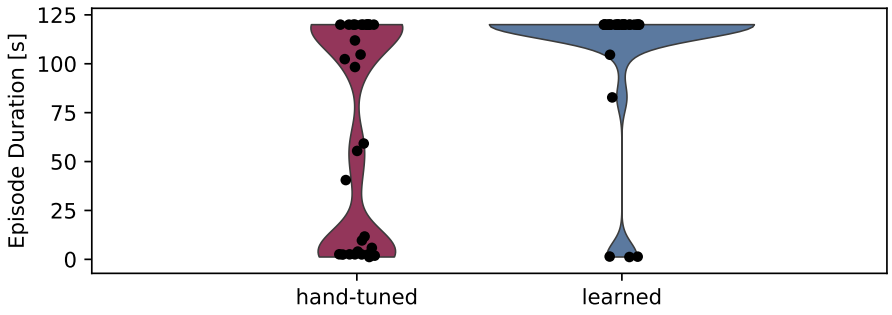


Figure 6.3.: Comparison of the hand-tuned and the learned policy on each 30 roll-outs with maximum duration of 120s on the real system, averaging at 66.52s and 106.82s.

7. Conclusion

In this work, we described a robot learning system capable of learning robot juggling with a single anthropomorphic manipulator and two balls using only binary rewards. We demonstrated that the system can learn this high acceleration task within 56 minutes experience time, when sufficient engineering and task knowledge is used for designing the robot learning system. Starting from a rudimentary expert initialization, the robot consistently improves until the robot achieves consistent repeated juggling of up to 33 minutes, which corresponds to about 4500 repeated catches. Furthermore, the learned policy outperforms a hand-tuned policy in terms of repeatability and achieves significantly higher rewards average across 30 trials. In addition, we highlighted and discussed the incorporated engineering and task expertise to make learning on the physical system viable. This discussion should help future scientists and practitioners to address the needs of a physical system when building future robot learning systems. Nevertheless, this approach also pointed out the shortcomings of state-of-the-art robot learning approaches for learning dynamic tasks on the physical system. Despite the incorporated engineering and task knowledge, the learning still takes up to 5 hours and hence, reiterates the necessity for more sample efficient representations and learning approaches for sparse and binary rewards.

8. Outlook

During this project, only open-loop control strategies were explored on the real system, to solve the task of juggling two balls with a single manipulator. The obvious next step is to investigate further into closed-loop strategies, that adapt to variations of the ball trajectories to solve more complex tasks.

To do so, fist the reliable tracking of the balls has to be solved facing two main problems: (1) The end-effector occludes and strongly accelerates the balls at the same time while catching them and (2) the partial filling of the balls can result in the geometric center of the ball oscillating around center of mass — similar to a spinning juggling club — and therefore increasing the complexity of the dynamics equations compared to balls with homogeneous weight distribution.

The application of closed-loop controllers does not lie in adapting the end-effector trajectory to catch positional outliers in the ball trajectories, which can be achieved by adding a simple task-space controller, but in the effect this adaption has on the next throw. The variance introduced into the subsequent throw has to be smaller than the variance of the caught balls to ensure stability. A significant deviation in the timing of a ball could be stabilized by estimating an approximate juggling state, as described in chapter 3, and subsequent online planning in the juggling state graph to return to the desired path.

Exploration strategies could first be evaluated in the formal domain of juggling sequences, offering discrete state and action-spaces while being arguably more interesting than a typical grid world.

Bibliography

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[3] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[4] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[6] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[7] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions," *arXiv preprint arXiv:1901.01753*, 2019.

[8] P. Klink, H. Abdulsamad, B. Belousov, and J. Peters, "Self-paced contextual reinforcement learning," *Conference on Robot Learning (CoRL)*, 2019.

[9] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, M. Riedmiller, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[10] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[11] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," *arXiv preprint arXiv:1909.11652*, 2019.

[12] S. Schaal and C. G. Atkeson, "Open loop stable control strategies for robot juggling," in *International Conference on Robotics and Automation (ICRA)*, 1993.

[13] S. Schaal and C. G. Atkeson, "Robot juggling: An implementation of memory-based learning," 1994.

[14] S. Schaal and C. G. Atkeson, "Robot juggling: implementation of memory-based learning," *IEEE Control Systems*, 1994.

[15] P. Reist and R. D'Andrea, "Bouncing an unconstrained ball in three dimensions with a blind juggling robot," in *International conference on Robotics and Automation (ICRA)*, 2009.

[16] E. W. Aboaf, C. G. Atkeson, and D. J. Reinkensmeyer, "Task-level robot learning," in *International Conference on Robotics and Automation (ICRA)*, 1988.

[17] E. W. Aboaf, S. M. Drucker, and C. G. Atkeson, "Task-level robot learning: Juggling a tennis ball more accurately," in *International Conference on Robotics and Automation (ICRA)*, 1989.

[18] A. A. Rizzi, L. L. Whitcomb, and D. E. Koditschek, "Distributed real-time control of a spatial robot juggler," *Computer*, 1992.

[19] A. A. Rizzi and D. E. Koditschek, "Further progress in robot juggling: The spatial two-juggle," *Departmental Papers (ESE)*, 1993.

[20] S. Schaal, D. Sternad, and C. G. Atkeson, "One-handed juggling: A dynamical approach to a rhythmic movement task," 1996.

[21] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *International Conference on Robotics and Automation (ICRA)*, 2010.

[22] T. Sakaguchi, Y. Masutani, and F. Miyazaki, "A study on juggling tasks," in *International Workshop on Intelligent Robots and Systems (IROS)*, 1991.

[23] T. Sakaguchi, M. Fujita, H. Watanabe, and F. Miyazaki, "Motion planning and control for a robot performer," in *International Conference on Robotics and Automation (ICRA)*, 1993.

[24] T. Kizaki and A. Namiki, "Two ball juggling with high-speed hand-arm and high-speed vision system," in *International Conference on Robotics and Automation (ICRA)*, 2012.

[25] M. Riley and C. G. Atkeson, "Robot catching: Towards engaging human-humanoid interaction," *Autonomous Robots (AURO)*, 2002.

[26] J. Kober, M. Glisson, and M. Mistry, "Playing catch and juggling with a humanoid robot," in *International Conference on Humanoid Robots (Humanoids)*, 2012.

[27] S. Schaal and D. Sternad, *Learning passive motor control strategies with genetic algorithms*, pp. 913–918. Redwood City, CA: Addison-Wesley, 1993.

[28] D. Schwab, T. Springenberg, M. F. Martins, T. Lampe, M. Neunert, A. Abdolmaleki, T. Herkweck, R. Hafner, F. Nori, and M. Riedmiller, "Simultaneously learning vision and feature-based control policies for real-world ball-in-a-cup," *arXiv preprint arXiv:1902.04706*, 2019.

[29] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research (IJRR)*, 2018.

[30] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems (NeuRIPS)*, 2007.

[31] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *nternational Conference on Intelligent Robots and Systems (IROS)*, 2010.

[32] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems (NeuRIPS)*, 2009.

[33] J. Kober, E. Öztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[34] 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 (IJRR)*, 2013.

[35] C. E. Shannon, "Scientific aspects of juggling," 1993.

[36] B. Tiemann and B. Magnusson, "A notation for juggling tricks. a lot of juggling tricks," *Juggler's World*, vol. 43, pp. 31–33, 1991.

[37] B. Polster, *The mathematics of juggling*. Springer Science & Business Media, 2006.

[38] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, 2013.

[39] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Conference on Artificial Intelligence (AAAI)*, 2010.

[40] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research (IJRR)*, 2013.

[41] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[42] Norwik Juggling Balls. https://www.norwikjuggling.com/index.php/. [Online; accessed 05-Feb-2020].

[43] OptiTrack. https://optitrack.com/. [Online; accessed 05-Feb-2020].

A. Parameter Constraints

Let $q_{i,j}^{(\mathrm{s})}$, $q_{i,j}^{(\mathrm{c})}$ be the joint angles of joint j at via-point i for the stroke movement and the cyclic movement and $t_i^{(\mathrm{s})}$, $t_i^{(\mathrm{c})}$ be the timings at via-point i, then the learned parameter vector is in the space of

$$\begin{aligned}\mathbf{\Omega}_{\text{learn}} = \{&q_{1,0}^{(\mathrm{s})}, q_{1,1}^{(\mathrm{s})}, q_{1,3}^{(\mathrm{s})}, q_{2,0}^{(\mathrm{s})}, q_{2,1}^{(\mathrm{s})}, q_{2,3}^{(\mathrm{s})}, q_{3,0}^{(\mathrm{s})}, q_{3,1}^{(\mathrm{s})}, q_{3,3}^{(\mathrm{s})}, t_1^{(\mathrm{s})}, t_2^{(\mathrm{s})}, t_3^{(\mathrm{s})},\\ &q_{0,0}^{(\mathrm{c})}, q_{0,1}^{(\mathrm{c})}, q_{0,3}^{(\mathrm{c})}, q_{1,0}^{(\mathrm{c})}, q_{1,1}^{(\mathrm{c})}, q_{1,3}^{(\mathrm{c})}, q_{2,0}^{(\mathrm{c})}, q_{3,0}^{(\mathrm{c})}, t_1^{(\mathrm{c})}\},\end{aligned}$$

and the parameter constraints are

$$\begin{aligned}
q_{i,2}^{(\mathrm{s})} &= 0, \text{ for } i \in \{0, \dots 4\},\\
\dot{q}_{i,j}^{(\mathrm{s})} &= 0, \text{ for } i \in \{0, \dots 4\}, j \in \{0, \dots 3\},\\
q_{i,2}^{(\mathrm{c})} &= 0, \text{ for } i \in \{0, \dots 3\},\\
\dot{q}_{i,j}^{(\mathrm{c})} &= 0, \text{ for } i \in \{0, \dots 3\}, j \in \{0, \dots 3\},\\
q_{4,j}^{(\mathrm{s})} &= q_{0,j}^{(\mathrm{c})}, \text{ for } j \in \{0, 1, 3\},\\
q_{2,j}^{(\mathrm{c})} &= q_{0,j}^{(\mathrm{c})}, \text{ for } j \in \{0, 1, 3\},\\
q_{3,j}^{(\mathrm{c})} &= q_{1,j}^{(\mathrm{c})}, \text{ for } j \in \{1, 3\},\\
t_3^{(c)} &= t_1^{(c)} + 0.5T.
\end{aligned}$$