A physics-based Juggling Simulation using Reinforcement Learning

Jason Chemin
Department of Computer Science and Engineering
Seoul National University
jason.chemin@hotmail.fr

Jehee Lee
Department of Computer Science and Engineering
Seoul National University
jehee@mrl.snu.ac.kr

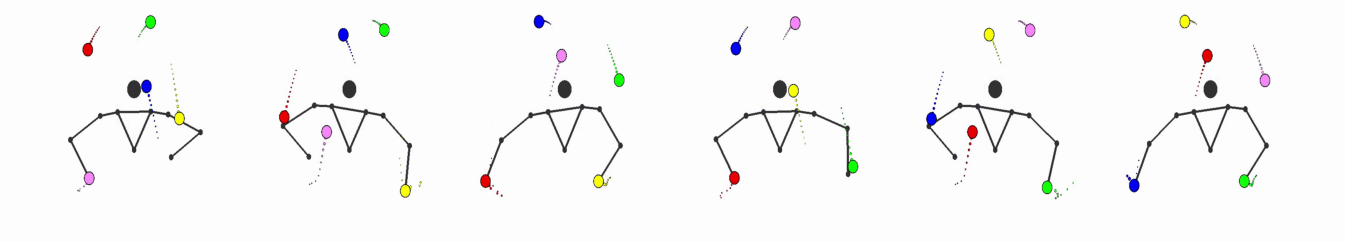


Figure 1: Demonstration of Juggling simulation with 5 balls.

ABSTRACT

Juggling is a physical skill which consists in keeping one or several objects in continuous motion in the air by tossing and catching it. Jugglers need a high dexterity to control their throws and catches which require speed, accuracy and synchronization. The more balls we juggle with, the more these qualities have to be strong to achieve this performance. This complex skill is good challenge for realistic physical based simulation which could be useful for jugglers to evaluate the feasibility of their tricks. This simulation has to understand the different notations used in juggling and to apply the mathematical theory of juggling to reproduce it. In this paper, we present a deep reinforcement learning method for both tasks catching and throwing, and we combine them to recreate the all juggling process. Our character is able to react accurately and with enough speed and power to juggle with up to 7 balls, even with external forces applied on it.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; **Reinforcement learning**;

KEYWORDS

Juggling, Physics-based character animation, Reinforcement Learning

MIG '18, November 8–10, 2018, Limassol, Cyprus

ACM ISBN 978-1-4503-6015-9/18/11…$15.00
https://doi.org/10.1145/3274247.3274516

ACM Reference Format:
Jason Chemin and Jehee Lee. 2018. A physics-based Juggling Simulation using Reinforcement Learning. In *MIG '18: Motion, Interaction and Games (MIG '18), November 8–10, 2018, Limassol, Cyprus.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3274247.3274516

1 INTRODUCTION

The art of juggling has appeared in various cultures in the history as a distracting skill which can sometime be very impressive as told in an ancient Chinese annals, a man who juggled nine balls in front of the enemy armies and won by impressing them with its dexterity. The juggler has to show his control over the timing of his tosses and catches if he wants to keep all objects in the air. Specifically, throwing a ball is a task requiring a balance between accuracy to avoid collision and power to throw the ball high enough. And the more balls we juggle with, the higher or faster we have to throw the ball while keeping track of others in the air. The actual human world record of juggling continuously is eleven balls and as explained by the professional juggler Jack Kalvan [2018], this is physically very difficult to beat.

Research on juggling started with Shannon [1981] when he formulated a mathematics juggling theorem and built the first machine which was able to perform bounce juggling with three balls. Since then, this physical skill has been an interesting challenge in science and in robotics [Beek and Lewbel 1995]. The first juggling machines were designed to make the balls follow a predefined trajectories and it is only recently that we started to use feedbacks from cameras and other sensors to correct trajectory errors while juggling. The Shannon's juggling machine was built with no feedback, repeating the same movement with a constant frequency and avoiding collision by the design of its system. The trajectory of the ball was corrected using groove cups or tracks to redirect it. Many juggling machines were conceived this way, but latest machines started to use feedbacks to learn and to adapt their behavior. The machine named ServoJuggler [Burget and Mezera 2010] can juggle with 5

balls using visual feedback and sensors to detect catches. The humanoid robot created by Kober et al. [2012] is able to juggle with a a human partner using one hand. It uses external cameras to locate the ball and to predict its trajectory. Then it throws it back at the right velocity to the other human participant.

In computer graphics field, a physics-based simulation could help jugglers to create, visualize and evaluate the feasibility of patterns. Some methods can be applied to calculate the movements needed to throw a ball at the right velocity and the position where the ball can be catch, but they can be difficult to apply for real-time. The interactive tool of Jain et al. [2009] is used to manually edit motion to create dynamic scenes with human and object interaction as toss or football juggling. By using a framework for simulation and control of a human musculo-skeletal system, Lee et al. [2018] demonstrates the efficacy of his model by performing a very realistic juggling. His character was able to juggle with up to 6 balls alone and more with two persons juggling together. The Finite Element Method used to calculate the action of the agent needs a preprocessing of several seconds per frame to create the simulation.

The recent research on machine learning provides more possibilities to solve the real-time problem. To demonstrate their method to learn a feedback policy, Ding et al. [2015] learns to keep a ball in the air using a racket and to kick a ball to reach a target. Reinforcement learning combined with neural networks has led to many successes in learning policies to perform complex tasks as playing Atari games [Mnih et al. 2013]. Applied to physically based simulation an agent can learn locomotion behavior in rich environment using Proximal Policy Optimization algorithm [Heess et al. 2017]. Using reinforcement learning to imitate actions in motion clips, Peng et al. [2018] make an agent learn to throw a ball on a target like a Baseball pitch, reproducing human movement from motion capture. Their character is able to throw a ball using the whole body to reach a target.

Our goal is to create an agent on a 2D simulated body ables to learn how to juggle in real time using Deep Deterministic Policy Gradients [Lillicrap et al. 2015]. Juggling is composed of two main subtasks which are catching and throwing a ball. We train two agents to perform these subtasks with one arm. The trained agents can be mirrored on the other arm and we use them sequentially to recreate the all process of juggling.

2 JUGGLING THEORY

We can divide Juggling into 3 subclasses : Roll Juggling where the object will roll over a body or a surface, Bounce juggling as in Shannon's machine which consists in making the object bounce on a surface before catching it, and finally, Toss Juggling which is the most known practice of juggling that we will focus on. It consists in throwing an object with one hand in the air before catching it with the same or the other hand.

2.1 Siteswap notation

Toss juggling can be resume as a sequence of throws and catches from one or two hands at with a different height, timing, direction and power. Therefore, to describe the figures performed and the future sequence of catches and throws, a codification is needed. The most common notation used is named Siteswap. It is a tool to help jugglers to communicate with each other about their tricks and to describe, to share and to discover new patterns and transitions. Some extensions of this notation can describe very complex patterns. In this work, we will use only the basic notation named vanilla Siteswap which describes solo patterns where one object is caught and thrown at a time. As shown in Figure 2, at every periodic beat time, a throw is executed from one hand alternating between the right and left hand. The hand period corresponds to the time between two throws from the same hand and the dwell time is the duration of the hand holding a ball during this period. The dwell time can also be seen as the time available to prepare the throw. Regarding the measurements made on several professional jugglers [Kalvan 2018], the hand period P_{hand} is around 0.55 seconds with three balls down to 0.44 seconds when juggling with seven balls . During this period, the performer has to catch the ball incoming and then needs some time to prepare the throw. The dwell time corresponds to the hand period multiplied by the dwell ratio D_{ratio} which is in average around 70%.

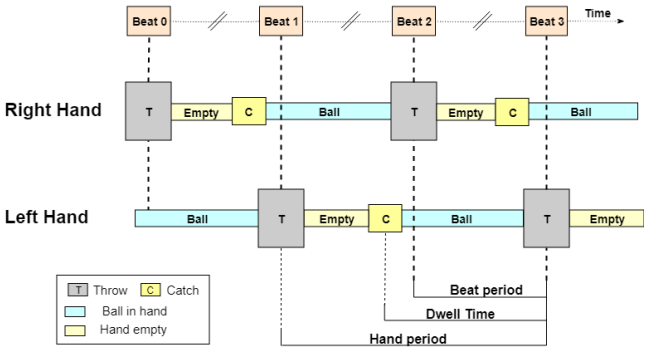


Figure 2: Asynchronous juggling system. One ball is thrown (T) from the right hand at every even beat, and one ball is thrown from the left hand at every odd beat. After a throw, our hand is empty and a new ball is caught (C).

To understand the juggling process, we will describe the actions of the left hand in Figure 2. At the beginning, the left hand has a ball and is preparing a throw. Then, the beat 1 comes and we throw this ball (T). After the throw, our left hand is empty until we catch a new ball (C). After the catch, we have a ball in hand and we prepare the next throw at beat 3. In this example, the odd beats correspond to a throw from the left hand, and the even beats, a throw from the right hand. We do not specify where the ball should be thrown, and this is explained by the basic patterns.

2.2 Juggling patterns

We name basic pattern or basic Siteswap, the simplest kind of juggling patterns and its number corresponds to the number of beats you have to wait before throwing this ball again. For example a ball thrown with a pattern '2' will have to be thrown again two beats later. This means that if a ball is thrown at beat 1, it should be thrown again at beat 3 and that the ball will have to be caught at the hand period just before this. As the hand who performs the throw alternates between right and left at every beat time, an odd number represents a throw that will go from one hand to another, and an

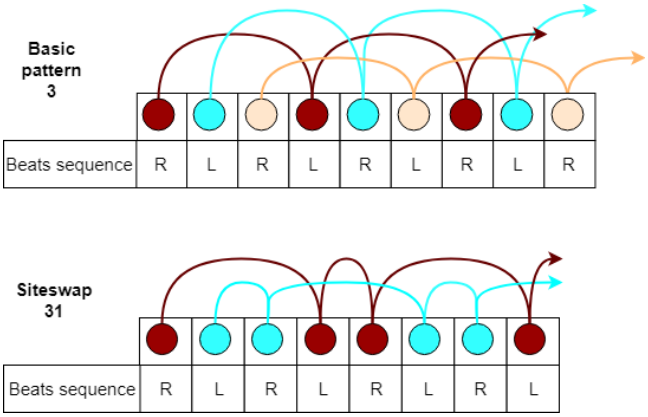


Figure 3: Example of basic pattern (3) and Siteswap (31).

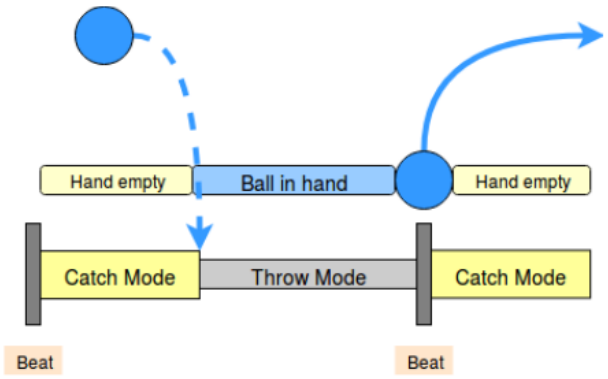


Figure 4: Sequence of catch and throw mode for one hand.

even number is a throw for the same hand. A vanilla Siteswap is a cyclic sequence of several basic patterns where we will throw a ball at each beat with the next pattern in this sequence. It exists as many vanilla Siteswap patterns as there are combinations possible where we have only one catch per hand period and one ball thrown at each beat. In this paper, we will name Siteswap only sequences of at least two different basic patterns. By juggling with only basic patterns, the number of ball we can have is equal to the pattern used. For example, juggling with a basic pattern '4' allows you to juggle with four balls and so on. For a Siteswap, the number of balls we juggle with is equal to the sum of all basic patterns of the sequence divided by its length. For example, a Siteswap '531' will be juggled with three balls.

We associate a throw to a basic pattern. The higher the basic pattern is, the more the ball will have to stay in the air and so the more powerfull the throw will have to be.

$$T_{ballInAir} = P_{hand}{}^{1}0{:}5 \quad C_{pattern} \quad D_{ratio}{}^{o} \tag{1}$$

where $T_{ballInAir}$ is the time the ball will have to spend in the air before being caught, $Pattern$ is the basic pattern of this throw, P_{hand} is the hand period, $C_{pattern}$ is the basic pattern of the throw superior or equal to two and D_{ratio} is the Dwell time divided by the hand period.

$$V_{yThrow} = 9{:}81{}^{1}T_{ballInAir} \bullet 2{:}0^{o} \tag{2}$$

$$V_{xThrow} = \frac{D_{Catch,Throw}}{T_{ballInAir}} \tag{3}$$

where V_{yThrow} and V_{xThrow} is the velocity wanted for the throw on the Y-axis and X-axis and $D_{Catch,Throw}$ is the distance between the throw and the catch at the same height. Performing a throw of a specific basic pattern consists in tossing the ball at the velocity V_{xThrow} and V_{yThrow} associated to it.

3 APPROACH

The overall juggling process can be explained as throwing the ball at the right velocity and time and catching the next ball incoming. Our approach is to learn separately these two distinct tasks using Deep Reinforcement Learning (Deep RL). We first train our agents only on the right arm of our body and we mirror it on the left arm. Then, in parallel on both sides of the body, we use our catching and throwing agent sequentially to recreate the all process of juggling.

3.1 Juggling sequence

During the simulation, we keep track of the beat time for the left and right hand. At every even beat we throw a ball from the right and at every odd beat, from the left. The switching between the catching and throwing agent for one arm is straightforward as shown in Figure 4. When we have a ball in hand, we are in throwing mode and when our hand is empty, we are in in catching mode. Therefore, we need to detect the frame when a ball is caught and when the ball is thrown. Also, Both of these tasks need to adapt to possible external forces that could be applied on balls while in the air or to a late catch which could lead to a very reduced Dwell time and so less time to prepare the throw.

3.2 Reinforcement learning

Reinforcement learning is defined as a Markov decision process $^{1}S; A; P; R; \gamma^{o}$ where S is a set of states, A a set of actions discrete or continuous, $P^{1}s; a; s^{0}{}^{o}$ is the transition probability $p^{1}s^{0}js; a^{o}$, R a reward function mapping S A ! R, and γ 2 »0; 1º a discount factor.

We train an agent to take some action in a fully observable environment, which returns him a reward to encourage him or not to do this action next time. The goal is to learn a policy which gives the action to perform in function of the actual state $a_t = \pi^{1}s_t{}^{o}$ in order to maximize the future cumulative rewards.

To the optimal policy, we associate a Q-function or action value function which satisfies the Bellman optimality equation :

$$Q^{1}s; a^{o} = R^{1}s; a^{o} + \gamma \max_{a^{0}2A} Q^{1}s^{0}; \pi^{1}s^{0}{}^{oo} \tag{4}$$

Deep Deterministic policy gradient (DDPG) [Lillicrap et al. 2015] is a model-free RL algorithm used for actions in continuous space. It is composed of two networks, the Actor and the Critic. The Actor network learns the optimal policy. The Critic network learns an approximation of the Q-value function and updates the Actor network toward the optimal actions to perform.

3.3 Catching

The ball is automatically caught by the hand if their position are close. So the goal of the catching task is to put the hand on the ball and if possible, near the height of catch wanted. The catching reward is composed of two terms :

$$R = w_1 R_{effort} + w_2 R_{catch} \tag{5}$$

where the weights w_1 and w_2 are constant values. The first term, R_{effort} encourages the agent to move smoothly by punishing jerks.

$$R_{effort} = \sum_{j \in J} \| \omega_t^j \quad \omega_{t \quad 1}^j \| \tag{6}$$

where J is a set of joints, ω_t^j and $\omega_{t \quad 1}^j$ are the angular joint velocity at time t and $t \quad 1$.

The second term $R_{catchin\partial}$ penalizes the agent for not catching a ball or catching it far from the height wanted.

$$R_{catchin\partial} = \begin{cases} 1; & \text{if fail to catch the ball} \\ \| \bar{p}_y \quad p_y \| & 2 \text{ (-1., 0.] otherwise} \end{cases} \tag{7}$$

where $\bar{p}_y$ and p_y are the height of catch wanted and the actual height of catch respectively.

The first term is applied at every step whereas the second one is applied only once per episode, on the terminal state. The episode ends when the ball is caught or lands out of the action range of the arm.

3.4 Throwing

To simplify the throwing task, we decide to release the ball automatically, respectively to the Juggling beat times. In this condition, the goal of throwing is at the release time to move our hand with the grasped object near the releasing position wanted at the right velocity. The throwing reward is composed of three terms :

$$R = w_3 R_{effort} + w_4 R_{positionThrow} + w_5 R_{velocity} \tag{8}$$

where the weights w_3, w_4 and w_5 are constant values. The first term, R_{effort} encourages the agent to move smoothly by punishing jerks.

$$R_{effort} = \sum_{j \in J} \| \omega_t^j \quad \omega_{t \quad 1}^j \| \tag{9}$$

The second term, $R_{positionThrow}$ penalizes the agent for throwing the ball far from the release point wanted :

$$R_{positionThrow} = \begin{cases} \| \bar{p} \quad p \|^2 & \text{when ball thrown} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $\bar{p}$ is the position where the hand should release the ball and p is the position of the actual release point. The last term $R_{velocity}$ penalizes the agent for not throwing the ball at the velocity wanted :

$$R_{velocity} = \begin{cases} \| v_{throw} \quad v_{throw} \|^2 & \text{when ball thrown} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where v_{throw} and v_{throw} are the velocity of the ball wanted and actual respectively when thrown.

The first term is applied at every step whereas the second and third terms are applied only once per episode, on the terminal state. The episode ends when the ball is thrown.

4 EXPERIMENTS

Table 1: Parameters

Simulation time step	10ms
Replay Buffer size	1e5
Hidden Layer size (1)	512
Hidden Layer size (2)	256
Learning rate (Actor)	1e-4
Learning rate (Critic)	1e-3
Discount factor (γ)	0.95
Pe_{max}	1.0
Pe_{min}	0.20
P_{hand}	0.5s
D_{ratio}	70%
k_p	1200.0
k_v	70.0
w_1	0.12
w_2	1.0
w_3	0.29
w_4	9.0
w_5	1.4
Gravity (N/Kg)	9.81

Our algorithm was implemented in Python, using DART for the simulation of articulated body dynamics and TensorFlow for the training of deep neural networks. The simulation was run on a PC with Intel Core i7-4930K (6 cores, 3.40GHz). Parameters can be seen in Table 1. During the training of our both agents catching and throwing, we linearly decay the exploration rate from Pe_{max} to Pe_{min} over 8000 episodes of training. The size of hidden layers is the same for both actor and critic and for both agents. For this experiment, we do not take in consideration collision which is part of a more complicated problem as we will explain in the later part of this paper.

4.1 States

The state of the agent is contains information about the side of the body we will use and the ball concerned. We get for the body state, the position p_{hand} and velocity v_{hand} of the hand, and a set of four joint angles θ_J in radians, with $J = \{\text{neck,shoulder,elbow,hand}\}$. We get the position p_{ball} and the velocity v_{ball} of the ball. For the throwing agent we augment the state with the time left before throwing the ball t_{throw} and the velocity of the throw wanted v_{throw}. As a result we have the catching state and the throwing state :

$$s_{catchin\partial} = \{p_{hand}, v_{hand}, p_{ball}, v_{ball}, \theta_J\}$$
$$s_{throwin\partial} = \{p_{hand}, v_{hand}, p_{ball}, v_{ball}, \theta_J, t_{throw}, v_{throw}\}.$$

4.2 Actions

For both policies, catching and throwing, an action represents a set of target joint angles $\bar{\theta}_J$. We have for both agents an action $a = \{\bar{\theta}_{neck}, \bar{\theta}_{shoulder}, \bar{\theta}_{elbow}, \bar{\theta}_{hand}\}$. These target joint angles are used by the PD-controller to compute the torques to apply to

the joints.

$$\tau_i = k_p^i(\bar{\theta}_i - \theta_i) + k_v^i(\bar{\omega}_i - \omega_i) \quad (12)$$

where joint $i \in J$, τ_i is the torque applied to it, $\bar{\theta}_i$ and θ_i are its joint angle target and actual respectively, $\bar{\omega}_i$ and ω_i are the joint angle velocity target and actual respectively and k_p^i and k_v^i are manually-specified gains. For this experiment, we set the same gain k_p and k_v for all joints and $\bar{\omega}_i$ is set to zero for all joints in J.

5 RESULTS

We first test each agent for what they have been trained and we finally use them sequentially to create our simulated juggler. The results for both tasks can be seen in the supplemental video. The training time needed to make our learning curves converging with our agents are around 1 hour 20 minutes for throwing and 2 hours for catching. The throwing task learning converges faster and is a far more stable than the catching task, but both succeeds in their own subtask. As a result, the combination of both agents on our character is able to perform Siteswap composed of basic pattern up to 7.

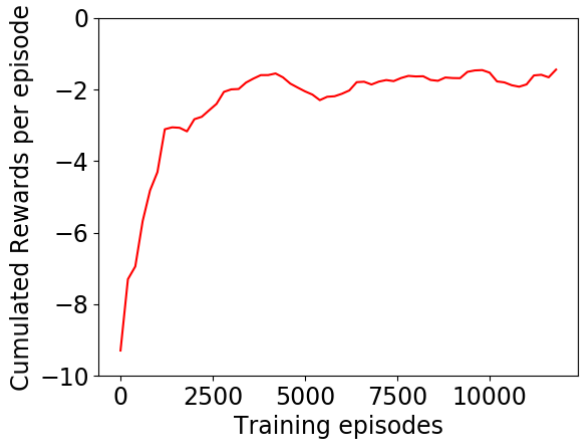


Figure 5: Learning curve for throwing task.

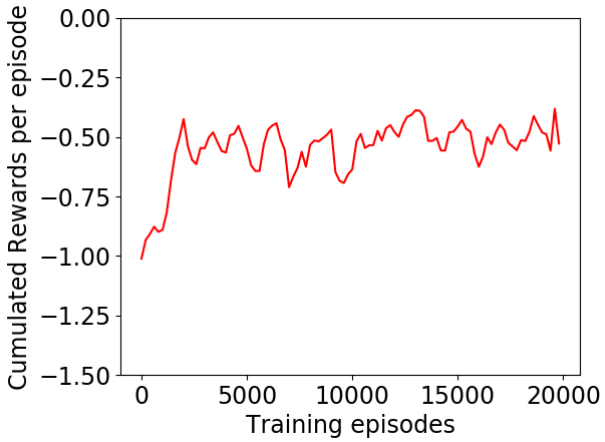


Figure 6: Learning curve for catching task.

5.1 Throwing results

We trained our throwing agent to move our hand with the grasped ball toward the the position wanted at the right velocity before the release time. This velocity is associated to the pattern we want to perform and we train it with patterns up to 7. Figure 5 shows the learning for the throwing task. At the start of the training, our hand fails to throw the ball at the right velocity resulting in a high negative reward. Finally our hand finds out a strategy which can balance value of negative rewards. At the final state of training, the reward average is around 2 with 40% of it due to the error in velocity, 25% due to error of position of throw wanted and 35% due to the effort of joints. We manually selected the reward weights to have the best and smoother agent as possible. Considering the velocity and position reward weights fixed, we found the effort reward weight being very sensitive to change as it can bring to a very jerky result or a fail in being able to throw. A vertical velocity error directly impacts the time the ball will stay in the air. If this difference of time is too high, the ball may land in the wrong hand period and as we can have only one ball in the hand per period, this means that the ball will not be caught and will fall on the ground. A horizontal error changes the position of ball landing. As long as this position is in the range of action of the hand receiving, the catch is possible, but if the ball lands outside of it then the pattern will fail. Our agent can successfully throw a ball with a pattern up to 7 with a non critical error in velocity.

5.2 Catching results

We trained our catching agent to place his hand on a ball to catch it near the height wanted. The ball can come from any position at any velocity. As a result, the hand waits the ball to be in its range of action and rapidly moves to its position to catch it. We test the robustness of our agent by adding some wind as a constant external force on balls, modifying its velocity. The reward when the catch fails is inferior or equal to -1.0 and superior if the catch is a success. The closer the catch is to the position wanted, the higher is the reward. The hand successfully catches the ball as long as the ball is in its arm range and as close as possible from the position wanted.

We manually selected the reward weights to ensure the catch while being smooth.

Catching is a very time sensitive task that makes the learning very unstable as we can see in Figure 6. We think that the number of frames where the ball is in catching range can be very short and is reduced if the velocity of the ball incoming is high or if we raise the simulation time step. The hand has to be precisely in the area of grasping of the ball on these key frames. Considering the catching reward weight fixed, if we set a low reward effort weight, our catching agent tries sweep the area very fast near the position of landing. This strategy gives a high probability to catch the ball and also makes the learning more stable but full of jerks and unnatural. In this work we will keep the value of effort reward weight where our agent performs the smoother catches.

5.3 Juggling results

After these agents learned successfully to perform their respective subtasks, we apply them to both arms and we switch between both agents as explained in Figure 4. The catching agent succeeds in catching the ball thrown from the same or other hand. The throwing agent succeeds in tossing the ball with the right velocity and to make it land on the aimed hand at the hand period wanted for all patterns up to 7. After the initialization phase where all balls will get a near constant trajectory, the result of the combination of both agents appears to be quite smooth when juggling continuously.

Figure 7 shows some results of our juggling simulation. The basic pattern 3 (a) consists in throwing a ball to the other hand at every beat time. Basic pattern 4 (b) is an even number, so the thrown will be for the same hand. Finally the Siteswap 53 (c) consists in throwing a ball from the left hand with a basic pattern 3 and from the right hand with a basic pattern 5. When trained with different reward weights for throwing, our character can show some limitations with patterns superior or equal to 7, as it does not have enough power to throw it high enough. As a result, the hand period when the ball is at catching range, is already occupied by another ball and the juggling fails.

We test the robustness of our agents by adding some wind on the balls while in the air, adding a constant horizontal acceleration and making them landing further or closer to our character. Our agents succeeds to catch all balls as long as they land in his range of action and to perform the next throw. These juggling simulation for different patterns can be seen in the supplemental video.

6 CONCLUSION

We presented a method to learn a catching and throwing task in a physics-based simulation using Deep Deterministic Policy Gradients and we used them sequentially by switching from catching to throwing mode, respecting the mathematics of juggling with the vanilla Siteswap notation. In our simulation, Reinforcement Learning allows us to have a real time Juggling simulation able to adapt to some external forces on balls and to perform some patterns continuously with success.

The unstable learning of the catching agent has an impact on his ability to catch some balls in difficult situations. To make this learning more stable, a solution could be to use directly the calculated position of ball landing to make the agent anticipate it. Concerning the throwing agent, our character fails to throw patterns higher than 7 which requires more power. An easy solution could be to raise the gain of the PD-controller k_p and k_v to make the joints moving faster but this would lead to more unnaturalness. There is a lot of improvements to do in order to fix the still jerky movements of our character, as setting customized gain for each joint. Another room of exploration will be to go for a 3D simulation. Adding new degrees of freedom may give to our character more options to throw high patterns and also to go for a realistic motion. Furthermore, considering collisions and adding some balls while juggling require to plan the timing, power and direction of our future throws. The planning of throws and movements could be performed by a high and low level controller respectively as demonstrated by Peng et al. [2017] applied to locomotion planning. Finally, our agents are trained to catch at a fixed height and to throw position at a fixed position. It could be interesting in a future work to train him to throw and catch at any position allowing more complex juggling patterns and also to modify our simulation to perform other types of juggling. Football juggling for example could be done by setting a null value to the Dwell time and by redefining the conditions of catches and throws in the simulation.

ACKNOWLEDGMENTS

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the SW STARLab support program (IITP-2017-0536-20170040) supervised by the IITP(Institute for Information & communications Technology Promotion.

REFERENCES

Peter J. Beek and Arthur Lewbel. 1995. The Science of Juggling - Studying the ability to toss and catch balls and rings provides insight into human coordination, robotics and mathematics. *Scientific American* 273, 5 (1995).

Pavel Burget and Pavel Mezera. 2010. A Visual-Feedback Juggler With Servo Drives. *The 11th IEEE International Workshop on Advanced Motion Control* (2010).

Kai Ding, Libin Liu, Michiel van de Panne, and KangKang Yin. 2015. Learning Reduced-Order Feedback Policies for Motion Skills. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. 2017. Emergence of Locomotion Behaviours in Rich Environments. *CoRR* abs/1707.02286 (2017).

Sumit Jain and C. Karen Liu. 2009. Interactive Synthesis of Human-Object Interaction. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*. ACM, New York, NY, USA, 47–53. DOI : http://dx.doi.org/10.1145/1599470.1599476

Jack Kalvan. 2018. The Human Limits - How Many Objects Can Be Juggled. (2018).

Jens Kober, Matthew Glisson, and Michael Mistry. 2012. Playing Catch and Juggling with a Humanoid Robot. *12th IEEE-RAS International Conference on Humanoid Robots* (2012).

Seunghwan Lee, Ri Yu, Jungnam Park, Mridul Aanjaneya, Eftychios Sifakis, and Jehee Lee. 2018. Dexterous Manipulation and Control with Volumetric Muscles. *ACM Transactions on Graphics (SIGGRAPH 2018)* 37, 57 (2018).

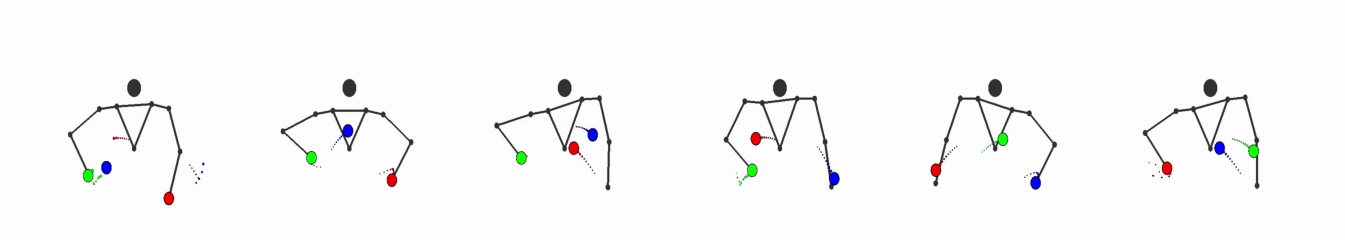
Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015).

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013).

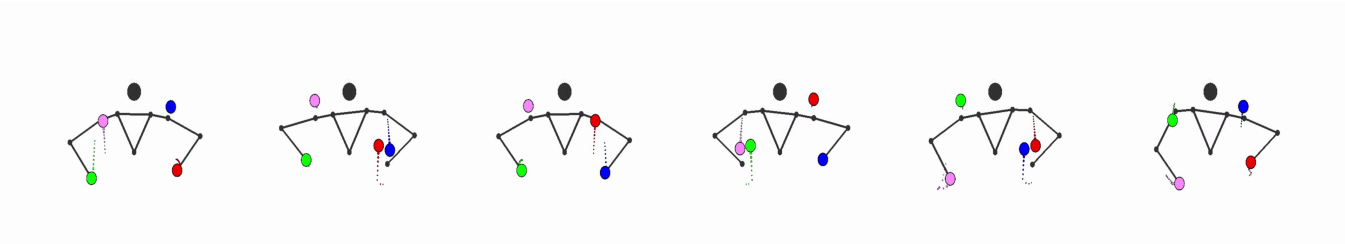
Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018 - to appear)* 37, 4 (2018).

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)* 36, 4 (2017).

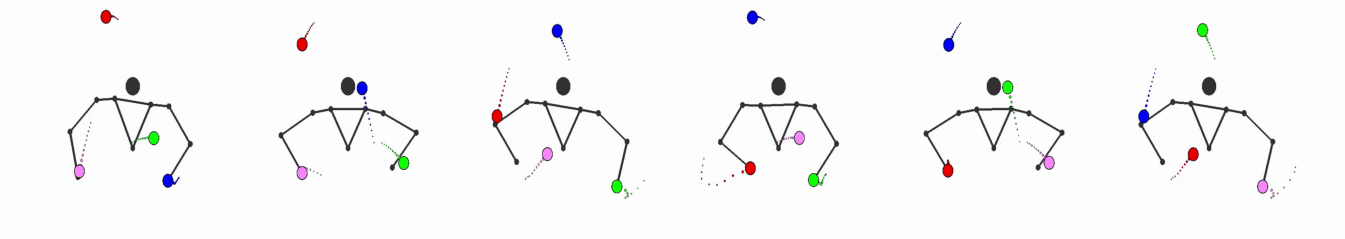
Claude Elwood Shannon. 1981. Scientific Aspects of Juggling. In *Claude Elwood Shannon: Collected papers*., Aaron D. Wyner and N.J.A. Sloane (Eds.). IEEE Press, New York, 850âĂŞ864.



(a) Basic Pattern "3"



(b) Basic Pattern "4"



(c) Siteswap "53"

Figure 7: Juggling patterns.