Deep Learning Based Motor Control Unit

Submitted by:
Abhinav Rai (13BCS-0005)
Ashhar Hasan (13BCS-0015)

Submitted to:
Mr. Danish Raza Rizvi

Department of Computer Engineering
Faculty of Engineering and Technology
Jamia Millia Islamia, New Delhi — 110025

Abstract

We are developing a deep reinforcement learning approach to control musculoskeletal biomechanical models which do not need motion-capture data or explicit programming to replicate a motion. We will develop a musculoskeletal biomechanical human model using *Stanford OpenSim* [Delp et al., 2007] trained to replicate motion like standing and walking without external controls using reinforcement learning. All actuation forces are the result of 3D simulated muscles and as a result our controllers will generate motion patterns that incorporate biomechanical constraints like metabolic energy expenditure. The generated controllers will be able to find different gaits based on target speed, target direction and other external control factors.

Introduction

The development of physics-based locomotion controllers, independent from captured motion data, has been a long-standing objective in computer graphics and robotics research and recently in biomechanical communities. In robotics with the help of reinforcement learning instead of expensive and complex explicit programming robots can teach themselves how to move in virtual spaces and real environments like in [Heess et al., 2016].

In biomechanics research a lot of models have been developed to fit the clinical data to understand underlying causes of injuries using inverse kinematics and inverse dynamics. For many of these models there are controllers designed for forward simulations of movement, however they are often finely tuned for the model and data. Applying reinforcement learning to this task may allow building more robust controllers for large number of tasks.

The use of reinforcement learning together with physics-based simulations offers the enticing prospect of developing classes of motion skills from first principles. This requires viewing the problem through the lens of a sequential decision problem involving states, actions, rewards, and a control policy. Given the current situation of the character, as captured by the state, the control policy decides on the best action to take, and this then results in a subsequent state, as well as a reward that reflects the desirability of the observed state transition. The goal of the control policy is to maximize the sum of expected future rewards, i.e., any immediate rewards as well as all future rewards.

In practice, a number of challenges need to be overcome when applying the reinforcement learning framework to problems with continuous and high-dimensional

states and actions, as required by movement skills. A control policy needs to select the best actions for the distribution of states that will be encountered, but this distribution is often not known in advance. Similarly, the distribution of actions that will prove to be useful for these states is also rarely known in advance. Furthermore, the state and action distributions are not static in nature; as changes are made to the control policy, new states may be visited, and, conversely, the best possible policy may change as new actions are introduced. It is also not obvious how to best represent the state of a character and its environment. Using large descriptors allows for very general and complete descriptions, but such high-dimensional descriptors define large state spaces that pose a challenge for many reinforcement learning methods. Using sparse descriptors makes the learning more manageable, but requires domain knowledge to design an informative and compact feature set that may nevertheless be missing important information.

Related Work

Methods for physics-based character animation that use forward dynamic simulations have been a research focus for over two decades, most often with human locomotion as the motion of interest. A survey of the work in this area can be found in [Geijtenbeek and Pronost, 2012]. Impressive results were achieved in classical works by [Coros et al., 2011], [Wang et al., 2012] and [Geijtenbeek et al., 2013]. But most of these works focus on controlling physics-based locomotion over flat terrain. Recently there have been some very interesting attempts to use deep reinforcement learning for locomotion on complex terrain, for example [Peng et al., 2016], but only in 2D at the moment. Another interesting approach using biologically inspired idea that motor systems are hierarchical was suggested in [Geijtenbeek and Pronost, 2012]. Animation researchers have been interested in the control of locomotion for 3D humanoid characters for almost 20 years. One important recent contribution is *SIMBICON* [Yin et al., 2007], a remarkably robust 3D humanoid locomotion controller based on the balance control of [Raibert and Hodgins, 1991]. A number of projects have since focused on expanding the controller repertoire for simulated bipeds and on locomotion in complex environments. At the same time, efforts have been made to make the synthesized motions more human-like, or "natural". The original *SIMBICON*-style controllers tend to produce gaits lacking hip extension with a constant foot orientation. Knee angles lack flexion during swing, but lack extension at heel strike. More recent controllers improve motions by designing better target trajectories in joint or feature space. While more human-like ankle motions have been produced, differences in the hip and knee angles persist. Perhaps more importantly, controllers relying on hand-tuned trajectories cannot be easily used to investigate how the

control strategies change with respect to new constraints. For example, how would the character's motion style change given a physical disability? Can we synthesize appropriate gaits for older or younger characters? Impressive results have also been achieved by controllers based on tracking motion capture data as in [da Silva et al., 2008] and [Ye and Liu, 2010]. However, as with methods that tune joint trajectories or controller parameters by hand, motion capture driven controllers have a limited ability to predict changes in gait.

Proposed Methods and Algorithms

Problem Formulation

In robotics with the help of reinforcement learning instead of expensive and complex explicit programming robots can teach themselves how to move in virtual spaces and real environments. For the project musculoskeletal models of human body will be used. Thereafter the model will be trained for performing activities such as standing still, walking-gait, jumping, etc. The models will be built using *Stanford OpenSim* — biomechanical physics environment for musculoskeletal simulations. For deep reinforcement learning, deep learning frameworks such as *TensorFlow* [Abadi et al., 2016] will be used, which can be combined with *Keras-rl* [Plappert, 2016] framework which implements state-of-the-art deep reinforcement learning algorithms in Python.

Problem Representation

The method consists of a standard reinforcement learning setup consisting of an agent interacting with an environment E in discrete time-steps. At each time-step t the agent receives an observation x_t, takes an action a_t and receives a scalar reward r_t. In all the environments considered here the actions are real-valued at $a_t \in R^N$. In general, the environment may be partially observed so that the entire history of the observation, action pairs $s_t = (x_1, a_1, \ldots, a_{t-1}, x_t)$ may be required to describe the state. Here, we assumed the environment is fully-observed so $s_t = x_t$.

An agent's behaviour is defined by a policy, π, which maps states to a probability distribution over the actions $\pi : S \rightarrow P(A)$. The environment, E may also be stochastic. We model it as a Markov decision process with a state space S, action space $A = R^N$, an initial state distribution $p(s_1)$, transition dynamics $p(s_{t+1}|s_t, a_t)$, and reward function $r(s_t, a_t)$.

The return from a state is defined as the sum of the discounted future reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r\left(s_i, a_i\right)$ with a discounting factor $\gamma \in [0, 1]$. Note that the return depends on the actions chosen, and therefore on the policy, and may be stochastic. The goal in reinforcement learning is to learn a policy which maximizes the expected return from the start distribution $J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}\left[R_1\right]$. We denote the discounted state visitation distribution for a policy π as ρ^{π}.

The action value function is used in many reinforcement learning algorithms. It describes the expected return after taking an action a_t in state s_t and thereafter following policy $\pi : Q^{\pi}\left(s_t, a_t\right) = \mathbb{E}_{r_{i \geq t}, s_{i>t} \sim E, a_{i>t} \sim \pi}\left[R_t | s_t, a_t\right]$. Many approaches in reinforcement learning make use of the recursive relationship known as the Bellman equation:

$$Q^{\pi}\left(s_t, a_t\right) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r\left(s_t, a_t\right) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}\left[Q^{\pi}\left(s_{t+1}, a_{t+1}\right)\right]\right] \tag{1}$$

If the target policy is deterministic we can describe it as a function $\mu : S \leftarrow A$ and avoid the inner expectation:

$$Q^{\mu}\left(s_t, a_t\right) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r\left(s_t, a_t\right) + \gamma Q^{\mu}\left(s_{t+1}, \mu\left(s_{t+1}\right)\right)\right] \tag{2}$$

The expectation depends only on the environment. This means that it is possible to learn Q^{μ} off-policy, using transitions which are generated from a different stochastic behaviour policy β.

Programming Environment and Tools Used

- Python
- TensorFlow
- Keras-RL
- Stanford OpenSim
- Stanford Simbody
- NumPy
- SciPy
- Scikit-Learn

References

[Abadi et al., 2016] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

[Coros et al., 2011] Coros, S., Karpathy, A., Jones, B., Reveret, L., and Van De Panne, M. (2011). Locomotion skills for simulated quadrupeds. In *ACM Transactions on Graphics (TOG)*, volume 30, page 59. ACM.

[da Silva et al., 2008] da Silva, M., Abe, Y., and Popović, J. (2008). Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics (TOG)*, 27(3):82.

[Delp et al., 2007] Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., Guendelman, E., and Thelen, D. G. (2007). Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering*, 54(11):1940–1950.

[Geijtenbeek and Pronost, 2012] Geijtenbeek, T. and Pronost, N. (2012). Interactive character animation using simulated physics: A state-of-the-art review. In *Computer Graphics Forum*, volume 31, pages 2492–2515. Wiley Online Library.

[Geijtenbeek et al., 2013] Geijtenbeek, T., van de Panne, M., and van der Stappen, A. F. (2013). Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, 32(6):206.

[Heess et al., 2016] Heess, N., Wayne, G., Tassa, Y., Lillicrap, T., Riedmiller, M., and Silver, D. (2016). Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*.

[Peng et al., 2016] Peng, X. B., Berseth, G., and van de Panne, M. (2016). Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):81.

[Plappert, 2016] Plappert, M. (2016). keras-rl. `https://github.com/matthiasplappert/keras-rl`.

[Raibert and Hodgins, 1991] Raibert, M. H. and Hodgins, J. K. (1991). Animation of dynamic legged locomotion. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 349–358. ACM.

[Wang et al., 2012] Wang, J. M., Hamner, S. R., Delp, S. L., and Koltun, V. (2012). Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM transactions on graphics*, 31(4).

[Ye and Liu, 2010] Ye, Y. and Liu, C. K. (2010). Optimal feedback control for character animation using an abstract model. In *ACM Transactions on Graphics (TOG)*, volume 29, page 74. ACM.

[Yin et al., 2007] Yin, K., Loken, K., and van de Panne, M. (2007). Simbicon: Simple biped locomotion control. In *ACM Transactions on Graphics (TOG)*, volume 26, page 105. ACM.