Deep reinforcement learning for 2D soft body locomotion

Junior Rojas
University of Utah
jrojasdavalos@gmail.com

Stelian Coros
ETH Zurich
scoros@gmail.com

Ladislav Kavan
University of Utah
ladislav@cs.utah.edu

Abstract

We present a character animation system that integrates deep reinforcement learning into soft body simulation to synthesize locomotion controllers for autonomous characters modeled as deformable 2D triangle meshes with no rigid components. A controller in our system is a mapping from state observations (vertex positions and velocities of the character's mesh) to actions that contract or relax the character's muscles. Our results show that our locomotion controllers can be run from different initial simulation states and recover from perturbations such as external impulses generated by unexpectedly poking the character.

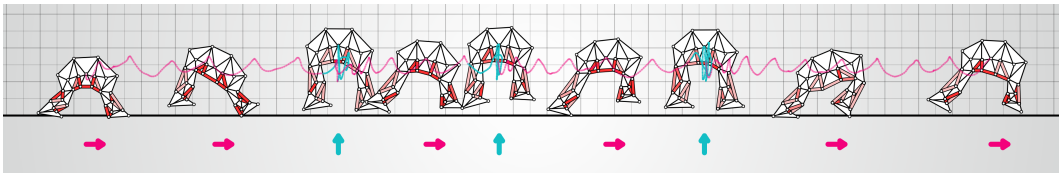

Figure 1: Our trained policies are robust to external perturbations and can be run from different initial states, which allows us to switch between independently trained policies to transition from horizontal locomotion to jumping and vice versa. Video available at `https://youtu.be/J36MOHfxWYg`.

1 Introduction

While common approaches to model physically realistic autonomous agents are based on structures composed of rigid links and joints, real-world animals are composed of many types of materials with different degrees of elasticity. The common articulated rigid-body model implies that motion is achieved by directly operating on joint angles, which serves as an approximation to the internal structure present in vertebrate animals. However, animal locomotion is achieved by controlling muscles, which has an indirect effect on joint angles as a result of the physical interactions between different types of tissue.

We present an approach that integrates deep reinforcement learning into soft body simulation to optimize controllers for autonomous characters modeled as deformable 2D triangle meshes with no rigid components that achieve locomotion using a muscle-like actuation mechanism. Our system is aware of the long-term effects of the actions taken every time step of the simulation and does not assume prior knowledge of environment dynamics for optimization purposes. This is in contrast to previous soft-bodied character animation methods which have an effective planning horizon of one time step and require tight coupling of the dynamics model with the control method using constrained optimization [1; 2]. Complementary methods based on spacetime optimization [3; 4] optimize a fixed sequence of actions to achieve a specified motion. In contrast, our system optimizes control policies,

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

i.e., functions from simulation state-dependent observations to actions, which results in effective locomotion controllers that are robust to external perturbations.

2 Simulation

Our animation system uses the finite element method to simulate characters as 2D triangle meshes with a Neo-hookean-based elasticity model. We also embed controllable springs in the triangle meshes to generate contractile forces that can be interpreted as muscle contraction or relaxation, and we implement basic models of friction and collision with the ground to enable terrestrial locomotion. For numerical integration, we use the variational formulation of backward Euler due to its excellent stability even with large time steps [5; 6].

3 Policy

Muscles are controlled by a stochastic policy. A neural network maps partial observations of the environment, represented as vertex positions and velocities of the character's mesh projected onto a local frame, to a multivariate Gaussian distribution. An action sampled from the distribution specifies how to change the rest lengths of the controllable springs and can be interpreted as muscle contraction (negative values) or relaxation (positive values). To avoid unrealistic motions, we do not allow muscles to expand beyond their original rest lengths or contract to zero rest length (by imposing an arbitrary minimum rest length).

4 Training

All our experiments use episodic tasks with a fixed time horizon H where the reward is only delivered at the end of the episode. One iteration of our training loop runs H time steps of m episodes in parallel using the policy π_θ, which provides us with training data to improve the policy parameters θ. We use R_i to refer to the reward delivered at the end of episode i, and o_{it} and a_{it} to refer to the observation and action at time step t of episode i. We optimize an objective function based on the surrogate objective used in proximal policy optimization (PPO) [7]:

$$L(\theta) = \frac{1}{m} \sum_{i=0}^{m-1} \sum_{t=0}^{H-1} \min(r_{it}(\theta)\Psi_i, \text{clip}(r_{it}(\theta), 1-\epsilon, 1+\epsilon)\Psi_i) \tag{1}$$

$$r_{it}(\theta) = \frac{\pi_\theta(a_{it}|o_{it})}{\pi_{\theta_{\text{old}}}(a_{it}|o_{it})} \tag{2}$$

Common implementations of PPO use a value function as a baseline for Ψ, which requires training another neural network to learn a value function. However, we adopted a simpler approach: we use the average reward of the episodes sampled in the current training iteration as a baseline, which resulted in significant improvements over not using a baseline at all, while at the same time avoided the complexities of training an additional neural network:

$$\Psi_i = R_i - \frac{1}{m} \sum_{k=0}^{m-1} R_k \tag{3}$$

For every training iteration, we run 10 iterations of the Adam optimizer [8] with learning rate 10^{-3} on the surrogate objective L (Equation 1) with $\epsilon = 0.2$. To train policies for horizontal locomotion we used a reward equal to the total horizontal displacement of the agent's local frame, and to train jumping policies we used a reward equal to the maximum height reached during the episode. Our results show that the trained policies are robust to external perturbations and can be run from different initial states, which allows us to switch between independently trained policies as shown in Figure 1. See the video demo for additional results (`https://youtu.be/J36MOHfxWYg`).

References

[1] S. Coros, S. Martin, B. Thomaszewski, C. Schumacher, R. Sumner, and M. Gross, "Deformable objects alive!" *ACM Trans. Graph.*, vol. 31, no. 4, pp. 69:1–69:9, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2185520.2185565

[2] J. Tan, G. Turk, and C. K. Liu, "Soft body locomotion," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 26:1–26:11, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2185520.2185522

[3] C. Schulz, C. von Tycowicz, H.-P. Seidel, and K. Hildebrandt, "Animating deformable objects using sparse spacetime constraints," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 109:1–109:10, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2601097.2601156

[4] Z. Pan and D. Manocha, "Active animations of reduced deformable models with environment interactions," *ACM Trans. Graph.*, vol. 37, no. 3, pp. 36:1–36:17, Aug. 2018. [Online]. Available: http://doi.acm.org/10.1145/3197565

[5] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *SIGGRAPH 98 Conference Proceedings*, 1998, pp. 43–54.

[6] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran, "Optimization integrator for large time steps," vol. 21, no. 10, pp. 1103–1115, 2015.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-1617172, IIS-1622360 and IIS-1764071. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also gratefully acknowledge the support of Activision, Adobe, and hardware donation from NVIDIA Corporation.