A Recurrent Differentiable Physics Engine for Tensegrity Robots

Kun Wang, Mridul Aanjaneya and Kostas Bekris

***Abstract*— Tensegrity robots, composed of rigid rods and flexible cables, are difficult to accurately model and control given the presence of complex dynamics and high number of DoFs. Differentiable physics engines have been recently proposed as a data-driven approach for model identification of such complex robotic systems. These engines are often executed at a high-frequency to achieve accurate simulation. Ground truth trajectories for training differentiable engines, however, are not typically available at such high frequencies due to limitations of real-world sensors. The present work focuses on this frequency mismatch, which impacts the modeling accuracy. We proposed a recurrent structure for a differentiable physics engine of tensegrity robots, which can be trained effectively even with low-frequency trajectories. To train this new recurrent engine in a robust way, this work introduces relative to prior work: (i) a new implicit integration scheme, (ii) a progressive training pipeline, and (iii) a differentiable collision checker. A model of NASA's icosahedron SUPERballBot on MuJoCo is used as the ground truth system to collect training data. Simulated experiments show that once the recurrent differentiable engine has been trained given the low-frequency trajectories from MuJoCo, it is able to match the behavior of MuJoCo's system. The criterion for success is whether a locomotion strategy learned using the differentiable engine can be transferred back to the ground-truth system and result in a similar motion. Notably, the amount of ground truth data needed to train the differentiable engine, such that the policy is transferable to the ground truth system, is 1% of the data needed to train the policy directly on the ground-truth system.**

I. Introduction

Prior work on *tensegrity locomotion* [1], [2], [3], [4], [5] has achieved complex behaviors, on uneven terrain, using the NTRT simulator [6], which was manually tuned to match a real platform [7], [8]. Recently, the authors have developed the first data driven differentiable engine for model identification of tensegrity robots [9], [10]. Nevertheless, critical gaps remain for the effective deployment of such tools on real robots. In particular, the observation sampling frequency on real robot is much lower than the simulation step frequency of a physics engine. This frequency discrepancy raises the challenge that the dynamics model needs to *predict* the missing data points of a discretely sampled sparse trajectory. In addition, the missing data points may include events that are critical to the motion, such as collisions (see Fig. 1(top)). A previous differential engine for tensegrities [10] used a feed-forward architecture and can only be trained from trajectories sampled at high frequencies, similar to that of the engine's frequency, i.e., in the order of 1000Hz, which is beyond what most sensors can achieve.

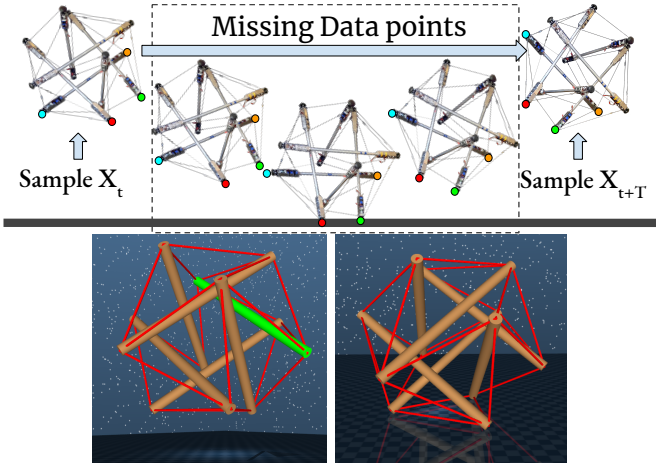

Fig. 1: (top) A sparsely sampled trajectory, given observation frequency T, may skip critical states with contacts. (bottom-left) A non-contact setup used first in a progressive training process: the green rod of the tensegrity is kept fixed, while random forces are applied to the other rods. (bottom-right) A MuJoCo terrain environment where the robot is rolling. Training data comes from MuJoCo, which is also used for visualization.

Motivated by this frequency gap, this work proposes (i) a recurrent architecture for a differentiable physics engine targeted for tensegrity robots, which remains explainable and can be trained effectively even with low-frequency ground-truth data; (ii) an implicit integration scheme; (iii) a progressive training algorithm, which avoids gradient explosion and leads to a fast and stable training process; (iv) a fast differentiable collision checker to further accelerate the training process. These components enable stable and robust recurrent training with low-frequency training data on a high-frequency physics engine. The long version is published recently [11].

II. Proposed Engine

A. Recurrent Differentiable Physics Engine

At a high level, the proposed training pipeline calls the differentiable physics engine in a recurrent fashion through a sequence of temporal connections. With help from Back-Propagation Through Time (BPTT), the engine parameters can be identified given low-frequency sampled trajectories. Figure 2 presents the recurrent nature of the proposed architecture.

To adapt this engine for recurrent training and achieve robustness, this work introduces an implicit integration scheme and a differentiable collision checker.

B. Implicit Integration

While implicit integration schemes are standard for simulating stiff multi-body systems, their dynamics are typically

The authors are with the Department of Computer Science, Rutgers University, NJ 08901, USA. Email: kun.wang2012, mridul.aanjaneya, kostas.bekris@rutgers.edu.
This work has been partially supported by NSF award IIS 1956027, IIS-2132972, CCF-2110861 and the Rutgers University startup grant.

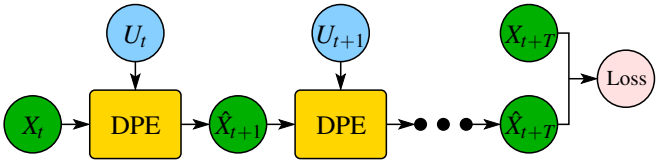

Fig. 2: The recurrent training pipeline is trained via supervision given a state-action tuple $X_t, \bar{U}_t, X_{t+T}$. The engine will generate all missing states between X_t and X_{t+T}. The mean square error (MSE) loss between the predicted state $\hat{X}_{t+T}$ and the ground truth X_{t+T} generates the gradients needed to update the engine's internal parameters. DPE: Differentiable Physics Engine.

handled by explicit methods, such as semi-explicit integration [12]. Semi-explicit integration is not stable for stiff systems, such as tensegrities. Deriving an implicit system, however, for the whole tensegrity is very complex. For instance, more complex than what has been achieved using a particle-based mesh system [13]. To address this complexity, this work follows a modular approach by focusing on the basic elements of tensegrity robots, rods and springs, and derives an $Ax = b$ form linear system for each time step, where A is a 21x21 matrix and x, b are 21x1 vectors (see appendix[1] for details).

C. Progressive Training via Implicit/Semi-implicit Integr.

The objective of progressive training is to mitigate the frequency discrepancy between the training data and the physics engine. We train the engine using implicit integration first in a time-stepping way to find a coarse-grain model first, and then fine tune with semi-implicit integration for a more accurate model. The implicit integration has a large region of absolute stability in terms of parameters and time step but it is computationally expensive and reduces system energy steadily over time. Fine tuning with semi-implicit integration is fast and more accurate, while it maintains system energy. The proposed progressive training approach takes advantage of both schemes' benefits.

D. Differentiable Collision Checker

The motion of the robot also depends upon the reaction forces and the friction between the robot and the ground. Thus, rich contacts should be modeled properly to simulate robot motions. In principle, the physics engine could use an off-the-shelf collision checker. The requirement for training the engine in a recurrent fashion and backpropagating the loss through the engine means that the collision checker should also be differentiable.

III. Experiments

The SUPERballBot tensegrity robot platform [14] is simulated in the MuJoCo engine as the ground truth system. The task is to estimate the robot's parameters, i.e., spring stiffness, damping, rod mass, and contact parameters, i.e., reaction and friction coefficients. The evaluation compares the differences of robot's center of the mass (CoM) between our engine and MuJoCo for the same controls.

The ground truth system on MuJoCo has been setup to mimic empirical motions of the real system at NASA Ames [14] and reflect the NTRT tensegrity robot simulator [6]. A non-contact and a contact environment were setup

[1] https://sites.google.com/view/recurrentengine

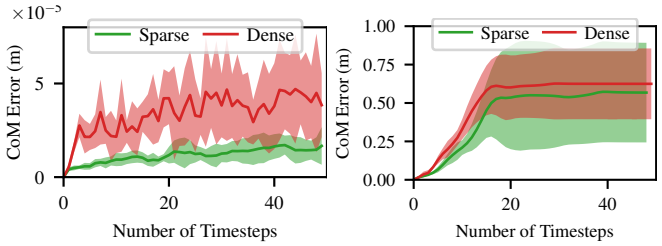

Fig. 3: (left) The proposed engine achieves smaller CoM position error in non-contact environments when trained on sparse trajectories, which contain only 1% of data points of dense trajectories, relative to previous work [10] trained on dense trajectories. (right) Even when using only 1% of data points, the proposed engine achieves comparable error regarding the CoM position in the contact environment.

as shown in Fig. 1. For each environment, 10 trajectories were samppled from MuJoCo for training, 2 for validation and 10 for testing. All trajectories are 5 seconds long. The sampling frequency is 10Hz, however, the engine's frequency is 1000Hz.

A. System Identification with Sparse Simulation Trajectories

We compared the proposed training using sparsely sampled trajectories against the alternative [10] which is trained with dense trajectories. Dense trajectories' sampling frequency is 1000Hz and the sparse ones' are 10Hz. Although the sparse dataset only contains 1% of the dense dataset, the proposed solution achieves better CoM error both for non-contact trajectories and contact trajectories as shown in Fig. 3. The recurrent training process can average out the noise over smaller simulation steps, resulting to a more robust and accurate model. The CoM error arises from the simplification of the contact model. This simplification reduces the need for training data, but also impacts identification accuracy, which is a trade-off between data requirements and model complexity.

B. System Identification with Sparse Real World Trajectories

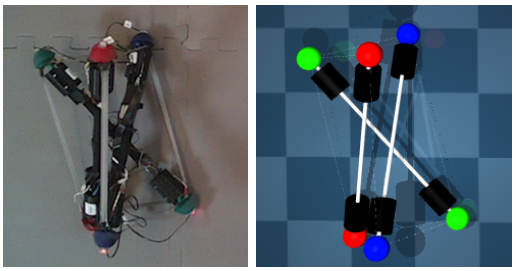

Fig. 4: A 3-bar tensegrity robot is used for sampling real world trajectories (left). We estimate the robot pose and then identify its parameters (right).

We are working on an identification task for a real, 3-bar tensegrity platform [15][2]. Due to the sensors' limitations, the sampling rate varies from 100Hz to less than 1Hz. We only have partially observed robot states since the end caps are usually occluded. The observations like linear and angular velocities are no longer available too. The observed values, e.g. cable lengths and end cap point cloud, are also noisy. All of these points emphasize the necessity for a recurrent differentiable physics engine and highlight the challenges of sim2real transfer tasks in this domain.

[2] Robot setup built by collaborators: Will Johnson, Xiaonan Huang, Joran Booth, Rebecca Kramer-Bottiglio at the Mech. Eng. dept. at Yale University.

References

[1] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 634–641.

[2] J. Luo, R. Edmunds, F. Rice, and A. M. Agogino, "Tensegrity robot locomotion under limited sensory inputs via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6260–6267.

[3] D. Surovik, K. Wang, M. Vespignani, J. Bruce, and K. E. Bekris, "Adaptive Tensegrity Locomotion: Controlling a Compliant Icosahedron with Symmetry-Reduced Reinforcement Learning," *International Journal of Robotics Research (IJRR)*, 2019.

[4] D. Surovik, J. Bruce, K. Wang, M. Vespignani, and K. Bekris, "Any-axis tensegrity rolling via symmetry-reduced reinforcement learning," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 411–421.

[5] D. S. Shah, J. W. Booth, R. L. Baines, K. Wang, M. Vespignani, K. Bekris, and R. Kramer-Bottiglio, "Tensegrity robotics," *Soft robotics*, 2021.

[6] NASA, "NASA Tensegrity Robotics Toolkit," Accessed 2020, https://github.com/NASA-Tensegrity-Robotics-Toolkit/NTRTsim.

[7] B. T. Mirletz, I.-W. Park, R. D. Quinn, and V. SunSpiral, "Towards bridging the reality gap between tensegrity simulation and robotic hardware," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5357–5363.

[8] K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," *Journal of the royal society interface*, vol. 11, no. 98, p. 20140520, 2014.

[9] K. Wang, M. Aanjaneya, and K. Bekris, "A first principles approach for data-efficient system identification of spring-rod systems via differentiable physics engines," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., vol. 120. PMLR, 10–11 Jun 2020, pp. 651–665. [Online]. Available: https://proceedings.mlr.press/v120/wang20b.html

[10] ——, "Sim2sim evaluation of a novel data-efficient differentiable physics engine for tensegrity robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1694–1701.

[11] ——, "A recurrent differentiable engine for modeling tensegrity robots trainable with low-frequency data," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[12] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[13] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin, "Scalable differentiable physics for learning and control," *arXiv preprint arXiv:2007.02168*, 2020.

[14] M. Vespignani, J. M. Friesen, V. SunSpiral, and J. Bruce, "Design of superball v2, a compliant tensegrity robot for absorbing large impacts," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2865–2871, 2018.

[15] C. Paul, F. J. Valero-Cuevas, and H. Lipson, "Design and control of tensegrity robots for locomotion," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 944–957, 2006.

Appendix

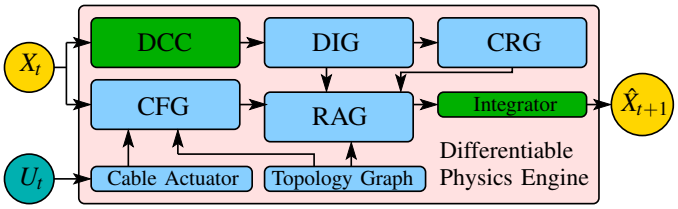

Fig. 5: The physics engine takes the current robot state X_t and control U_t as inputs and predicts the next state $\hat{X}_{t+1}$. Compared to previous attempts [10], this work introduces recurrent training (as shown in Fig. 2), a new numerical integrator, a progressive training pipeline and a new Differentiable Collision Checker (DCC) to account for the frequency mismatch. DIG: Dynamic Interaction Graph, CRG: Collision Response Generator, CFG: Cable Force Generator, RAG: Rod Acceleration Generator.

Physics Engine Components

At the core of the proposed approach lies a differentiable physics engine, shown in Fig. 5, which builds on top of previous work [10]. The engine brings together a series of analytical models based on first principles, which are linear and differentiable. The input to the engine is the current robot state X_t and the instantaneous control U_t. The engine internally stores a representation of the robot in a static topology graph indicating the connectivity of rods and cables. The control U_t is passed to a Cable Actuator module, which maps the control to desired cable rest-lengths. Together with the topological graph, the cable actuator informs the Cable Force Generator (CFG), which is responsible to predict the forces applied on the rods due to the cables given the latest robot state. In parallel and given the robot state, a Differentiable Collision Checker (DCC) detects collisions and informs a Dynamic Interaction Graph (DIG), which stores the colliding bodies (either rod-to-rod or rod-to-ground) and the corresponding contact points. This information is passed to a Collision Response Generator (CRG), which is responsible to compute the reaction forces applied to the rods. The forces and torques from the cables (as computed by CFG) and those from contacts (as computed by CRG) are forwarded to the Rod Acceleration Generator (RAG), which computes the linear and angular acceleration experienced by the rods. These accelerations and torques are integrated by an Integrator module so as to update the robot state.