Efficient Differentiable Articulated Dynamics

Yi-Ling Qiao*, Junbang Liang*, Vladlen Koltun, and Ming Lin

Presenter: Yi-Ling Qiao
Contact: yilingq@cs.umd.edu
Code & data: https://github.com/YilingQiao/diffarticulated

Content

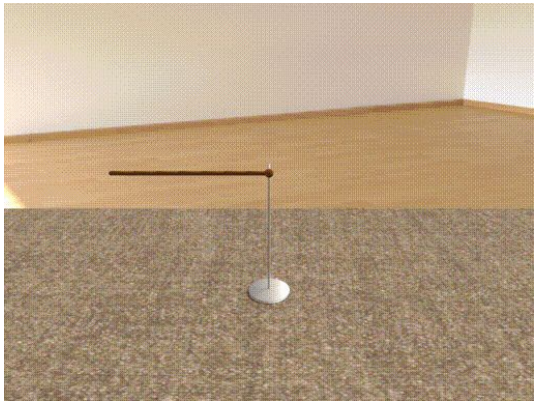
- Motivation
- Related Work
- Our Method
  - Differentiating the simulation
  - Application to reinforcement learning
- Results

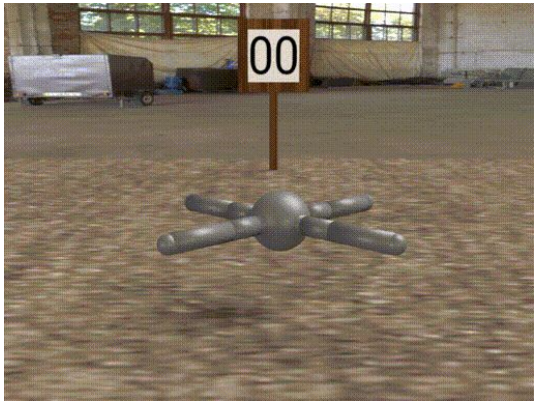
Content

- Motivation
- Related Work
- Our Method
  - Differentiating the simulation
  - Application to reinforcement learning
- Results

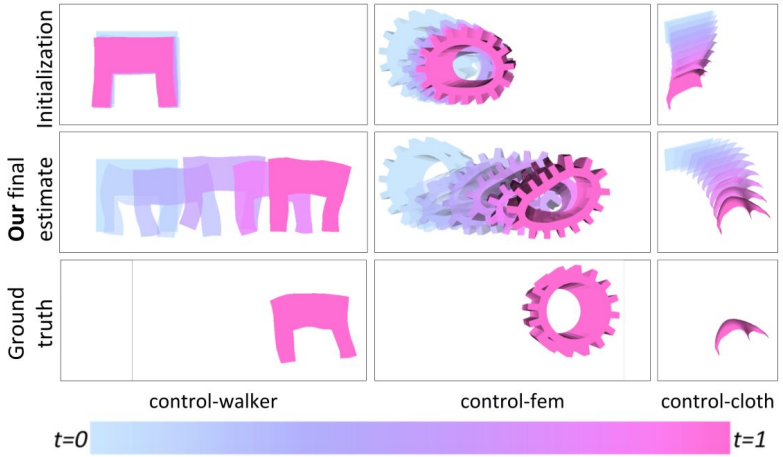
Motivation

- Differentiable articulated body simulation as a network layer
  - Control physical systems
  - Enhance reinforcement learning
  - Estimate physics parameters





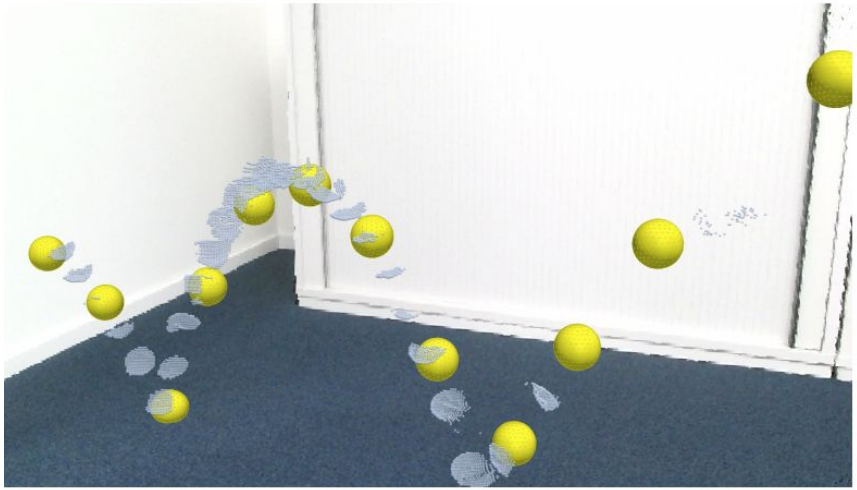
Applications



[2] Murthy et al. (2021)

[5] Song et al. (2020)

Related/concurrent work



[1] Geilinger et al. (2020)

[7] Werling et al. (2021)

References

- [1] Geilinger, M., Hahn, D., Zehnder, J., Bacher, M., Thomaszewski, ¨ B., and Coros, S. ADD: analytically differentiable dynamics for multi-body systems with frictional contact. ACM Trans. Graph., 2020.

- [2] Murthy, J. K., Macklin, M., Golemo, F., Voleti, V., Petrini, L.,Weiss, M., Considine, B., Parent-L´evesque, J., Xie, K., Erleben,K., Paull, L., Shkurti, F., Nowrouzezahrai, D., and Fidler, S.gradsim: Differentiable simulation for system identification and visuomotor control. In ICLR, 2021.

- [3] Liang, J., Lin, M. C., and Koltun, V. Differentiable cloth simulation for inverse problems. In NeurIPS, 2019.

- [4] Qiao, Y.-L., Liang, J., Koltun, V., and Lin, M. C. Scalable differentiable physics for learning and control. In ICML, 2020.

References

- [5] Song, C. and Boularias, A. Learning to slide unknown objects with differentiable physics simulations. In Robotics: Science and Systems (RSS), 2020a

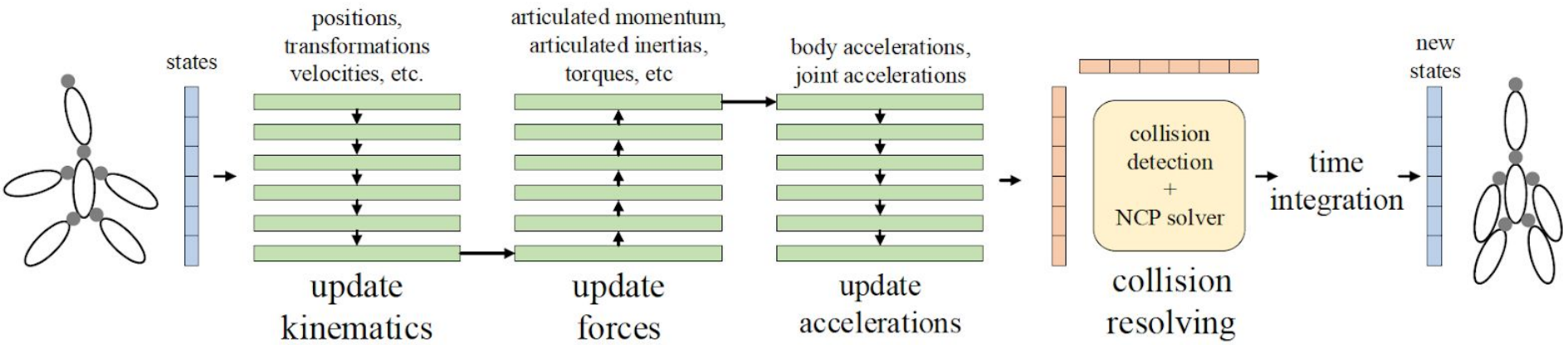
- [6] Takahashi, T., Liang, J., Qiao, Y.-L., and Lin, M. C. Differentiable fluids with solid coupling for learning and control. In AAAI, 2021.

- [7] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos and C. Karen Liu. Fast and Feature-Complete Differentiable Physics for Articulated Rigid Bodies with Contact. In RSS, 2021.

Content

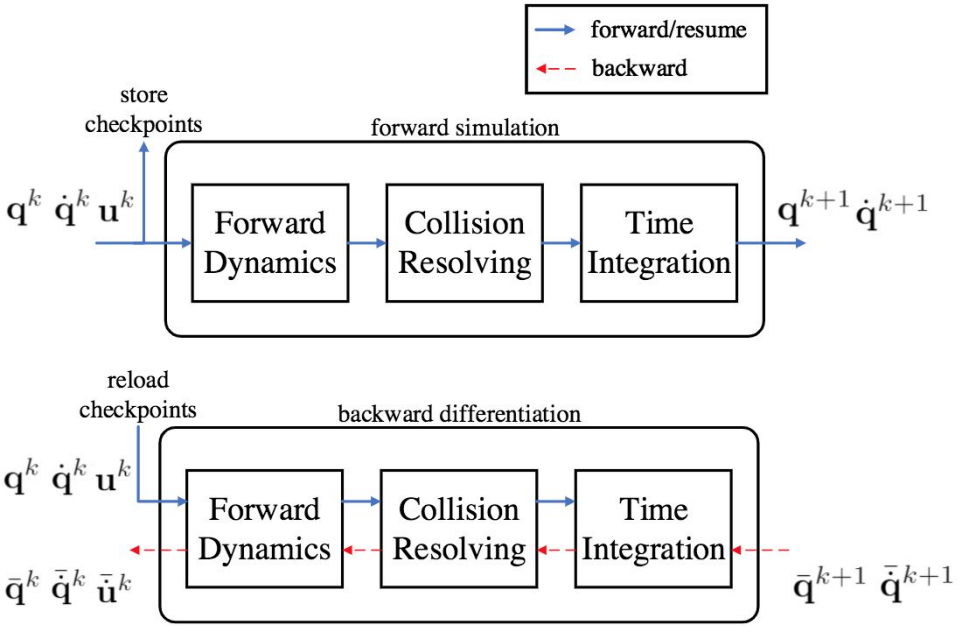
- Motivation
- Related Work
- Our Method
  - Differentiating the simulation
  - Application to reinforcement learning
- Results

Workflow of one simulation step

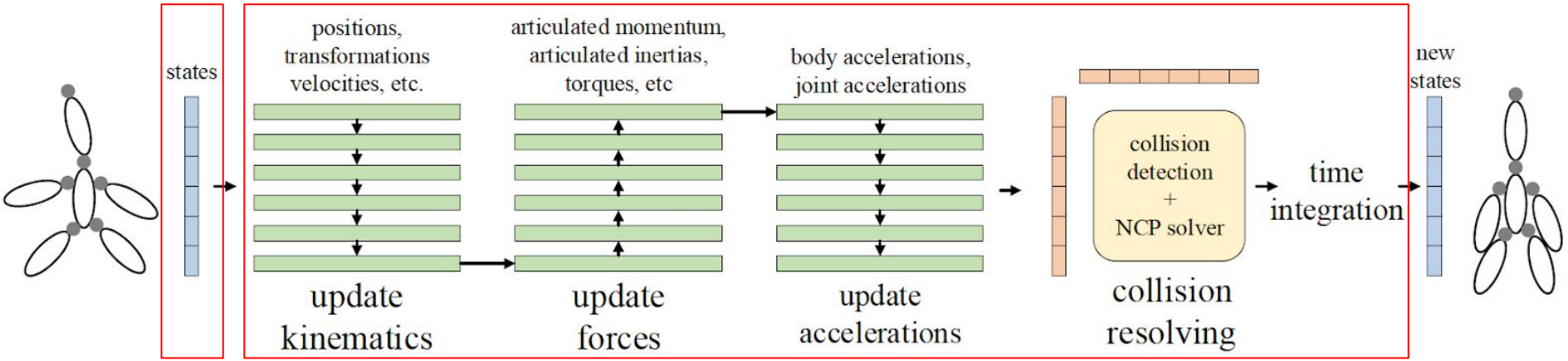


Checkpointing

Forward and backward workflow with checkpointing scheme



Checkpointing



checkpoints

Intermediate variables

Application with Reinforcement Learning

- Sample enhancement
  - Increase sample efficiency
  - Faster convergence
- Policy enhancement
  - Update the policy using analytic gradients
  - Better scalability in high dimensionality

Sample Enhancement

- Idea: Use simulation gradients to generate extra nearby examples
- Point sample → patch sample
  - Faster convergence

$$a_k = a_0 + \Delta a_k$$

$$s'_k = s'_0 + \frac{\partial s'_0}{\partial a_0} \Delta a_k$$

$$r_k = r_0 + \frac{\partial r_0}{\partial a_0} \Delta a_k$$

Enabled by differentiable simulation!

a: action
s: observation
s': next-step observation
r: reward

Policy Enhancement

- Idea: Use simulation gradients to compute better policy gradients
- Use one-step rollout to approximate the action gradients

$$\mathcal{L}_\mu = -Q(s, \mu(s)) + Z$$

Soft Actor-Critic

$$\frac{\partial Q(s,a)}{\partial a} = \frac{\partial r}{\partial a} + \gamma \frac{\partial Q(s', \mu(s'))}{\partial s'} \frac{\partial s'}{\partial a}$$

$$\mathcal{L}'_\mu = -\frac{\partial Q(s,a)}{\partial a}\mu(s) + Z$$

Ours

Enabled by differentiable simulation!

a: action
s: observation
s': next-step observation
r: reward
Q: critic network
μ: policy network
Z: regularization term

Content

- Motivation
- Related Work
- Our Method
  - Adjoint derivation
  - Application with reinforcement learning
- Results

Results - Performance

- Compare the runtime and memory usage.
- Scene: One Laikago released from the air and hitting the ground
  - Scale the simulation length: 50, 100, 500, 1000, 5000 steps
- Comparisons:
  - Use autodiff tools in the same simulation pipeline

Results - Performance

- Compare the runtime and memory usage.
- Scene: A Laikago released from the air and hitting the ground
- Our method has the highest speed and the lowest memory usage
    - x10 faster than autodiff tools with 1% of memory usage

steps	50	100	500	1000	5000
ADF	25.7	25.5	25.1	32.1	58.4
Ceres	27.2	27.5	27.2	34.0	58.2
CppAD	2.4	2.4	2.3	2.3	4.5
JAX	53.3	46.1	43.1	42.7	42.3
PyTorch	195.6	192.2	199.2	192.8	N/A
Ours	**0.3**	**0.3**	**0.2**	**0.2**	**0.2**

Forward simulation time (ms) per step

steps	50	100	500	1000	5000
ADF	25.7	25.5	25.1	32.1	58.4
Ceres	27.2	27.5	27.2	34.0	58.2
CppAD	2.4	2.4	2.3	2.3	4.5
JAX	53.3	46.1	43.1	42.7	42.3
PyTorch	195.6	192.2	199.2	192.8	N/A
Ours	**0.3**	**0.3**	**0.2**	**0.2**	**0.2**

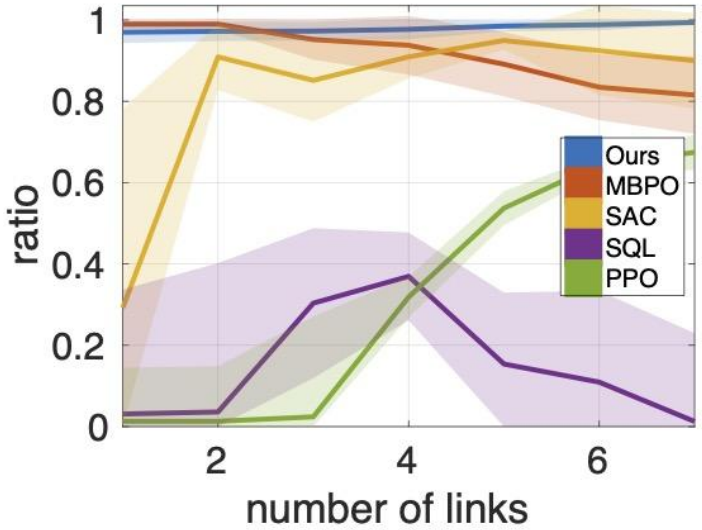
Peak Memory (MB)

Policy Enhancement

- Scenario: N-link pendulum
- Objective: reaching the highest point within 100 frames
- Reward
  - -dist_to_target^2
- Baseline: MBPO, SAC, SQL, PPO
- Number of links: 1-7
- Number of training epochs: 100 * n_links
  - Samples per epoch: 100

Policy Enhancement

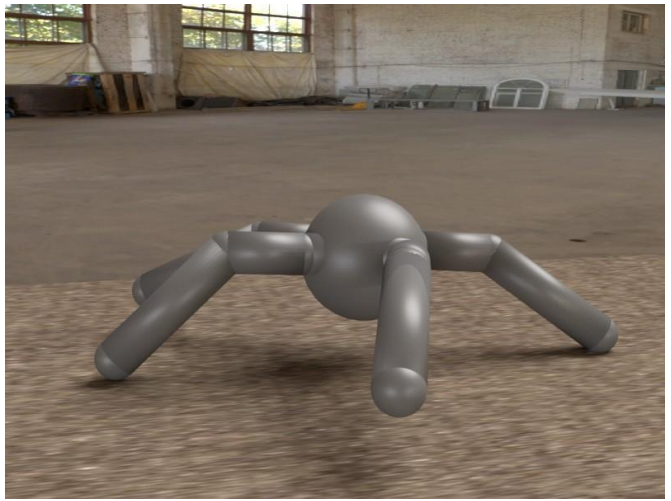
- Test metric: Best relative reward
  - Absolute reward / maximum possible reward (reaching exactly the target)



Our method scales with increasing system complexity

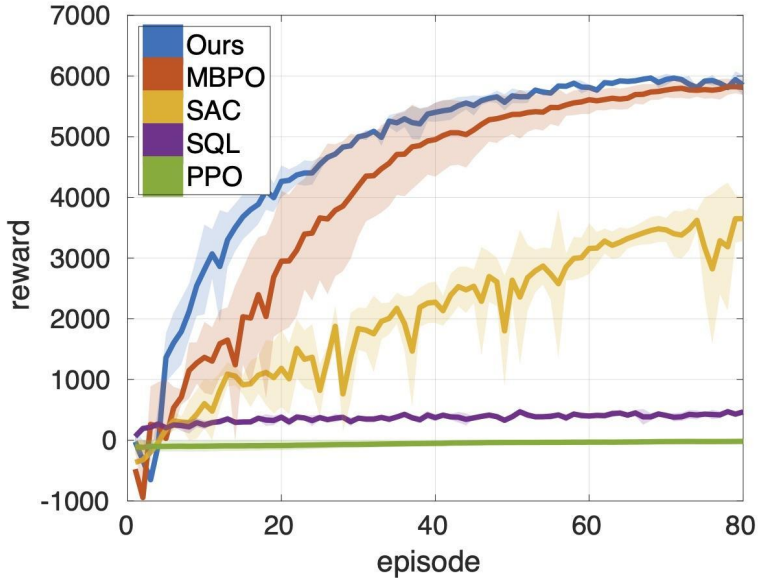
Sample Enhancement

- Scenario: Mujoco Ant
- Objective: walking towards +x axis
- Reward
  - v_x - sum(action^2)
- Baseline: MBPO, SAC, SQL, PPO
- Number of training epochs: 100
  - Samples per epoch: 1000



Sample Enhancement

- Test metric:
  - Maximum (absolute) reward



Our method achieves the same best reward and converges faster

Thanks for your attention!