PhysWM: Physical World Models for Robot Learning*

Marc Otto[1,*,†], Octavio Arriaga[2,*,†], Chandandeep Singh[1,†], Jichen Guo[2,†] and Frank Kirchner[1,2]

[1]Robotics Innovation Center, DFKI GmbH, Robert-Hooke-Straße 1, 28359 Bremen, Germany
[2]Robotics Research Group, University of Bremen, Bibliothekstraße 1, 28359 Bremen, Germany

Abstract

Within the last decade machine learning methods have shown remarkable results in pattern recognition tasks and behavior learning. However, when applied to real-world robotics tasks, these approaches have limitations, such as sample inefficiency and limited generalization to out-of-distribution samples. Despite the availability of precise physics in simulation engines, model-based reinforcement learning (RL) resorts to learning an approximation of these dynamics. On the other hand, optimal control approaches often assume a static, complete model of the world, addressing the simulation-reality gap by adding low level controllers. In order to handle these issues, we propose a hybrid simulator consisting of differentiable physics and rendering modules, which employ symbolic representations and reduce the model complexity of neural policies, while retaining gradient computation for model and behavior optimization. Moreover, this reduced parametric representation enables the use of Bayesian inference to estimate the uncertainty over physical parameters. This uncertainty quantification allows us to generate a curriculum of exploration behaviors for continuously improving the world model.

Keywords

differentiable physics, neural networks, uncertainty quantification, robot learning

1. Introduction

Despite remarkable success in pattern recognition and behavior learning tasks, developing intelligent robots remains a challenge for artificial intelligence algorithms [1]. Robots require an adaptable environment representation to plan movements under contact while interacting with objects with unknown properties such as mass, friction or shape. Moreover, performing tasks alongside humans requires autonomous systems that can explain their behavior and accurately quantify their uncertainty. The current machine learning paradigm addresses these issues by acquiring a large dataset of possible circumstances and testing generalization in an unseen fraction of our collected samples. However, this formulation has certain problems within the robotics domain [2]. The space of all possible robot experiences is too large and datasets for

NeSy'23: 17th International Workshop on Neural-Symbolic Learning and Reasoning, Certosa di Pontignano, Siena, Italy
*Corresponding author.
†These authors contributed equally.
marc.otto@dfki.de (M. Otto); arriagac@uni-bremen.de (O. Arriaga); chandandeep.singh@dfki.de (C. Singh); jichen@uni-bremen.de (J. Guo); frank.kirchner@dfki.de (F. Kirchner)
0000-0002-5800-0578 (M. Otto); 0000-0002-8099-2534 (O. Arriaga); 0000-0003-4100-1002 (C. Singh); 0000-0002-0247-1987 (J. Guo); 0000-0002-1713-9784 (F. Kirchner)

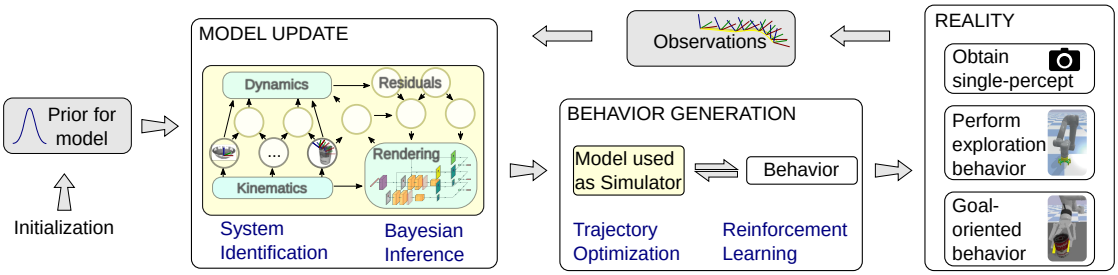


Figure 1: PhysWM world model framework

certain robotic tasks require millions of samples. Moreover, optimizing millions of parameters to train a model may require power-intensive GPUs. As pointed out by [3, 4], deep networks and black-box AI in general ignore known physical equations and often remain uninterpretable solutions.

Thus, we propose an architecture that leverages new advances in inverse rendering [5, 6], differentiable physics [7, 8], probabilistic programming [9, 10, 11] and curriculum learning [12] to equip robots with an adaptable world model. We hypothesize, that our framework can be used to generate efficient exploration behaviors in order to quantify the scene parameters.

2. Approach

We propose to use a hybrid differentiable physics simulation which is a combination of a physics engine and neural networks, similar to the one illustrated in [13]. Extended with parameter uncertainty, the simulation becomes a probabilistic graphical model. The model parameters are updated when observations have been gathered by the execution of behaviors. The iterative model update and behavior optimization are inspired by the Estimation-Exploration Algorithm [14] and by maximizing model disagreement [15].

Figure 1 presents our approach to obtaining and updating a world model using exploration behaviors. Given a prior for all model parameters and an observation (an image of the scene), one can use Bayesian inference with a differentiable renderer to obtain the posterior over those parameters. These scene parameters include the object's shape, pose, color material properties as well as the scene's lighting. These identified quantities are used by the hybrid simulation to model interactions with a robot manipulator. Using our simulation, RL or trajectory optimization can generate behaviors in order to explore the environment further; for example, by lifting an object to validate its mass or pushing an object to obtain a friction model. Moreover, exploration can also capture an image from another perspective in order to validate an object's dimension. The model is continuously updated using Bayesian inference until the model is accurate enough to generate the goal-oriented behavior specified by a user.

Tasks such as poking, pushing, pick-and-place, stacking, or billiard are used for evaluating approaches for building world models [16, 17, 18]. For instance, poking is used in [17] to learn an intuitive physical world model. We re-use these tasks as benchmarks and aim at pouring water and learning curling behaviors for unknown objects to test our framework's generalization.

3. Adaptable world model representation

Differentiable physics and rendering Differentiable physics engines can be used by learning methods and optimal control and provide gradients for the optimization criteria. An overview of differentiable simulators and applications is outlined in Appendix A.1. The approach proposed in this paper uses a combination of differentiable physics simulation and a differentiable renderer to create a model of the world. Our differentiable simulator is hybrid and is augmented with neural networks to make it more data-efficient and generalizable than data-driven models, thereby allowing efficient reduction of sim-to-real gap [13]. The proposed rendering engine is built on JAX [19], which enables it to render images on CPU, GPU, and TPU. Additionally, it maintains compatibility with modern optimization libraries [20], deep learning frameworks [21], probabilistic programming languages [9], and posterior sampling libraries [22].

Probabilistic graphical model The world model can be represented as a probabilistic graphical model [23] in a hybrid differentiable physics simulation. It consists of multiple nodes corresponding to the simulation parameters associated to an object in the environment. Links between objects represent possible causal relationships. Optimizing probabilistic programs often resorts to sampling algorithms such as Markov Chain Monte Carlo (MCMC), which are known to be computationally expensive. In order to counter the computational costs of MCMC, probabilistic programming languages have been built using hardware-accelerated kernels [24]. Nevertheless, world models can be learned, as in [25, 26, 18] and can be used as environment representations for optimal control. Furthermore, the world model is not static but evolves w.r.t. changes in the environment, and it can be explicitly updated using causal interventions [27].

4. Simulation parameter estimation

Optimization of parameter distributions Uncertainty and noise are two major concerns in robotics [28]. Using active inference, agents improve the predictions made by the internal world model and behave in a way that prevents the occurrence of ambiguity [29]. In order to account for uncertainty, simulations are often enhanced by dynamics randomization or model ensembles making robot behaviors learned in simulation more robust for transfer to the real robot [30, 31, 32, 33]. Expert knowledge is required to set up the randomization mean and variance, which has been reduced via adaptive domain randomization in [34]. Robot specific choices of relevant parameters [35] and the computational effort of these sampling-based methods, can be overcome when the uncertainty of parameters is part of the simulation, and it is propagated to behavior outcomes. Thus, we propose to update model parameter *distributions* instead of single values. Given our prior distributions and observations, we compute the posterior of simulation parameters using MCMC as exemplified in Figure 3 in Appendix A.2.

Exploration Behaviors A white-box model of robot dynamics can be obtained by system identification with robot movements, explicitly optimized to obtain suitable data, called excitation trajectories. Our hybrid approach of model-based and data-driven simulation components can profit from a similar exploration strategy. Ideally, one would compute the expected entropy

reduction of each possible action for probing the environment. As this doesn't scale well to large state spaces, local optimizations of the expected information gain have been proposed [36]. In [15], the idea of maximizing model disagreement is applied as an intrinsic motivation to explore the candidate models' areas of uncertainty. As we model simulation parameters as distributions, we can define the loss function to explicitly reward uncertainty in the outcome. We expect this approach to explore the environment more effectively than using the sampling-based disagreement measure, since the latter relies on a fixed number of candidate models to only approximate parameter uncertainty.

5. Curriculum Learning with Complexity levels

Complexity levels The parameter set of a world model can become arbitrarily large, as the model is fine-tuned to represent reality more detailed. As shown by [37] a multi-fidelity simulation for RL, in which the same environment is modeled by simulations of different complexity, can reduce the training time spent in the more complex environments, including real trials. Simulations of lower and higher fidelity share parameters, making one complexity level profit from model improvements on another. We define an iterative approach to the minimum required complexity [38] that describes a world model, a policy and a reward model for a given task (see Table 2 in Appendix A.3). We expect that the search in the more abstract simulation is faster and thus a global search is feasible.

Automatizing the curriculum For a highly autonomous system improving its world model and behavior, the ability to switch between complexity levels is needed. In curriculum sets [39], the agent focuses on improving the modules for which it is making most progress, while in active domain randomization [33], the parameter distribution is adjusted automatically to select an intermediate level of difficulty. This is the driver of a curriculum as the learning agent improves on the given environment setting. Once the task is solved for the current simulation values, settings that were previously too difficult, become feasible at *intermediate difficulty*. Extending this principle to complexity levels, we can temporarily exclude reward model components as well as physical aspects to focus learning on a part of the policy and parameters as exemplified in Appendix A.4. We hypothesize that our learning framework will therefore allow sample efficient model updates with MCMC and policy updates with RL.

6. Conclusion and Outlook

This work presents our approach to overcoming the sample inefficiency of data-driven methods for estimating simulation parameters. In order to do so, behaviors for exploring the remaining world model uncertainties are generated and the uncertainty is quantified explicitly or via candidate model disagreement. For efficiently generating behaviors, deep RL and optimal control can use the gradients provided by the differentiable simulation directly. Creating an adaptable world model of appropriate complexity is addressed by automatizing a curriculum in which the model complexity is adapted based on experience gathered in the given scenario.

Acknowledgments

This work has been performed in the PhysWM project funded by the German Aerospace Center (DLR) with federal funds (Grant numbers 50RA2126A and 50RA2126B) from the German Federal Ministry of Economic Affairs and Climate Action (BMWK). We would like to thank Dr.-Ing. Alexander Fabisch and Dr. rer. nat. Shivesh Kumar as well the reviewers from the NeSy workshop for their insightful comments on our paper.

References

[1] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, et al., The grand challenges of science robotics, Science robotics 3 (2018) eaar7650.

[2] J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, The International Journal of Robotics Research 32 (2013) 1238–1274.

[3] R. Yu, P. Perdikaris, A. Karpatne, Physics-guided ai for large-scale spatiotemporal data, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 4088–4089.

[4] M. Lutter, J. Peters, Combining physics and deep learning to learn continuous-time dynamics models, arXiv preprint arXiv:2110.01894 (2021).

[5] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, T. Aila, Modular primitives for high-performance differentiable rendering, ACM Transactions on Graphics 39 (2020).

[6] W. Jakob, S. Speierer, N. Roussel, D. Vicini, Dr.jit: A just-in-time compiler for differentiable rendering, Transactions on Graphics (Proceedings of SIGGRAPH) 41 (2022). doi:10.1145/3528223.3530099.

[7] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, O. Bachem, Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL: http://github.com/google/brax.

[8] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: 2012 IEEE/RSJ international conference on intelligent robots and systems, IEEE, 2012, pp. 5026–5033.

[9] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, R. A. Saurous, Tensorflow distributions, arXiv preprint arXiv:1711.10604 (2017).

[10] M. F. Cusumano-Towner, F. A. Saad, A. K. Lew, V. K. Mansinghka, Gen: A general-purpose probabilistic programming system with programmable inference (2019) 221–236. URL: http://doi.acm.org/10.1145/3314221.3314642. doi:10.1145/3314221.3314642.

[11] D. Phan, N. Pradhan, M. Jankowiak, Composable effects for flexible and accelerated probabilistic programming in numpyro, arXiv preprint arXiv:1912.11554 (2019).

[12] R. Portelas, C. Colas, L. Weng, K. Hofmann, P.-Y. Oudeyer, Automatic Curriculum Learning For Deep RL: A Short Survey, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, International Joint Conferences

on Artificial Intelligence Organization, Yokohama, Japan, 2020, pp. 4819–4825. URL: https://www.ijcai.org/proceedings/2020/671. doi:10.24963/ijcai.2020/671.
[13] E. Heiden, D. Millard, E. Coumans, Y. Sheng, G. S. Sukhatme, Neuralsim: Augmenting differentiable simulators with neural networks, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 9474–9481.
[14] J. Bongard, H. Lipson, Nonlinear System Identification Using Coevolution of Models and Tests, IEEE Transactions on Evolutionary Computation 9 (2005) 361–384. URL: http://ieeexplore.ieee.org/document/1492385. doi:10.1109/TEVC.2005.850293.
[15] D. Pathak, D. Gandhi, A. Gupta, Self-Supervised Exploration via Disagreement, in: Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019, pp. 5062–5071. URL: https://proceedings.mlr.press/v97/pathak19a.html, iSSN: 2640-3498.
[16] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, Y. Bengio, B. Schölkopf, M. Wüthrich, S. Bauer, Causalworld: A robotic manipulation benchmark for causal structure and transfer learning, arXiv preprint arXiv:2010.04296 (2020).
[17] P. Agrawal, A. Nair, P. Abbeel, J. Malik, S. Levine, Learning to Poke by Poking: Experiential Learning of Intuitive Physics, 2017. URL: http://arxiv.org/abs/1606.07419, arXiv:1606.07419 [cs].
[18] O. Biza, T. Kipf, D. Klee, R. Platt, J.-W. van de Meent, L. L. Wong, Factored world models for zero-shot generalization in robotic manipulation, arXiv preprint arXiv:2202.05333 (2022).
[19] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, et al., Jax: composable transformations of python+ numpy programs (2018).
[20] I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, e. a. Cai, The DeepMind JAX Ecosystem, 2020. URL: http://github.com/deepmind.
[21] P. Kidger, C. Garcia, Equinox: neural networks in jax via callable pytrees and filtered transformations, arXiv preprint arXiv:2111.00254 (2021).
[22] J. Lao, R. Louf, Blackjax: A sampling library for JAX, 2020. URL: http://github.com/blackjax-devs/blackjax.
[23] N. Gothoskar, M. Cusumano-Towner, B. Zinberg, M. Ghavamizadeh, F. Pollok, A. Garrett, J. B. Tenenbaum, D. Gutfreund, V. K. Mansinghka, 3DP3: 3D Scene Perception via Probabilistic Programming, 2021. URL: http://arxiv.org/abs/2111.00312. doi:10.48550/arXiv.2111.00312, arXiv:2111.00312 [cs].
[24] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, R. A. Saurous, TensorFlow Distributions (2017). URL: http://arxiv.org/abs/1711.10604. doi:10.48550/arXiv.1711.10604, arXiv:1711.10604 [cs, stat].
[25] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, K. Goldberg, Daydreamer: World models for physical robot learning, in: Conference on Robot Learning, PMLR, 2023, pp. 2226–2240.
[26] L. Zhang, G. Yang, B. C. Stadie, World model as a graph: Learning latent landmarks for planning, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of *Proceedings of Machine Learning Research*, PMLR, 2021, pp. 12611–12620. URL: https://proceedings.mlr.press/v139/zhang21x.html.
[27] J. Pearl, Causal inference, Causality: objectives and assessment (2010) 39–58.
[28] A. Fabisch, C. Petzoldt, M. Otto, F. Kirchner, A survey of behavior learning applications in robotics–state of the art and perspectives, arXiv preprint arXiv:1906.01868 (2019).

[29] T. Parr, G. Pezzulo, K. J. Friston, Active Inference: The Free Energy Principle in Mind, Brain, and Behavior, 2022. URL: https://direct.mit.edu/books/oa-monograph/5299/Active-InferenceThe-Free-Energy-Principle-in-Mind. doi:10.7551/mitpress/12441.001.0001.

[30] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter, Learning agile and dynamic motor skills for legged robots, Science Robotics 4 (2019) eaau5872. URL: https://www.science.org/doi/abs/10.1126/scirobotics.aau5872. doi:10.1126/scirobotics.aau5872, publisher: American Association for the Advancement of Science.

[31] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, L. Zhang, Solving Rubik's Cube with a Robot Hand, arXiv:1910.07113 [cs, stat] (2019). URL: http://arxiv.org/abs/1910.07113, arXiv: 1910.07113.

[32] Y. Wu, W. Yan, T. Kurutach, L. Pinto, P. Abbeel, Learning to Manipulate Deformable Objects without Demonstrations: 16th Robotics: Science and Systems, RSS 2020, Robotics (2020). URL: http://www.scopus.com/inward/record.url?scp=85127981155&partnerID=8YFLogxK. doi:10.15607/RSS.2020.XVI.065, publisher: MIT Press Journals.

[33] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, L. Paull, Active Domain Randomization, in: Proceedings of the Conference on Robot Learning, PMLR, 2020, pp. 1162–1176. URL: https://proceedings.mlr.press/v100/mehta20a.html, iSSN: 2640-3498.

[34] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, D. Fox, Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience, in: 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 8973–8979. doi:10.1109/ICRA.2019.8793789, iSSN: 2577-087X.

[35] Z. Xie, X. Da, M. van de Panne, B. Babich, A. Garg, Dynamics Randomization Revisited: A Case Study for Quadrupedal Locomotion, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 4955–4961. doi:10.1109/ICRA48506.2021.9560837, iSSN: 2577-087X.

[36] A. T. Taylor, T. A. Berrueta, T. D. Murphey, Active learning in robotics: A review of control principles, Mechatronics 77 (2021) 102576. URL: https://www.sciencedirect.com/science/article/pii/S0957415821000659. doi:10.1016/j.mechatronics.2021.102576.

[37] M. Cutler, T. J. Walsh, J. P. How, Real-World Reinforcement Learning via Multifidelity Simulators, IEEE Transactions on Robotics 31 (2015) 655–671. URL: http://ieeexplore.ieee.org/ielx7/8860/7117487/07106543.pdf?tp=&arnumber=7106543&isnumber=7117487. doi:10.1109/TRO.2015.2419431.

[38] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, Texts in Computer Science, Springer International Publishing, Cham, 2019. URL: http://link.springer.com/10.1007/978-3-030-11298-1. doi:10.1007/978-3-030-11298-1.

[39] C. Colas, P. Fournier, M. Chetouani, O. Sigaud, P.-Y. Oudeyer, CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning, in: Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019, pp. 1331–1340. URL: https://proceedings.mlr.press/v97/colas19a.html, iSSN: 2640-3498.

[40] J. Liang, M. C. Lin, Differentiable physics simulation, in: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020.

[41] P. M. Wensing, S. Kim, J.-J. E. Slotine, Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution, IEEE Robotics and Automation Letters 3 (2017) 60–67.
[42] T. Lee, P. M. Wensing, F. C. Park, Geometric robot dynamic identification: A convex programming approach, IEEE Transactions on Robotics 36 (2019) 348–365.
[43] F. Meier, A. Wang, G. Sutanto, Y. Lin, P. Shah, Differentiable and learnable robot models, arXiv preprint arXiv:2202.11217 (2022).
[44] E. Heiden, D. Millard, H. Zhang, G. S. Sukhatme, Interactive differentiable simulation, arXiv preprint arXiv:1905.10706 (2019).
[45] A. Patel, S. L. Shield, S. Kazi, A. M. Johnson, L. T. Biegler, Contact-implicit trajectory optimization using orthogonal collocation, IEEE Robotics and Automation Letters 4 (2019) 2242–2249.
[46] A. O. Onol, P. Long, T. Padlr, A comparative analysis of contact models in trajectory optimization for manipulation, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 1–9.
[47] K. Werling, D. Omens, J. Lee, I. Exarchos, C. K. Liu, Fast and feature-complete differentiable physics for articulated rigid bodies with contact, arXiv preprint arXiv:2103.16021 (2021).
[48] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, et al., Theano: A python framework for fast computation of mathematical expressions, arXiv e-prints (2016) arXiv–1605.
[49] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).
[50] D. Maclaurin, Modeling, inference and optimization with composable differentiable procedures, 2016.
[51] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, Journal of Marchine Learning Research 18 (2018) 1–43.
[52] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, J. Z. Kolter, End-to-end differentiable physics for learning and control, Advances in neural information processing systems 31 (2018).
[53] J. Degrave, M. Hermans, J. Dambre, et al., A differentiable physics engine for deep learning in robotics, Frontiers in neurorobotics (2019) 6.
[54] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, F. Durand, DiffTaichi: Differentiable Programming for Physical Simulation, 2020. URL: http://arxiv.org/abs/1910.00935. doi:10.48550/arXiv.1910.00935, arXiv:1910.00935 [physics, stat].
[55] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, O. Bachem, Brax – A Differentiable Physics Engine for Large Scale Rigid Body Simulation, 2021. URL: http://arxiv.org/abs/2106.13281, arXiv:2106.13281 [cs].
[56] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, S. Coros, Add: Analytically differentiable dynamics for multi-body systems with frictional contact, ACM Transactions on Graphics (TOG) 39 (2020) 1–15.
[57] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, P. Agrawal, An end-to-end differentiable framework for contact-aware robot design, arXiv preprint arXiv:2107.07501 (2021).
[58] J. Liang, M. Lin, V. Koltun, Differentiable cloth simulation for inverse problems, Advances

in Neural Information Processing Systems 32 (2019).
[59] Y.-L. Qiao, J. Liang, V. Koltun, M. C. Lin, Scalable differentiable physics for learning and control, arXiv preprint arXiv:2007.02168 (2020).
[60] C. Schenck, D. Fox, Spnets: Differentiable fluid dynamics for deep neural networks, in: Conference on Robot Learning, PMLR, 2018, pp. 317–335.
[61] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).
[62] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, A. M. Dollar, Yale-cmu-berkeley dataset for robotic manipulation research, The International Journal of Robotics Research 36 (2017) 261–268.
[63] K. Mülling, J. Kober, O. Kroemer, J. Peters, Learning to select and generalize striking movements in robot table tennis, International Journal of Robotics Research 32 (2013) 263–279. Publisher: Sage Publications, Inc. Thousand Oaks, CA, USA.

A. Appendix

A.1. Differentiable physics simulation

Differentiable physics simulation is a computational tool that utilizes gradient-based techniques for learning and control of physical system [40]. During the past few years, it has been successfully applied in many areas, such as system identification [41, 42], design optimization [43, 44] and motion optimization [45, 46], as shown in Figure 2, which also emphasizes the wide range of potential applications. As mentioned in [47], leveraging the recent developments in automatic differentiation techniques and libraries [48, 49, 19, 50, 51], various differentiable physics engines have been proposed to address the control and parameter estimation issues for rigid bodies [47, 52, 53, 54, 55, 13, 56, 57], as listed and categorized in Table 1, and non-rigid bodies, such as cloth [58, 59] and fluid [60].

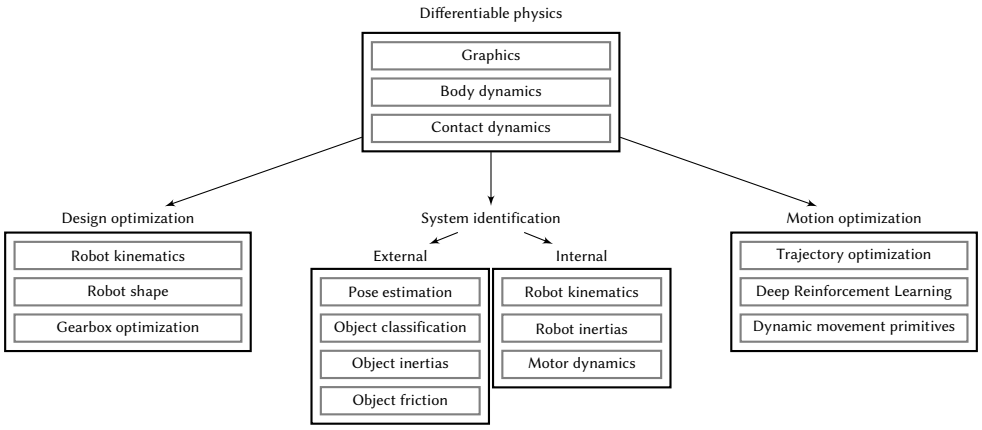


Figure 2: Application of differentiable physics

Table 1
Differentiable engines for articulated rigid bodies, extended from [47]

Engine	Contacts	State	Collisions	Gradients	Language	URDF
MuJoCo [8]	custom	reduced	complete	finite	C++	support
Degrave et al. [53]	impulsive	maximal	primitves	auto	Theano	no
DiffTaichi [54]	impulsive	maximal	primitives	auto	Taichi	no
TinyDiffSim [13]	iterated LCP	reduced	primitives	auto	C++	support
De A.B.-P. et al. [52]	direct LCP	maximal	primitives	symbolic	Pytorch	no
Gelinger et al. [56]	custom	reduced	primitives	symbolic	*not released*	unknown
Nimble [47]	direct LCP	reduced	complete	symbolic	DART	support
DiffREDMax [57]	custom	reduced	primitives	symbolic	C++	support
Brax v1 [55]	impulsive	maximal	primitives	auto	Python(JAX)	support
Brax v2 [7]	custom	reduced	primitives	auto	Python(JAX)	support

A.2. Inverse rendering with MCMC

The initial step of our framework is to obtain a graph-based scene representation using priors for the scene parameters (see Figure 1) as well as a single image to obtain a posterior distribution. Here, a scene with a single object $\mathcal{O}$ (see Figure 3a) is characterized by a transformation $\mathcal{T}$ consisting of translation t, rotation θ and scale s as well as a stochastic material $\mathcal{M}$ consisting of ambient α, diffuse β and color κ and a shape $\mathcal{P}$ taking the values sphere, cylinder and box. Furthermore, it is rendered in an environment $\mathcal{G}$ (flat ground with lights and specific camera perspective) matching the current observation $\mathcal{J}_r$. We define a likelihood function based on RGB-data considering the pixel-wise disparity as well as image features obtained from VGG16 [61]. The posterior (see Figure 3b) of all parameters is computed using the Rosenbluth-Metropolis-Hastings algorithm (RMH). We can use this posterior for sampling, as shown in Figure 3c or as a proto-program to generate similar scenes by excluding parts of the scene graph.

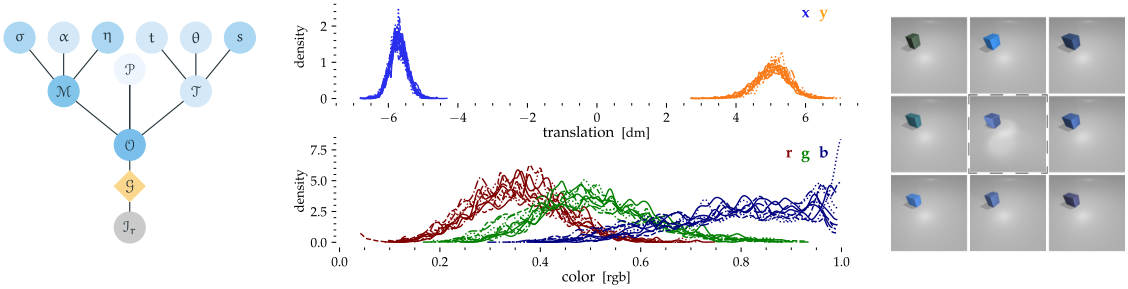
The depicted example is representative for our preliminary results, which indicate that the method can also be applied on out-of-distribution samples to find suitable approximations for rendered objects from the YCB dataset [62].

A.3. Scenario complexity levels

Table 2
Scenario complexity levels

level	goal	reward model components	simulation parameters	policy
1	reach object	object-gripper-distance (OGD)	kinematics (K)	2 waypoints (2WP)
2	pick object	OGD + grasp stability (GS)	K + collisions (C)	3WP/ sequential DMP
3	poke object	OGD + object travel distance	K + C + object dynamics	DMP with end-velocity
4	object curling	OGD + GS + final location	full dynamics	neural network

(1) To reach the object with a robot arm movement defined by start and end point, a kinematics



(a) Object graph (b) Posterior distributions (c) Rendered images

Figure 3: Inverse rendering with MCMC. The scene is represented as a graph (a) in which an object $\mathcal{O}$ has properties with parameters (top row) for which a prior is defined. MCMC is applied to compute the posterior for all properties including the translation and the color as shown in (b). In (c), the target image, highlighted by a dashed frame in the center, and 8 samples from the posterior are depicted.

simulation is sufficient. (2) For picking an object, the gripper's relative pose and collisions with the object are required to allow a sequential movement representation to solve the task. (3) Poking the object with a desired effect intensity such as the resulting object displacement, the simulation needs to include the object's inertia and friction with the ground. A DMP with end-velocity is a suitable behavior representation [63]. (4) A task, such as in the sport curling, requires a stable grasp and accurate release to reach the target location. It can be learned with a neural network policy when dynamics parameters are sufficiently optimized, using policy parameters learned for the previous level as initialization.

A.4. Incomplete physical modelling

From our simulation, physical aspects can be reduced, which means, e.g. in the case of collisions, that some objects' collisions are excluded from the computation. When physical aspects cannot be excluded entirely from the manipulation scenario, such as friction, we make use of the parameter uncertainty and provide feedback to learning and optimization algorithms considering sample-specific, optimal values. For example, a curling behavior may be directed at the correct target but due to an inadequate friction model, it receives a low reward. By setting the friction coefficients temporarily to values of the parameter distribution for which the reward is maximized, the agent can focus on improving the direction of the curling behavior first. On the other hand, in the estimation of model parameters, this allows MCMC to set a subset of parameters while the excluded parameters can take any sample-specific value. Once the reduction of uncertainty of the simulation parameters does not significantly influence the task learning progress, the set of physical aspects modeled in our simulation is increased by choosing the one with the highest gradient w.r.t. the reward.