Co-Designing Hardware and Control for Robot Hands*

Tianjian Chen+, Zhanpeng He+, and Matei Ciocarlie
Department of Mechanical Engineering
Columbia University, New York, NY 10027

Abstract: *Policy gradient methods can be used for mechanical and computational co-design of robot manipulators.*

Roboticists have long argued that embodied agents' intelligence can extend beyond a purely computational brain: hardware plays a key role in determining behavior, and can embody intelligence just like computational components do. Design optimization thus goes hand in hand with deriving effective control algorithms or policies. Here, we focus specifically on the concept of co-optimization (or co-design), which aims to achieve desired behaviors by optimizing mechanical and computational components simultaneously, with and for each other.

A number of foundational studies on co-optimization in robotics used genetic algorithms [5] or evolutionary computation [4] to optimize both hardware and software. The choice of biologically-inspired optimization algorithms seems natural given the well-known evidence of co-optimization in nature, including the evolution of dexterous manipulators [1]. However, when designing artificial agents, many other optimization strategies have also been used, such as efficient gradient-based methods [3]. As more powerful optimization algorithms are introduced, they could potentially enable new directions in co-optimization. In particular, the last decade has seen the advent of GPU-accelerated optimization for powerful computational components such as universal function approximators. Could such methods be used to simultaneously advance mechanical intelligence, in robot hands and beyond?

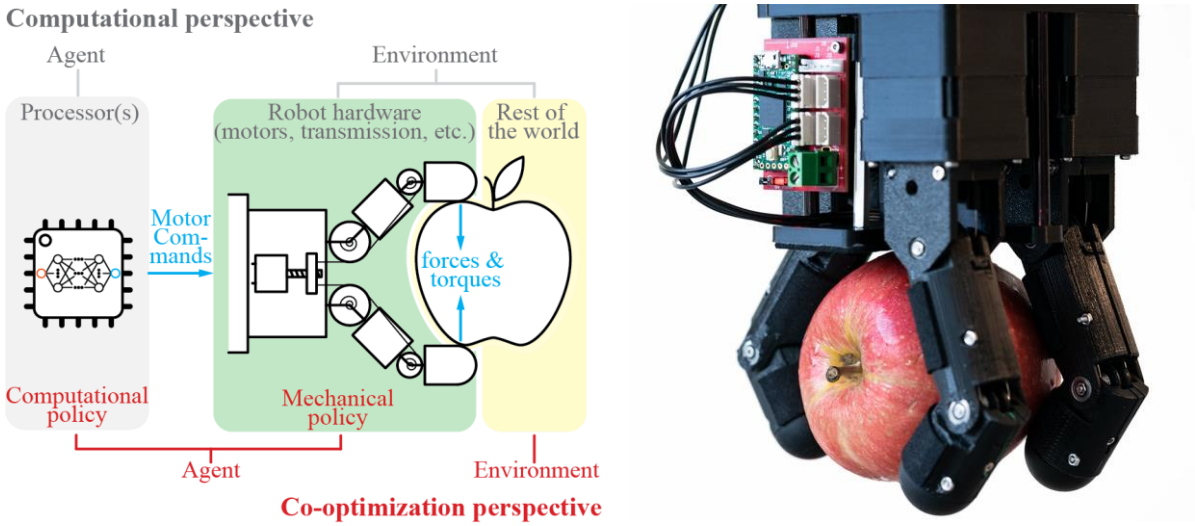

Fig. 1. Hardware as policy. (Left) We contrast the co-optimization perspective, where part of the robot's hardware is considered as a mechanical policy and optimized alongside the computational policy, against the computational-only optimization perspective, where the robot's hardware is typically considered part of the environment. **(Right)** A robot hand with a compliant underactuated transmission designed using the co-optimization paradigm

DOI 10.1126/scirobotics.abg2133

+These authors contributed equally to this work.

New computational results have only strengthened the evidence favoring co-design, by showing that end-to-end optimization often outperforms sequential alternatives. We now know that filters for low-level feature detection in images work best when optimized end-to-end based on performance on the desired task, which could range from image classification to autonomous game play [2]. Through this lens, mechanical / computational co-design in robotics is simply end-to-end optimization taken to its logical conclusion. Aligned with such renewed incentives, we bring attention to three developments, none of them motivated primarily by co-optimization, but all bearing notable promise in this direction:

1. Continued progress in rigid- and soft-body dynamics simulators. Co-optimization in robotics can rarely afford to wait for the physical prototyping of each hardware iteration, thus simulation has always been central to the effort. Over the past five years, multi-purpose physics engines (such as MuJoCo or Bullet) able to handle complex kinematic chains and transient frictional contact, have finally become general and robust enough to be considered commodities. Concurrently, general access to cloud computing has lowered the computational cost barrier to widespread use of large-scale simulation.

2. Sim-to-real transfer via domain randomization. Building accurate simulators is still an active and compelling research problem. However, domain randomization has emerged as a complementary approach: by introducing variation in the behavior of an imperfect simulator, we can obtain a policy or controller that is general enough to discount simulation inaccuracies and thus transfer to the real world. Domain randomization inherently requires optimization algorithms that can formally reason about the large ensuing variance in the behavior of the agent, which is particularly difficult because the simulated agent itself can rarely be given access to observe the parameters of the simulation.

3. Advances in policy optimization for complex motor skills. The last half decade has seen impressive advances in the optimization of the software side of a motor ability. Although examples also include new results in evolutionary algorithms or optimal control, we focus here specifically on policies optimized through deep reinforcement learning, using models and stochastic algorithms that are inherently able to consider different outcomes from similar observations (due to partially observable states). As a result, these algorithms are a natural match for domain randomization, allowing the physical realization of complex motor skills trained in simulation.

These developments all point to new opportunities for mechanical/computational co-optimization. We have better simulation tools, better methods for transferring simulation results to real robots, and better methods to derive controllers or policies for complex motor skills. Indeed, recent years have seen many promising new results. Reinforcement Learning algorithms have optimized both design parameters and policies for simulated robots performing difficult motor skills [7,8,9]. Bayesian Optimization has also been used to similar effect [6]. Looking beyond simulation, studies are also explicitly considering the fact that a number of real physical hardware iterations might still be expected when doing simulated co-optimization [6,9].

In our own recent work [10], we notice that *hardware components can be deeply similar to policies*. A computational policy takes as input sensor observations, processes them (e.g. through a deep neural network) and outputs actions as motor commands. Hardware takes the problem from here: it accepts motor commands as input and processes them through the robot's body to finally output actions such as forces acting on the world. The behavior of a computational policy is governed by tunable parameters (e.g. weights and biases), much like hardware behavior is determined by tunable design parameters.

We applied this concept to the problem of underactuated manipulation, where a computational policy can move the hand and provide setpoints for the motor controlling the fingers. The hand transmission mechanism then acts as a mechanical policy, converting motor forces to joint torques. We combine models of the computational and mechanical policies into a joint policy, which we optimize via off-the-shelf policy gradient algorithms. During training, we simulate the rest of the world (beyond the transmission mechanism) in an off-the-shelf dynamics simulator, with extensive domain randomization. Once training is complete, the optimized parameters of the mechanical policy are used to build physical prototypes, which are in turn controlled by the computational policy to operate in the real world. Importantly, simultaneous optimization of hardware and software is shown to outperform approaches that alternate between the two.

The results of the studies we cited above highlight the fact that current optimization methods originally developed for the computational engines of intelligent systems can also be applied, in the case of embodied agents, to hardware. They suggest that, for complex motor skills such as manipulation, intelligence is not confined to computation: it can reside in actuators, sensors or other hardware components just as well. Mechanical and computational co-design can take advantage of such methods to jointly optimize hardware and software, aiming for intelligent behavior by physical agents in complex environments.

References

1. R. W. Young. Evolution of the human hand: the role of throwing and clubbing. Journal of Anatomy, 202(1):165–174, 2003
2. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013
3. J.-H. Park and H. Asada. Concurrent design optimization of mechanical structure and control for high-speed robots. Journal of dynamic systems, measurement, and control, 116(3):344–356, 1994
4. H. Lipson and J. B. Pollack. Automatic design and manufacture of robotic lifeforms. Nature, 406(6799):974, 2000
5. K. Sims. Evolving virtual creatures. In ACM Conference on Computer Graphics and Interactive Techniques, 1994
6. T. Liao, G. Wang, B. Yang, R. Lee, K. Pister, S. Levine, and R. Calandra. Data-efficient learning of morphology and controller for a microrobot. In IEEE International Conference on Robotics and Automation (ICRA), pages 2488–2494, 2019.
7. D. Ha. Reinforcement learning for improving agent design. arXiv preprint arXiv:1810.03779, 2018
8. C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter. Jointly learning to construct and control agents using deep reinforcement learning. In IEEE International Conference on Robotics and Automation (ICRA), pages 9798–9805, 2019
9. K. S. Luck, H. Ben Amor, and R. Calandra. Data-efficient co-adaptation of morphology and behavior with deep reinforcement learning. In Conference on Robot Learning (CoRL), 2019
10. T. Chen, Z. He and M. Ciocarlie. Hardware as Policy: mechanical and computational co-optimization using deep reinforcement learning, In Conference on Robot Learning (CoRL), 2020