N-LIMB: Neural Limb Optimization for Efficient Morphological Design

Charles Schaff
Toyota Technological Institute at Chicago
Chicago, IL 60637 USA
cbschaff@ttic.edu

Matthew R. Walter
Toyota Technological Institute at Chicago
Chicago, IL 60637 USA
mwalter@ttic.edu

Abstract: A robot's ability to complete a task is heavily dependent on its physical design. However, identifying an optimal physical design and its corresponding control policy is inherently challenging. The freedom to choose the number of links, their type, and how they are connected results in a combinatorial design space, and the evaluation of any design in that space requires deriving its optimal controller. In this work, we present N-LIMB, an efficient approach to optimizing the design and control of a robot over large sets of morphologies. Central to our framework is a universal, design-conditioned control policy capable of controlling a diverse sets of designs. This policy greatly improves the sample efficiency of our approach by allowing the transfer of experience across designs and reducing the cost to evaluate new designs. We train this policy to maximize expected return over a distribution of designs, which is simultaneously updated towards higher performing designs under the universal policy. In this way, our approach converges towards a design distribution peaked around high-performing designs and a controller that is effectively fine-tuned for those designs. We demonstrate the potential of our approach on a series of locomotion tasks across varying terrains and show the discovery novel and high-performing design-control pairs.

Keywords: Co-optimization, Reinforcement Learning

1 Introduction

In this work, we study the efficient and automatic co-optimization of a robot's physical structure as well as the corresponding controller. The morphological design of a robot is inherently coupled with its control policy. Traditional approaches either reason separately over design and control or perform design-control optimization by iteratively updating and fabricating the current set of candidate designs, optimizing their corresponding controllers, and evaluating the performance of these design-control pairs. Data-driven approaches capable of efficient co-optimization provide a promising alternative to what is currently a time-consuming process.

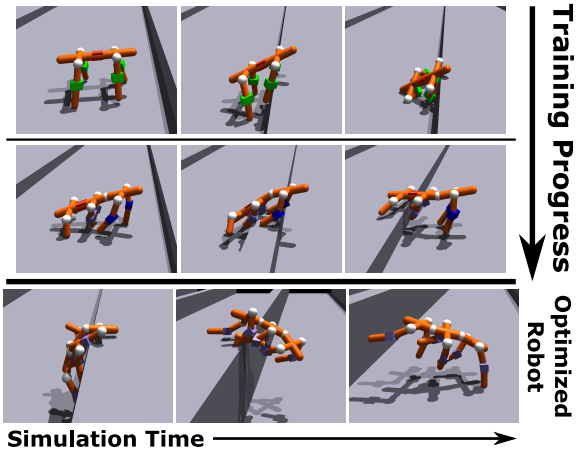


Figure 1: Our approach produces optimized robots and controllers for a given task by searching over large sets of morphologies. This figure shows the optimization of a wall-climbing robot from early in the training process (top row) to the final design (bottom row).

However, the search over discrete morphologies and their controllers is inherently challenging. The optimization process is

two-fold, requiring: 1) a search over a prohibitively large, discrete, and combinatorial space of morphologies, and 2) the evaluation of each examined morphology by solving for its optimal controller. The second step is particularly non-trivial and computationally expensive, especially for learned controllers, which makes it challenging to examine a large set of morphologies.

We present Neural Limb Optimization (N-LIMB), an efficient algorithm for computing optimal design-control pairs across a large space of morphologies (Fig. 1). N-LIMB is a non-trivial extension to the work of Schaff et al. [1], which provides a model-free approach to co-optimization over continuous design parameters, together with a learned controller *for robots with a single, fixed morphology*. Their approach exploits the shared structure of each design to train a universal, design-conditioned controller that is able to generalize across the space of valid designs, thus avoiding the need to solve an optimal control problem for each design. This controller is trained in expectation over a distribution of designs, which is then subsequently updated towards higher-performing designs. Adapting these components (i.e., a universal controller and design distribution) to a large set of morphologies is challenging. In fact, developing universal controllers that generalize across morphologies has emerged as an important topic independent of morphological optimization [2, 3, 4].

Our N-LIMB architecture overcomes these challenges to jointly optimize robot design and control over a large space of morphologies through three key contributions. Our approach represents the space of morphologies with a context-free graph grammar. This has two benefits: it allows for the easy incorporation of fabrication constraints and inductive biases into the symbols and expansion rules of the grammar; and it provides a way to iteratively generate designs by sampling expansion rules. The latter allows us to define complex, multi-modal distributions over the space of morphologies via a novel autoregressive model that recursively applies expansion rules until a complete graph has been formed. We parameterize the universal controller with a morphologically-aware transformer architecture. We evaluate the effectiveness of N-LIMB on a series of challenging locomotion tasks, demonstrating the ability to learn complex, high-performing designs coupled with agile control policies. For videos and code, please visit our webite: https://sites.google.com/ttic.edu/nlimb.

2 Related Work

Co-optimization of robot design and control: There exists a large body of work that addresses the problem of optimizing the physical design of a robot along with the corresponding control policy. Initial research on co-optimization is concerned with identifying the discrete morphology of high-performing robots [5, 6, 7]. This morphology may take the form of a composition of fixed-size 3D blocks [5] or deformable voxels [8, 9, 10], or as a topology of rigid parts connected by fixed or actuated joints [11, 12, 13, 14, 15, 16, 17, 18], which is the approach that we take here.

A common approach to optimizing over discrete designs is to perform evolutionary search (ES) over a population of candidate designs [5, 6, 19, 20, 21, 7, 22, 23, 24, 25, 8, 26, 9, 27]. However, ES is prone to local minima and is sample-inefficient, in part due to the need to maintain separate control policies for each candidate design in the population. Our approach improves upon the sample-efficiency of these methods sharing experience across designs through a single control policy. Alternatively, several approaches improve sample efficiency by leveraging additional knowledge or assumptions about the system dynamics, typically in the context of optimizing the continuous parameters of a fixed morphology. These approaches include trajectory optimization [28, 29], linear approximations to the dynamics [30], and leveraging differentiable simulators [31].

Similar to our approach, several reinforcement learning-based strategies to co-optimization exist for both continuous [1, 32, 33, 34] and discrete design spaces [10, 35, 14]. Luck et al. [33] use a soft actor-critic algorithm and use a design-conditioned Q-function to evaluate designs. Chen et al. [34] model the design space as a differentiable computational graph, which allows them to use standard gradient-based methods. In the context of discrete design spaces, Spielberg et al. [10] propose an autoencoder-based method that jointly optimizes the placement and control of a large number deformable voxels for soft-body locomotion tasks. Yuan et al. [36] represent the design generation process as part of the environment and train a dual-purpose design generation and control policy.

Universal control policies: Traditional population-based co-optimization strategies are difficult to scale due to the need to maintain separate controllers for each design within the population as it changes over time. Recent approaches adopt a single design-aware control policy to improve optimization efficiency. Schaff et al. [1] propose such an approach for optimization within fixed morphology. More sophisticated controllers leverage graph neural networks (GNN) [37] structured according to the robot morphology [38, 14, 15, 2, 36]. Recently, shared controllers based on self-attention mechanisms [3, 4] have outperformed GNNs due to the ability to propagate information across nodes more effectively than the message-passing schemes of GNNs. Motivated by the empirical success of self-attention over GNNs, we leverage transformers [39] to model our controller.

3 Co-optimization of Design and Control

In this section, we introduce the problem of jointly optimizing the physical design and corresponding control of a robot in the context of a specified task.

3.1 Problem Definition

We formulate the problem of co-optimizing design and control as a set of related reinforcement learning problems. Given a set of robot designs Ω and a task definition, we define a *design-specific* Markov decision process (MDP) for each $\omega \in \Omega$: $\mathcal{M}_\omega = \text{MDP}(\mathcal{S}_\omega, \mathcal{A}_\omega, \mathcal{P}_\omega, \mathcal{R}_\omega)$, where $\mathcal{S}_\omega$ is the state space, $\mathcal{A}_\omega$ is the action space, $\mathcal{P}_\omega : \mathcal{S}_\omega \times \mathcal{A}_\omega \times \mathcal{S}_\omega \to [0, 1]$ is the transition dynamics, and $\mathcal{R}_\omega : \mathcal{S}_\omega \times \mathcal{A}_\omega \times \mathcal{S}_\omega \to \mathbb{R}$ is the reward function. When optimizing for a single task objective, it is natural that these MPDs share some common structure, e.g., the state space only changes based on the proprioceptive information of available to each design, the action spaces differ only in the number of controllable degrees of freedom, and each $\mathcal{R}_\omega$ encodes the same task objective.

Let $\pi^*_\omega : \mathcal{S}_\omega \times \mathcal{A}_\omega \to [0, 1]$ be the optimal policy for MDP $\mathcal{M}_\omega$ and $\mathbb{E}_{\pi^*_\omega}\left[\sum_t \gamma^t r_t\right]$ be its expected return. The goal of co-optimization is to find the optimal design-controller pair $(\omega^*, \pi^*_{\omega^*})$ such that:

$$\omega^*, \pi^*_{\omega^*} = \arg\max_{\omega, \pi^*_\omega} \mathbb{E}_{\pi^*_\omega}\left[\sum_t \gamma^t r_t\right]. \tag{1}$$

3.2 Co-optimization via Universal and Transferable Policies

Based on Equation 1, the search over the design space Ω requires access to the optimal controller π^*_ω for each design. Obtaining these controllers can be challenging, as it is often computationally intractable to train independent controllers for each design. However, we can leverage the shared structure between each design to efficiently train a *design-conditioned* controller: $\pi : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$. Ideally, this controller would serve as a proxy for optimal controllers as well as generalize zero-shot to unseen designs in the design space. Implementing such a controller has been studied in the context of co-optimization [1, 36, 40] as well as the transfer of control policies across morphologies [2, 3, 4].

Algorithm 1: Joint Optimization of Design and Control

```
Initialize π_θ(a|s, ω), p_φ(ω), T = 0, T_0
while T < BUDGET do
    Sample designs ω_1, ω_2, ..., ω_n ~ p_φ
    Control ω_1, ω_2, ..., ω_n with π_θ for t timesteps
    Update θ using PPO with collected trajectories
    Set timestep T = T + nt
    if T > T_0 then
        Compute average episode returns R_{ω_1}, R_{ω_2}, ..., R_{ω_n}
        Update φ using ∇_φ ≈ Σ_{i=0}^{n} ∇log p_φ(ω_i) R_{ω_i}
    end if
end while
```

With such a control policy, the co-optimization procedure reduces to search over the design space Ω. In this work, we use a zero-order search algorithm based on policy gradients that optimizes a distribution over the design space towards higher performing designs. Let p_ϕ be a distribution over Ω parameterized by ϕ and π_θ be a design-conditioned policy. Then, our training objective is to

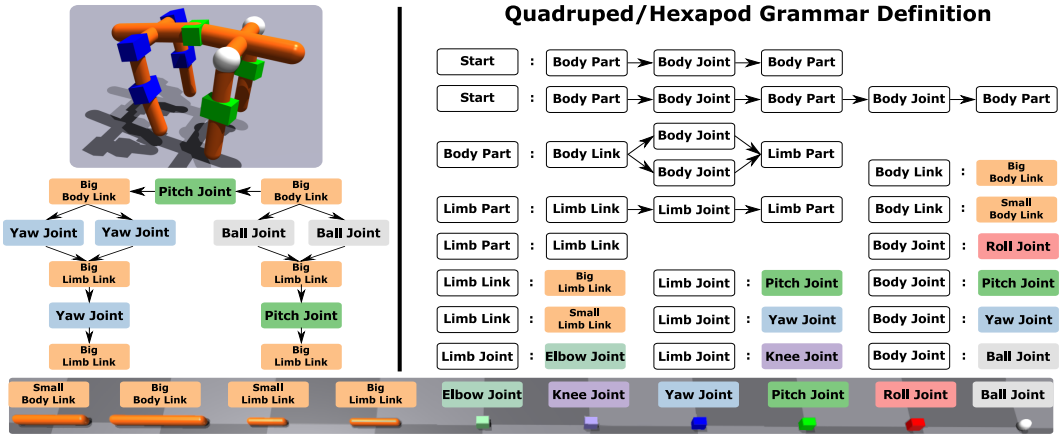


Figure 2: Our approach defines the space of valid designs using a context-free graph grammar in which terminal symbols denote robot parts and expansion rules describe how those parts can be combined. **Left**: An example design and its corresponding graph structure. **Right**: The definition of a bio-inspired grammar that produces quadruped and hexapod designs. Nonterminal symbols are denoted with white boxes, terminal symbols with colored boxes, and expansion rules are denoted with a colon. **Bottom**: Terminal symbols and their corresponding parts, where joints are colored according to their degrees-of-freedom.

maximize the expected return of π_θ under the design distribution p_ϕ:

$$\phi^*, \theta^* = \arg\max_{\phi,\theta} \mathbb{E}_{\omega\sim p_\phi} \mathbb{E}_{\pi_\theta} \left[\sum_t \gamma^t r_t \right]. \tag{2}$$

The controller can then be trained using any standard RL algorithm and the design distribution can be updated with any zero-order method. Similar to past work [1], we use policy gradients to update both the policy and the design distribution. This leads to the following update equations:

$$\nabla_\phi = \mathbb{E}_{\omega\sim p_\phi} \left[\nabla_\phi \log p_\phi(\omega) \mathbb{E}_{\pi_\theta} \left[\mathcal{R}_t \right] \right] \qquad \nabla_\theta = \mathbb{E}_{\omega\sim p_\phi} \left[\mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s_t, a_t) \mathcal{R}_t \right] \right], \tag{3}$$

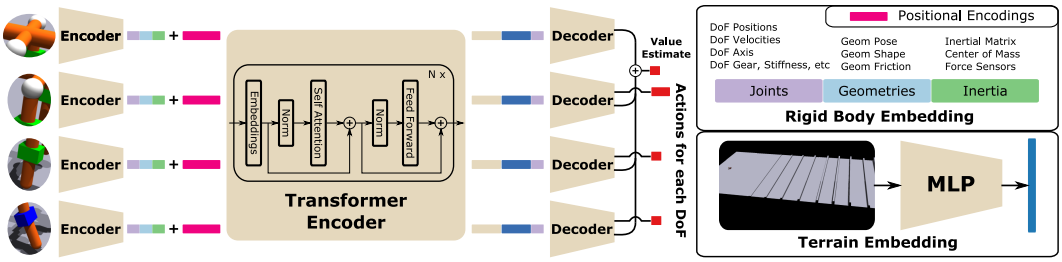
where $\mathcal{R}$ is the expected return at time t. This algorithm trains a controller to maximize performance in expectation over the design distribution p_ϕ. When this controller is sufficiently trained, it is used as a proxy for the optimal controllers of each design and p_ϕ is updated towards higher performing designs. Then training continues until p_ϕ converges on a single design and π_θ is optimized for that design. See Algorithm 1 for details.

4 N-LIMB: Optimizing Design and Control over Morphologies

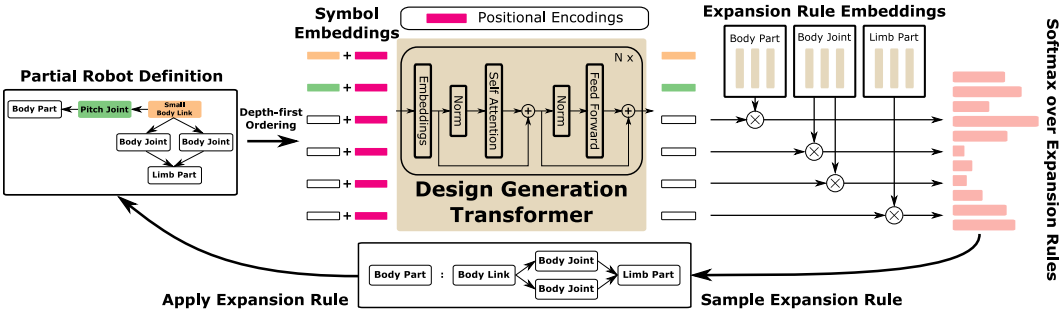
While Schaff et al. [1] successfully employ the above approach to find optimal design-control pairs, their approach is limited to simple, fixed (i.e., pre-defined) kinematic structures and is not able to optimize across morphologies. We propose a framework that builds on this approach to perform morphological optimization. We describe a general approach to defining combinatorial spaces of robot designs, a policy architecture that is able to control and generalize across that space, and a way to parameterize and optimize complex distributions over the design space.

4.1 Graph Grammars for Defining Design Spaces across Morphologies

In this work, we explore the use of context-free graph grammars to define the space of valid morphologies. We define a context-free graph grammar as the tuple (N, T, R, S), where N denotes the set of non-terminal symbols, T denotes the set of terminal symbols, $R = \{r_s^i = (s, G_s^i)\}$ is the set of production rules that map a non-terminal symbol $s \in N$ to a graph G_s^i, and S is the starting graph. The application of a production rule replaces a node with non-terminal symbol s with the



(a) Actor-critic architecture



(b) Autoregressive model for design generation

Figure 3: **(a)** Self-attention based actor-critic architecture. Rigid body pose and attributes are encoded and processed via a transformer encoder to produce actions for each DoF and a value estimate. **(b)** The design distribution is modeled via an autoregressive transformer architecture. Partial graphs are processed by the model to produce a distribution over expansion rules of the grammar. Rules are sampled and applied until no non-terminal symbols remain within the graph.

corresponding graph G_s^i. Nodes within these graphs express limb information (e.g., geometry, pose, mass, inertia, etc.), and edges contain joint information (e.g., type, the kinematic transformation between limbs, max torque, gear ratio, etc.).

With the end goal being to build and deploy optimized design-control pairs, fabrication constraints must be taken into account, including what parts can be used and in which ways can they be feasibly combined. Context-free grammars can naturally capture these constraints, with terminal symbols that correspond to available parts and expansion rules that represent limits on how those parts can be combined. In this way, the grammar focuses the optimization only on feasible designs.

In our experiments, we use a simple grammar (Fig. 2) to define the space of quadruped and hexapod robots. This grammar generates robots as a composition of simple shapes, maintains left-right symmetry, and allows for a variety of joint types and limbs with varying numbers of parts and sizes.

4.2 Universal Controllers via Morphologically-aware Transformers

Our co-optimization algorithm requires a universal controller that can act as a proxy for the optimal controller across the design space Ω. Control policies that employ graph neural networks [38, 2] as well as those that use self-attention [3, 4] have proven effective at handling the varying number of rigid-bodies and controllable degrees-of-freedom across morphologies. In this work, we use the transformer architecture [39] to parameterize an actor-critic network as it was shown to outperform graph neural networks [3]. For an overview of our actor-critic architecture, see Figure 3(a).

At each point in time, the agent receives various types of state information: local information about each rigid body and joint (e.g., pose, velocity, etc.), morphological information (e.g., shape, inertial, joint axes, etc.), and global sensory information (e.g., cameras, terrain observations, task-relevant observations, etc.). Our architecture consists of three core components: the first module encodes

local and morphological information; the second processes that information through self-attention and then appends global information; and the third module decodes actions and a value estimate.

Rigid-body embeddings: We encode each robot by flattening its graph structure into a sequence of rigid-body embeddings using a depth-first traversal. Each of these embeddings contains local pose and morphological information about the rigid body, as well as each controllable degree-of-freedom that connects it to its parent. Information about geometries, degrees-of-freedom, and inertia are encoded separately using two-layer MLPs and concatenated to obtain the rigid-body embedding.

Process: The rigid-body embeddings are combined with positional encodings and processed with a transformer encoder. Similar to Gupta et al. [4], we postpone the fusion of global information, such as observations about the surrounding terrain, until after the self-attention layers to both reduce the number of transformer parameters as well as avoid diluting the low-dimensional pose and morphological information. Terrain observations are comprised of sparsely sampled height-field information that is encoded into a flat embedding vector using a three-layer MLP and concatenated with each processed rigid-body embedding.

Decoding actions and value estimates: After the addition of global sensory information, we produce a value estimate by averaging the outputs of a value-decoder MLP across each rigid-body. For each degree-of-freedom associated with a rigid body, we concatenate an embedding of that DoF before applying an action-decoder MLP that produces the mean and state-independent standard deviation of a Gaussian distribution.

4.3 Autoregressive Models for Design Generation

Given the definition of a graph grammar, designs can be generated by autoregressively sampling expansion rules until the resulting graph contains only terminal symbols. To build such a model, we again use the transformer architecture [39]. Sampling designs in this way allows, in principle, for arbitrary distributions over the design space that can capture multi-modal behavior and complex dependencies between symbols. For an overview of our model, see Figure 3(b).

At each stage of generation, the model receives as input a partial graph G. This graph is then flattened to a sequence of symbols $s_1, \ldots, s_n$ using a depth-first traversal. Those symbols are then embedded using an embedding table, combined with a positional encoding, and processed with a transformer encoder to produce representations $h_{s_1}, \ldots, h_{s_n}$. We decode logits for each expansion rule by computing a dot-product between each representation h_{s_i} and learnable embeddings corresponding to each expansion rule $r^j_{s_i}$ associated with symbol s_i. These logits are combined across all symbols in G using a softmax operation. An expansion rule is then sampled from this distribution, used to update the graph G, and then this process is repeated until G contains only terminal symbols.

5 Results

We evaluate our N-LIMB framework by performing co-optimization of design and control for locomotion tasks involving a variety of terrains. In all cases, N-LIMB efficiently finds high-performing design-control pairs that outperform baselines.

Experiment Details: We optimize and evaluate design-control pairs using the IsaacGym GPU-based physics simulator [41], allowing us to train with thousands of parallel environments on a single GPU. The locomotion task rewards each robot for making as much forward progress as possible within a fixed amount of time, with penalties for energy consumption, large actuation torques, and reaching joint limits. Episodes are terminated early if the robot has fallen over. We consider three different terrain types: (i) flat terrain ("Flat"); (ii) a terrain with randomly placed gaps of random widths that the robot must cross ("Gaps"); and (iii) a terrain with randomly placed walls of different heights that the robot must climb over ("Walls"). We run the N-LIMB algorithm on a single NVIDIA A4000 GPU for 1B timesteps (roughly two days). We use Proximal Policy Optimization [42] to train both the controller and design distribution. For more details and hyperparameters, see Appendix A.3.

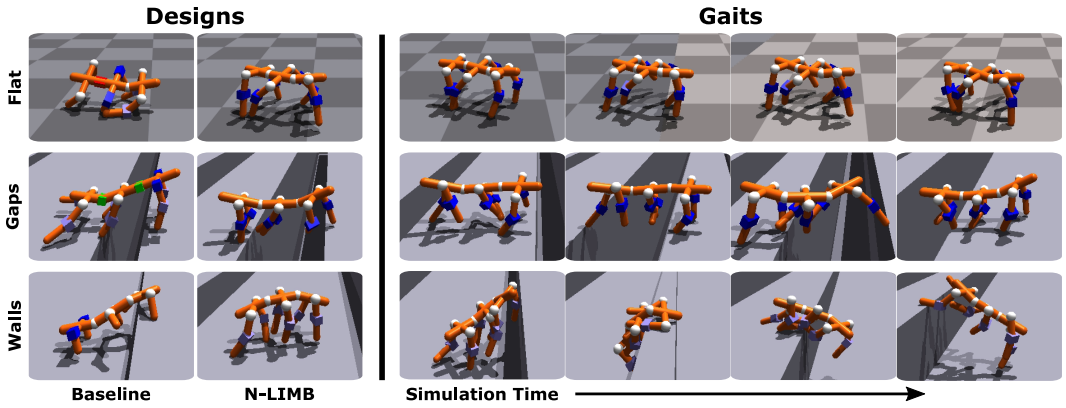


Figure 4: **Left:** A comparison between the designs learned by N-LIMB and those of the random search baseline for the three terrain types. **Right:** A visualization of the learned gaits.

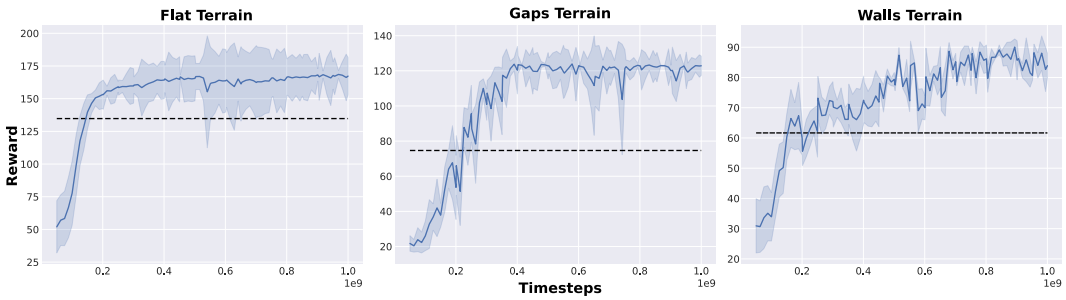


Figure 5: Plots that show the reward (mean and standard deviation) of designs sampled from the design distribution over the course of training for the three terrain types. Dashed horizontal lines denote the rewards of the highest performing design-control pairs for the baseline algorithm.

Baselines: Similar to previous work in co-optimization [1, 16, 17], we compare our framework against a decoupled approach that randomly samples designs from the hexapod grammar and trains controllers for each design individually. To ensure a fair comparison with our approach, we provide the baseline with the same computational budget as N-LIMB.

Results: Across all three terrains, we find that N-LIMB is able to outperform the random search baseline by a large margin. Figure 4 shows the best designs found by N-LIMB and the baseline, as well as the learned gaits while traversing difficult sections of each of the three terrain types. While N-LIMB initially favors quadruped designs in some cases (Fig. 1), it converges to hexapod designs for all three terrain types. N-LIMB chooses to connect the body links with ball joints, suggesting that additional degrees of freedom outweighs a reduction in available torque and narrower joint limits. The choice of a hexapod design may be a result of the fact that achieving longer bodies with our grammar requires a design with six legs. Indeed, on the Gaps terrain (Fig. 4, middle row), N-LIMB selects a design that has the largest body available within the grammar through the use of larger body links, which is necessary to cross wider gaps. The baseline similarly identifies a long-bodied hexapod, but it connects the body links with fewer degrees of freedom, reducing its flexibility. On the Walls terrain, however, the optimal design (Fig. 4, bottom row) has a smaller body and connects limb links with "knee" joints, which can fold back $180°$, allowing the robot to fold its limbs while cresting or pushing off the wall. In contrast, the optimal baseline design has legs with a single, small link, which limits the height of the walls that the baseline can climb over. For locomotion on the Flat terrain, N-LIMB converges to a design identical to that of the Gaps terrain, except that the torso is comprised of three small (vs. large) body links, trading off the need to span wide gaps for the availability of lower-torque bounding gates. The baseline identifies a hexapod with single-link legs and replaces some of the ball joints with pitch and roll joints.

Figure 5 shows the quantitative performance of N-LIMB compared against the baseline. It displays the mean and standard deviation of returns obtained by designs sampled from the design distribution throughout training. Early on, the ability for the policy to control the set of morphologies drawn from the design distribution is limited. However, as the controller improves and N-LIMB updates the design distribution, we see that the algorithm yields designs paired with the universal control policy that quickly outperform the baseline.

Generalization of the Universal Controller: Crucial to our approach is a universal controller that provides a proxy for the optimal controller when evaluating a design. If this controller is unable to provide a reasonable estimate of a design's optimal performance, the design distribution may incorrectly focus the search around suboptimal designs. To verify that our controller provides a reasonable proxy, we compare it against controllers trained individually on each design sampled from the initial design distribution. Figure 6 shows the performance of the universal controller as a fraction of the reward of the individually trained controllers at four checkpoints during training. We keep the design distribution fixed for the first 100M steps of training, during which the universal controller impressively obtains up to 80% of the performance of individually trained controllers. Once we begin updating the design distribution (after 100M timesteps), we see a performance expectedly declines as the controller specializes to the shifting design distribution.

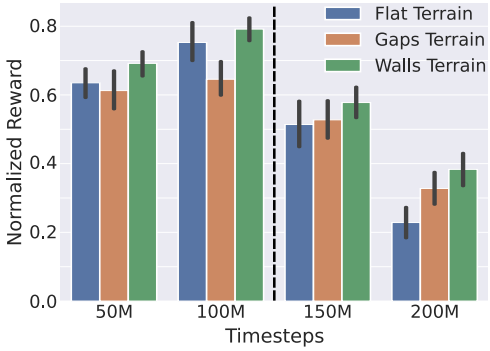


Figure 6: A visualization of the generalizability of the universal N-LIMB controller as a fraction of the reward achieved using controllers trained separately on designs sampled from the initial design distribution. In all three domains, the relative reward improves until, after 100M timesteps (dashed line), we begin to update the design distribution, upon which the N-LIMB policy learns to specialize to higher-performing designs.

6 Limitations

Among the limitations of this work, we only show results in simulated settings. While we designed our grammar to preclude difficult-to-fabricate designs, we do not demonstrate real-world transfer of the learned design-controller pairs. Inspired by the success of sim-to-real transfer for control [43, 44] and co-optimization [35], future work will investigate the real-world performance of the resulting designs. Further, we place bio-inspired constraints on our grammar to avoid unreasonable designs and, in turn, to improve the efficiency of optimization. These rules may be overly conservative, preventing our method from learning designs that while being atypical, are well-suited to the task.

7 Conclusion

We presented N-LIMB, an efficient and effective approach to co-optimizing robots and their controllers across large, combinatorial sets of morphologies. N-LIMB formulates the set of valid morphologies as a context-free graph grammar, allowing users to easily incorporate fabrication constraints and inductive biases, focusing search on realizable robots. Given such a grammar, we parameterize a distribution over the design space using a novel autoregressive model that recursively samples the expansion rules of the grammar ntil a completed robot is formed. The optimization is carried out by training a universal controller in expectation over the design distribution, while simultaneously shifting that distribution towards higher performing designs. In this way, the optimization process converges to a design-control pair that is jointly optimal for the given task. We demonstrate the potential of our approach by learning high-performing design-control pairs on a variety of locomotion tasks and terrains.

References

[1] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter. Jointly learning to construct and control agents using deep reinforcement learning. *arXiv preprint arXiv:1801.01432*, 2018.

[2] W. Huang, I. Mordatch, and D. Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. *arXiv:2007.04976*, 2020.

[3] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson. My body is a cage: The role of morphology in graph-based incompatible control. *arXiv:2010.01856*, 2021.

[4] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei. MetaMorph: Learning universal controllers with transformers. *arXiv preprint arXiv:2203.11931*, 2022.

[5] K. Sims. Evolving virtual creatures. In *Proceeding of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1994.

[6] H. Lipson and J. B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.

[7] S. Murata and H. Kurokawa. Self-reconfigurable robots. *IEEE Robotics & Automation Magazine*, 14(1):71–78, 2007.

[8] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1): 11–23, 2014.

[9] N. Cheney, J. Bongard, V. SunSpiral, and H. Lipson. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15(143):20170937, 2018.

[10] A. Spielberg, A. Zhao, Y. Hu, T. Du, W. Matusik, and D. Rus. Learning-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[11] G. S. Hornby, H. Lipson, and J. B. Pollack. Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 19(4): 703–719, 2003.

[12] R. Desai, B. Li, Y. Yuan, and S. Coros. Interactive co-design of form and function for legged robots using the adjoint method. *arXiv preprint arXiv:1801.00385*, 2018.

[13] S. Ha, S. Coros, A. Alspach, J. M. Bern, J. Kim, and K. Yamane. Computational design of robotic devices from high-level motion specifications. *Transactions on Robotics*, 34(5):1240–1251, 2018.

[14] D. Pathak, C. Lu, T. Darrell, P. Isola, and A. A. Efros. Learning to control self-assembling morphologies: A study of generalization via modularity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[15] T. Wang, Y. Zhou, S. Fidler, and J. Ba. Neural graph evolution: Towards efficient automatic robot design. *arXiv preprint arXiv:1906.05370*, 2019.

[16] A. Zhao, J. Xu, M. Konaković-Luković, J. Hughes, A. Spielberg, D. Rus, and W. Matusik. RoboGrammar: Graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6), 2020.

[17] D. J. Hejna III, P. Abbeel, and L. Pinto. Task-agnostic morphology evolution. *arXiv preprint arXiv:2102.13100*, 2021.

[18] J. Xu, A. Spielberg, A. Zhao, D. Rus, and W. Matusik. Multi-objective graph heuristic search for terrestrial robot design. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9863–9869, 2021.

[19] C. Paul and J. C. Bongard. The road less travelled: Morphology in the optimization of biped robot locomotion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[20] R. A. Watson, S. G. Ficici, and J. B. Pollack. Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18, 2002.

[21] R. Pfeifer and J. Bongard. *How the body shapes the way we think: A new view of intelligence*. MIT press, 2006.

[22] K. Wampler and Z. Popović. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)*, 28(3):1–8, 2009.

[23] J. Hiller and H. Lipson. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466, 2011.

[24] J. Bongard. Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 108(4):1234–1239, 2011.

[25] J. C. Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8):74–83, 2013.

[26] L. Brodbeck, S. Hauser, and F. Iida. Morphological evolution of physical robots through model-free phenotype development. *PLOS One*, 10(6), 2015.

[27] X. Pan, A. Garg, A. Anandkumar, and Y. Zhu. Emergent hand morphology and control from optimizing robust grasps of diverse objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7540–7547, 2021.

[28] A. Spielberg, B. Araki, C. Sung, R. Tedrake, and D. Rus. Functional co-optimization of articulated robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5042, 2017.

[29] G. Bravo-Palacios, A. Del Prete, and P. M. Wensing. One robot for many tasks: Versatile co-design through stochastic programming. *IEEE Robotics and Automation Letters*, 5(2):1680–1687, 2020.

[30] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane. Joint optimization of robot design and motion parameters using the implicit function theorem. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[31] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal. An end-to-end differentiable framework for contact-aware robot design. *arXiv preprint arXiv:2107.07501*, 2021.

[32] D. Ha. Reinforcement learning for improving agent design. *arXiv preprint arXiv:1810.03779*, 2018.

[33] K. S. Luck, H. B. Amor, and R. Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. *arXiv preprint arXiv:1911.06832*, 2019.

[34] T. Chen, Z. He, and M. Ciocarlie. Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning. *arXiv preprint arXiv:2008.04460*, 2020.

[35] C. Schaff, A. Sedal, and M. R. Walter. Soft robots learn to crawl: Jointly optimizing design and control with sim-to-real transfer. *arXiv preprint arXiv:2202.04575*, 2022.

[36] Y. Yuan, Y. Song, Z. Luo, W. Sun, and K. Kitani. Transform2Act: Learning a transform-and-control policy for efficient agent design. *arXiv preprint arXiv:2110.03659*, 2021.

[37] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[38] T. Wang, R. Liao, J. Ba, and S. Fidler. NerveNet: Learning structured policy with graph neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[40] T. Chen, Z. He, and M. Ciocarlie. Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning. *arXiv:2008.04460*, 2020.

[41] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[44] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020.

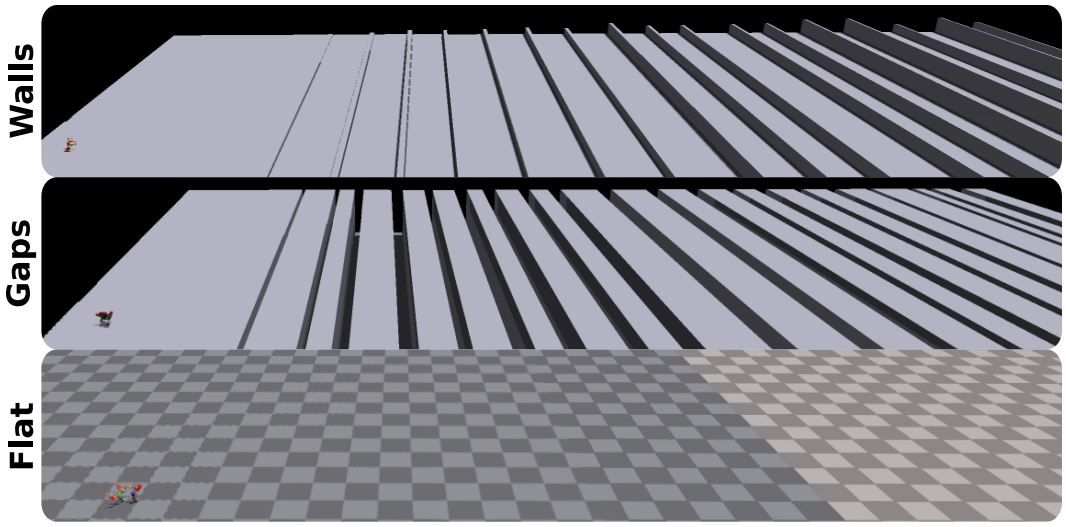


Figure 7: Images depicting the three terrains used in our experiments. In the "Walls" and "Gaps" environments, the walls (gaps) are randomly spaced with heights (widths) that increase farther from the agent's initial pose.

A Implementation Details

A.1 Terrain Details

We test our approach on the three terrain types pictured in Figure 7. The "Walls" terrain places barriers that the robot must climb over. These barriers are randomly spaced throughout the terrain and are assigned a random height. To enable some form of curriculum learning, we limited the height of the barriers in the vicinity of the agents initial pose. The "Gaps" terrain similarly places gaps at random locations with random widths that we limited in the same way as initial wall heights. We randomly generated the terrains each time the algorithm sampled new designs.

A.2 Network Details

Our actor-critic network consists of a transformer encoder with one layer of self-attention, a model dimension of 256, and 4 attention heads. The input to the encoder is a sequence of rigid body embeddings created from a concatenation of geometric, inertial, and joint embeddings. We use a two-layer MLP to encode information about each geometry and subsequently added these geometric encodings together. Similarly, we use a two-layer MLP to encode information about each degree-of-freedom and then add them together. The network also uses a two-layer MLP to encode inertial information. These three embeddings are then concatenated together. We order the rigid-body embeddings by flattening the robot's kinematic tree with a depth-first traversal, added with a learned positional encoding, and dropout is applied with $p = 0.1$. We separately encode terrain information with a three-layer MLP. After the transformer encoder, terrain information is concatenated with each token. We apply a three-layer value-decoder MLP to each token and then average the output to produce a value estimate. To produce actions, we concatenate degree-of-freedom information with each token and then pass the result to a three-layer action-decoder network, which produces the mean and (state-independent) standard deviation of a Gaussian distribution.

Our autoregressive design distribution also consists of a transformer encoder with one layer of self-attention, a model dimension of 256, and 4 attention heads. We order partial graphs using a depth-first traversal, and embed symbols for each node with an embedding table, that we then add with a positional encoding and feed to the transformer encoder. For non-terminal symbols, we decode the resulting representations with a two-layer MLP and compute the dot product with embeddings of the grammar's expansion rules to produce logits. We combine these logits across nodes, and create

a categorical distribution via softmax. Robot generation proceeds by sampling expansion rules and updating the graph until only terminal symbols remain.

A.3 Hyperparameters

The following table details the specific hyperparameter settings that we used for the experiments presented in the paper.

Environment Hyperparameter	Value
Number of environments	2048
Maximum episode length	1000
dt	$1/60$ sec
Termination height	0.08
Reward: forward progress weight	3.8
Reward: alive bonus	0.5
Reward: termination cost	2.0
Reward: energy cost scale	0.01
Reward: squared action cost scale	0.035
Reward: joints at limit cost scale	0.2
PPO Hyperparameter	**Value**
Discount factor (γ)	0.99
GAE λ	0.95
Rollout length	128
Batch size	16384
Epochs per rollout	4
Clip param	0.2
Policy loss coefficient	1
Value loss coefficient	3.7
Entropy bonus coefficient	0.0
Action bounds loss coefficient	100
Max gradient norm	1.0
KL target	0.04
Initial learning rate	0.0003
Optimizer	ADAMW($\beta_1 = 0.9, \beta_2 = 0.999$, weigth decay $= 0.018$)
N-LIMB Hyperparameter	**Value**
Maximum timesteps	1B
Policy warm-up period	60M
Timesteps per design	3000
Update period	12M
Batch size	2048
Epochs per update	2
Clip param	0.2
Policy loss coefficient	1
Entropy bonus coefficient	0.02
Max gradient norm	0.1
KL target	0.0126
Initial learning rate	0.001
Optimizer	ADAMW($\beta_1 = 0.9, \beta_2 = 0.999$, weight decay $= 0.018$)

Table 1: Environment, PPO, and N-LIMB Hyperparameters

B Visualization of the Design Space

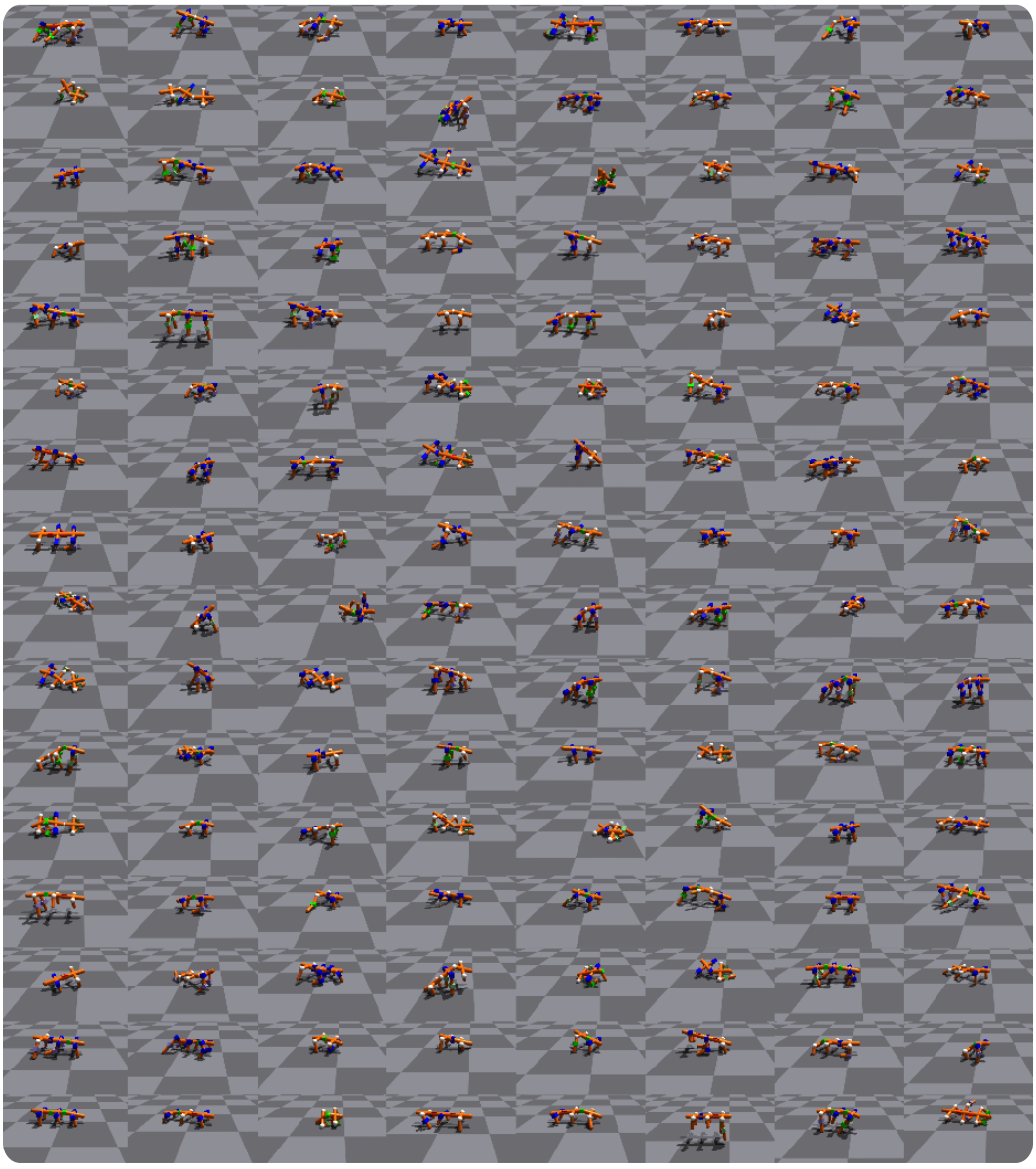


Figure 8: A visualization of randomly sampled designs from the hexapod grammar. The grammar contains millions of combinations of limb and joint parameters.