Zero-Shot Robot Design with Residual Physics

David Matthews and Sam Kriegman

Northwestern University
david.matthews@northwestern.edu

Abstract

The automatic design of robots usually occurs in simulation and remains challenging due to unmodeled dynamics that arise during deployment. These dynamics gaps are typically viewed as a nuisance to be accommodated by domain randomization or removed by filtering the search space. This prevents robots from acquiring morphological innovations that exploit the dynamics of their physical environment. Recently neural networks have been trained to capture the residual dynamics missing from a base simulator, but this approach has been limited to control policy optimization in a predesigned robot; optimization could only exploit a fixed, morphology-specific subset of possible dynamics. Here we show for the first time the evolution of a robot's body plan in a neural augmented simulator. This is a uniquely challenging problem as the body plan is optimized to exploit the full set of fitness-relevant dynamics—including those due to neural inaccuracies, which in effect rewards the exploitation of hallucinated dynamics. We show that an ensemble of residual models can identify and avoid these hallucinations, and that morphologies optimized in our augmented training environment recover the majority of the performance degradation induced by the missing dynamics. Although we study the dynamics gap between simulations (of slush), which provides a convenient and transparent testbed, our approach only relies on data that could in principle be efficiently collected in the real world. We also validate the simulations by building an evolved design. Future work will extend our approach to simulation-reality gaps.

Project Website: `https://residual-design.github.io`

Introduction

Animals succeed because they evolved designs that exploit, rather than ignore, the dynamics of their environment. The evolution of dynamics-exploiting robot designs is also possible but typically requires simulation. The problem with evolving robots in simulation is that the best designs are those that exploit dynamics like animals, but these are in most cases the least transferrable to reality.

Numerous approaches have been developed to mitigate the reality gap for control policy optimization. Early approaches focused on domain randomization (Jakobi et al., 1995) or by avoiding policies predicted to transfer poorly (Bongard et al., 2006; Koos et al., 2010; Cully et al., 2015). Recently the field has moved from accommodating unknown dynamics to exploiting them. For example, state estimation approaches (Kumar et al., 2021; Ji et al., 2022) have been used to encourage the policy to predict environmental properties and hidden robot states, such as friction, foot height, and linear velocity. This permits the policy to identify and adapt to changes in real time, however it does not enable adaptation to entirely unmodeled dynamics. Data-driven models such as neural dynamics models or residual physics models learn from data to directly model real world dynamics or the dynamics gap between simulation and reality. This approach has been applied to contact dynamics (Ajay et al., 2018; Heiden et al., 2021; Jing et al., 2026), robot actuators (Golemo et al., 2018; Hwangbo et al., 2019; Serifi et al., 2023; Sontakke et al., 2023; Gao et al., 2024), as well as aerodynamic effects (Zeng et al., 2020; Bauersfeld et al., 2021; O'Connell et al., 2022; Sontakke et al., 2023; Kaufmann et al., 2023), leading to champion level drone racing performance (Kaufmann et al., 2023), precise flight in gusty wind (O'Connell et al., 2022) as well as record breaking agility, speed, and efficiency in a commercial quadrupedal robot (Hwangbo et al., 2019).

Serifi et al. (2023) used neural residual physics models of both actuators and rigid bodies within a robot to improve its control. While their actuator model was portable across robots, a distinct neural model must be retrained from scratch for each design. Others optimized static structures using neural dynamics (Allen et al., 2022) or the stiffness of a soft finger (Yi et al., 2025); however, no prior work has optimized body shape or topology of a complete robot using neural dynamics.

The evolution of robots is a challenging optimization problem due to complex feedback loops among the robot's structure, actuation, and behavior. Compared to learning a control policy for a predesigned robot, evolving the robot's

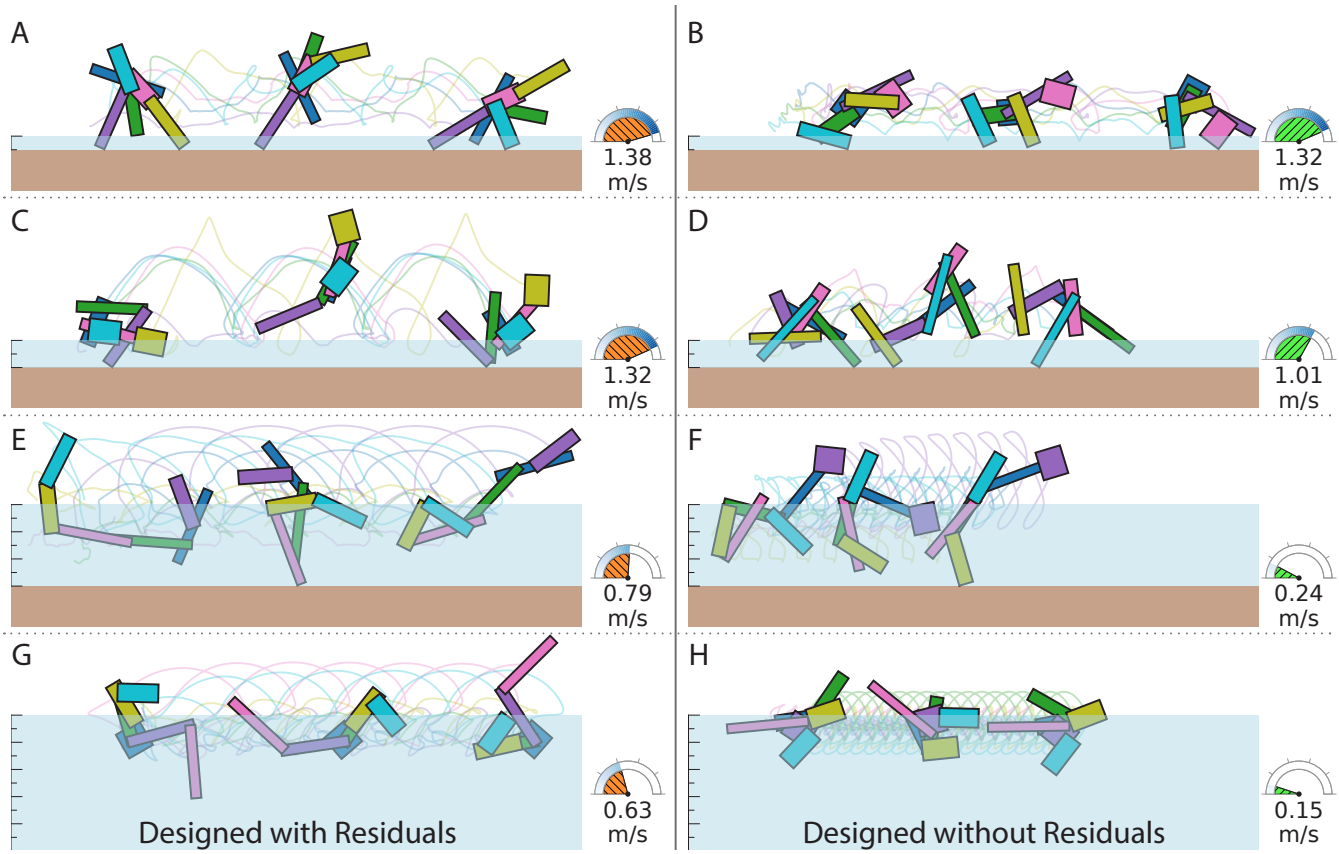

Figure 1: **Designs emerge in the neural residual augmented environment that increase fitness in the true slush environment.** In shallow slush (0.5m), robots that evolved with (**A**) and without (**B**) neural residuals both tend to exhibit legged locomotion at more or less the same speed. In slightly deeper slush (1m), unipedal body plans emerge in the neural augmented environment that leap out of the fluid like a dolphin to minimize fluid drag and maximize speed (**C**). In contrast, designs evolved without neural residuals continue to attempt legged locomotion through the slush (even after retraining in slush; **D**). In deeper slush (3m), the performance gap between designs evolved with and without residuals increases considerably (**E,F**). Designs that evolved with residuals usually maintain intermittent contact with the ground to prevent backward motion while adopting paddling morphologies to propel themselves forward through the slush (E). Body plans that evolved without residuals attempted to use their legs as paddles with limited success (F). In very deep slush (9m), designs are unable to touch the hard surface below the slush and must exclusively rely on fluid forces for locomotion (**G,H**). With residuals, evolution optimizes whole bodies for paddling (G); designs that evolved without residuals struggle to move forward (H). **Video summary**: youtu.be/UTUpAdbjtbc

design is particularly susceptible to exploiting simulation inaccuracies which render behavior incompetent. This is because different morphologies can interact with the same environment in different ways, uncovering new subsets of behavior-relevant dynamics, which may be more or less accurately modeled.

Prior work that evolved robots in simulation mitigated the reality gap by simplifying the environment so that it would be easier to accurately simulate (Lipson and Pollack, 2000; Hiller and Lipson, 2011; Cellucci et al., 2017; Kriegman et al., 2020a,b, 2021; Schaff et al., 2022; Matthews et al., 2023; Kobayashi et al., 2024; Strgar et al., 2024). There is only one case reported in the literature in which an evolved robot was shown to be capable of operating across a diversity of real-world terrains (Yu et al., 2026), and this was accomplished through dynamics randomization that hindered dynamics exploitation. While effective at mitigating the reality gap, these simplifications limit the upper bound on performance achievable through design optimization since they reduce the available dynamics and design complexity rather than allowing physical robots to fully exploit their environments.

To realize robot designs that exploit the dynamics available in an environment with as much grace and diversity as animals, robots need to be exposed to these dynamics during evolution. Manually adjusting simulator dynamics is neither scalable nor likely to permit designs which fully exploit available dynamics because of the difficulty in analytically modeling real-world dynamics accurately. Thus, in this work, we utilize a data-driven learning approach to cap-

ture missing dynamics by training a neural residual physics model and demonstrate zero-shot robot design with our neural augmented simulation (Fig. 1). More specifically, we explore the dynamics gap between a pair of simulated environments, one that contains the dynamics of slush (partly melted, watery snow) and one that does not. While this does not immediately solve the problem of transferring evolved designs from simulation to reality, it provides a convenient testbed with a tunable dynamics gap. We believe that this is a good place to begin to understand how to evolve morphological features in a virtual training environment that increase fitness by exploiting hidden, hard-to-model, or unknown dynamics of the actual environment.

Methods

In this section, we introduce a pair of simulation environments which share a common base simulator but differ through an explicit and controllable dynamics gap. We then discuss how dynamics data can be collected from the testing environment and used to train a neural residual physics model which augments the base training environment to recover un-modeled dynamics from the test environment. Next, an evolutionary algorithm is layered on top of these environments to optimize and evaluate robot designs. Finally, we describe how we validate our simulation results on physical hardware.

Simulation Environments: $E, E', \hat{E}$

All simulated environments consist of a base simulation provided by Box2D with a hard surface positioned d units of depth below the $y = 0$ horizontal line. We consider two simulators: E', a "dry land" environment, consisting of standard Box2D dynamics with a ground plane; and E, a "slush" environment which contains additional fluid-like forces (f_f, defined in the next subsection) that act only on the region below $y = 0$. A depth parameter d controls the vertical placement of the ground and thus the depth of the slush region. We chose to model a "puddle of slush" as it has high drag which produces a significant affect on legged locomotion at a shallow depth — and due to buoyancy entirely precludes legged locomotion with ground contact in deep puddles. At a depth of $d = 0$, the puddle "dries out" and E reduces to E'. We define a neural augmented simulator, $\hat{E}$, which applies a learned correction to the dry land dynamics to recover the slush dynamics:

$$\hat{E}(x_i) = E'(x_i) + \phi(o_i),$$

Here, ϕ predicts the residual vector $\hat{e}_i$ conditioned on the observation o_i. o_i consists of the rigid body geometry parameter (a, define below), and the full state representation of the rigid body with angle cosine-encoded. The horizontal position is omitted due to translational invariance along the x-axis. Although ϕ predicts residual corrections in the full state space, only velocity residuals are applied, since the impulse-based simulator operates at the velocity level, with positions implicitly determined via time integration.

The Fluid Model: f_f

The True Slush environment (E) is constructed as a one-way coupled rigid-fluid simulation $E = E' + f_f$, in which a fluid-drag model, f_f, injects impulses independently onto each jointed link within the rigid bodied robot in E'. The model can be decomposed into three terms: buoyancy, linear drag, and angular drag. We use this model to approximate the dynamics of a dense, high-drag fluid (i.e. slush). This regime lies between a conventional fluid and a granular suspension: exhibiting substantial drag and partial buoyancy but does not model strongly viscous or adhesive effects (e.g. as in honey or mud). We define a submerged region, A_s, as the area of the rigid body below $y = 0$. **Buoyancy** is thus $F_b = -A_s g \rho$, for submerged region A_s in a fluid of density $\rho = 1.25\ \mathrm{kg/m^2}$, and with gravitational acceleration $g = 9.8\ \mathrm{m/s^2}$. **Linear Drag** is $F_d = \frac{1}{2}\rho v^2 C_{dl} A_s$, for a linear velocity v and linear coefficient of drag $C_{dl} = 10$. **Rotational Drag** is $\tau_d = -C_{dr}\rho I_s \omega|\omega|$, for an angular velocity, ω , angular moment of inertia of the submerged rigid body region about the center of mass of the rigid body, I_s, and an coefficient of drag, $C_{dr} = 20$. This applies a rotational drag force to counter the rotation of the rigid body, and scales with the total area of the submerged region. Gravity is not included in f_f since it is already handled by the base Box2D simulator.

Dataset Collection

We probe the dynamics discrepancy between E and E' by executing rollouts in the slush environment E and collect a dataset of the simulated trajectories. (The entire training dataset is depicted in Fig. 2.) Each trajectory consists of throwing a rectangular rigid body into the slush environment E and simulating its motion for 3.3 seconds (200 time steps, dt=1/60s). Due to high drag and buoyancy individual projectiles will rarely enter deep slush, however multi-linked robots can push links into such regions. As such, some projectiles trajectories are initialized from within the slush region. For each throw, we randomly sample a projectile length, a, from $a \in \{1, 2, 3\}$ and define its geometry as height $h = a$ and width $w = 1/a$, ensuring constant area. We perform 500 independent rollouts, each initialized from a random initial state x_0. The state $x_i \in \mathbb{R}^6$ represents a planar rigid body with position (m, m), orientation (rad), linear velocity (m/s, m/s), and angular velocity (rad/s). The initial state projectile states are sampled as $x_0 \sim U(\mathcal{X}_0)$, where the state space is bounded by $\mathcal{X}_0 = [-0.2, 0.2]\ \mathrm{m} \times [-6, 6]\ \mathrm{m} \times [0, 2\pi]\ \mathrm{rad} \times [-20, 20]\ \mathrm{m/s} \times [-20, 20]\ \mathrm{m/s} \times [-20, 20]\ \mathrm{rad/s}$. At each time step, i, we record the state transition produced by the slush environment E and compare it to the dry-land simulator. The residual is defined as the difference between

the two next-state predictions: $e_i = E(x_i) - E'(x_i)$. We then train a neural network ϕ to predict this residual from an observation o_i, $\phi(o_i) = \hat{e}_i \approx e_i$. The learned residual network is used to augment the base simulator, yielding: $E(x_i) \approx E'(x_i) + \phi(o_i)$, which approximates the full slush dynamics of E.

Although this approach cannot learn to correct global dynamics that depend on interactions between multiple links (e.g. if one link is in the wake of another) it is applicable for environments that feature localized dynamics interactions due to high damping or surface contacts.

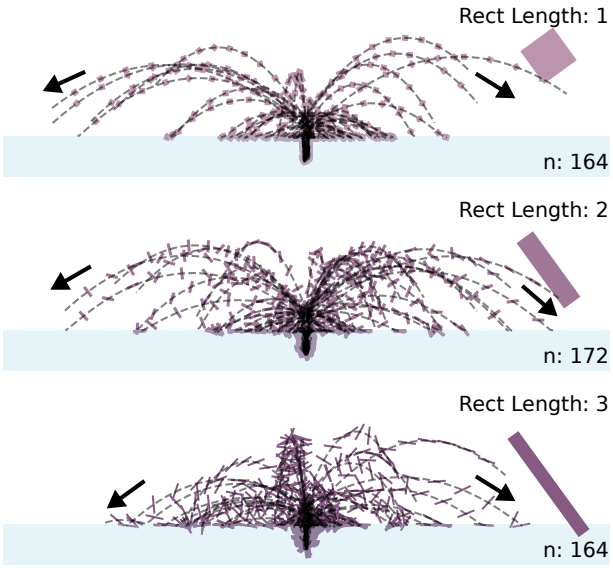

Figure 2: **The training dataset.** Three different rectangular projectiles (short, medium, long) were randomly thrown into the slush environment E a total of 500 times. The purpose of this exercise is to collect data generated by the "true" dynamics of E that are missing from the base simulator E'. We can imagine a robot arm performing similar throws in a physical environment and collecting the resulting dynamics data with motion capture or computer vision tools (Jiang et al., 2025).

Neural Network Training

We model the residual dynamics using a multilayer perceptron (MLP) with four hidden layers of width 256 and ReLu activations. Dropout ($p = 0.001$) is applied after each hidden layer. Neural network input and outputs are described above in the environment section. We train the model using AdamW (Loshchilov and Hutter, 2019) with an initial learning rate of 10^{-3} and an exponential decay after a short warmup period, and clip the learning rate to a minimum of 10^{-6}. Training uses minibatches of 262,144 sampled drawn with replacement from the dataset and proceeds for 50,000 iterations. To improve robustness, we train an ensemble of eight independently initialized networks.

We optimized a focalized mean square error loss (Lin et al., 2017) to prioritize rare high-magnitude residual events (e.g. high-speed impacts into the slush region). The per-sample loss is defined as: $\mathcal{L}_{\text{MSE},i} = \frac{1}{6}\|\phi(o_i) - e_i\|_2^2$. We then apply a focal weighting: $\mathcal{L}_i = \alpha\left(1 - \exp(-s\mathcal{L}_{\text{MSE},i})\right)^\gamma \mathcal{L}_{\text{MSE},i}$ and the total loss is $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_i$. We use $\alpha = 1.0$, $s = 100$, $\gamma = 2.0$.

Evolutionary Design

Each robot is represented as a kinematic tree, stored in topological order. A robot design consists of six rigid rectangular links connected by revolute joints. Each link is defined by: its length scale, $a \in [1, 3]$, which determines its width $(1/a)$ and height (a); an ID number as well as the ID of its parent link (parent ID $<$ child ID); the location where it attaches along the long-axis of the parent (height axis), normalized to -1 to 1; the location along the long-axis where the parent joint attaches to it (normalized to -1 to 1); a resting angle relative to the parent link. The actuation for each robot follows a sinusoidal wave with a global actuation frequency between 0 and 0.3 Hz. Each joint has a individual phase offset ($\in [0, 2\pi)$), a range of motion ($\in [0.2, \pi/2]$), and has a boolean flag to enable or disable the motorized joint.

We used Age Fitness Pareto Optimization (AFPO; Schmidt and Lipson (2011)) to co-optimize the robot's structure and behavior. AFPO is a multi-objective evolutionary algorithm that optimizes robot performance while maintaining a diversity of candidate solutions by tracking how long each design candidate and its descents have existed in the population. Here we optimize with a population of 64 designs randomly sampled. Each generation, the population is doubled by adding new random designs (15%), randomly modified copies (mutation, 65%) of existing designs (parents), and recombination (crossover, 20%) of randomly selected pairs of designs. From the enlarged population, 64 designs are selected to survive to the next generation. Specifically, all designs are categorized as dominated or non-dominated where designs are considered to be non-dominated if they have the highest fitness of all robots their age or younger and dominated otherwise. The next-generation's population is formed by selecting designs from the non-dominated pool in order of decreasing fitness until either the non-dominated pool is empty, or the surviving population is filled. Remaining slots in the surviving population are filled with designs uniformly randomly selected from the dominated pool without replacement. Evolution proceeds for 1000 generations of cumulative selection for each independent trial.

Performance Evaluation

Displacement of the robot was measured as the distance traveled by the root node in the kinematic tree during 1200 steps of simulation, after a short (200 step) period for settling under gravity. When evaluating a robot design using an

ensemble of neural residual models, we re-simulate the candidate design once for each ensemble member and utilize a single model for each rollout period (Chua et al., 2018). Robot performance is aggregated across the ensemble as the mean minus the standard deviation ($\mu - \sigma$) of the performance achieved when augmenting the training environment with each ensemble member.

Manufacturing

The main experiments permit link interpenetration so as to permit more complex behaviors despite the constraints of 2D simulation. To simplify the transfer to reality, we required that links not directly connected by a joint to not collide in their rest pose and prevented such links from interpenetrating during simulation. To match available motor hardware, we also lowered the simulated motor torque from 100Nm to 30Nm and restricted the number of actuated joints to two. Remaining joints were locked in their rest poses, forming up to three distinct rigid bodied structures. This torque limitation effectively reduced the realizable motor speeds in simulation. As such, we increased the evaluation period (to 5000 steps) as well as the settling period (1000 steps).

Optimized designs are manufactured at a scale of 70mm per meter of simulation, and extruded 25mm into 3D and replicated a second time 300mm along the extrusion dimension to form two pontoons. Servo-motor cradles and joint interfaces were manually added onto the CAD files and 3D printed with 45% infill in PLA (Polylactic acid). The two pontoons are connected via 3mm stainless steel rods and actuated using a 20kg-cm servo powered via an umbilical cord supplying 3V and pulse-width modulated (PWM) control commands.

We constructed two physical environments which are analogous to our simulated dry land and slush environments. A plastic tank (117 cm long, 33 cm wide, and 27 cm tall) was filled with 15 cm of water mixed with clear sodium polyacrylate gel beads which are nearly neutrally buoyant in the water. The gel beads increase the fluid damping in the tank, more closely matching the simulated slush environment with a fluid of depth d above a hard surface. Draining the fluid and drying the surface provides a physical dry land environment. Physical robots were placed at one end of the tank and evaluated for up to 2 min 30s or until they reached the opposite end of the tank.

Results

In this section we evaluate the prediction accuracy of the ensemble of residual models and compare against the individual accuracy of each submodel. Next, we quantify the performance of robots evolved across various depths of slush in the three different training environments (E, E', $\hat{E}$). Finally, we assess the validity of our Dry Land (E') and True Slush (E) simulation environments by transferring one of the designs evolved with residuals (in $\hat{E}$) to analogous physical environments.

Prediction Accuracy

The utility of a neural residual physics model for robot design depends on how well it predicts the true behavior of candidate designs. Prior to running evolutionary optimization with our residual augmented environment, we first assess its prediction accuracy using 2376 designs discovered during evolution across True Slush and Dry Land simulation environments. To ensure a variety of design behaviors and performances, the designs were sampled uniformly across all generations of evolution and (re)simulated nine times with a slush depth of $9m$, once in True Slush and once in each of the eight submodels within the Residual Slush Ensemble. For each ensemble submodel, a 2D histogram of predicted and actual fitness is plotted in Fig. 3A-H with a regression line (solid) along with an ideal $y = x$ line (dashed). The predictive accuracy of the submodels varies considerably, with Pearson correlation coefficients ranging from 0.24 to 0.68 and regression slopes ranging from 0.25 to 0.76. However, aggregating the predicted fitness scores across the ensemble (i.e. mean submodel fitness) produces a strong Pearson correlation coefficient of 0.77 and a slope of 1.26 (Fig. 3I, solid).

Because any differences in fitness between submodels must derive from differences in learned residual dynamics, designs that exhibit high fitness variance across ensemble submodel evaluations are likely to be experiencing "hallucinated" dynamics in some submodels. We can say that the robot is exploiting hallucinated dynamics if its actual fitness is worse than the predicted fitness.

Indeed, any deviations in regression slope from 1 imply persistent dynamics inaccuracies since locomotion is directly tied to fitness. For example, a slope of 1.26 implies that every 1 meter a robot moves through the Residual Slush environment, it is likely to move 1.26 meters in the True Slush environment. While this may result in some robots performing better in true slush than predicted by the residual ensemble, it also implies that the neural simulations are systematically misrepresenting the hidden dynamics. Designs are likely to transfer well only if they exploit neural dynamics that are available in the actual environment. As a result, optimization within neural simulations risks discovering designs which transfer poorly.

To mitigate the risk of an optimizer converging to deceptive regions, we included a penalty term to discourage ensemble uncertainty: ensemble aggregate fitness was measured by the mean minus the standard deviation ($\mu - \sigma$) of each submodel's fitness. This in effect prioritizes designs for which the ensemble predicts high performance and is confident in that prediction. This improved the slope of the regression (1.26 to 0.99; dotted line in Fig. 3I), but caused the Pearson correlation coefficient to drops slightly (0.77 to 0.6). While the Pearson correlation coefficient drops some-

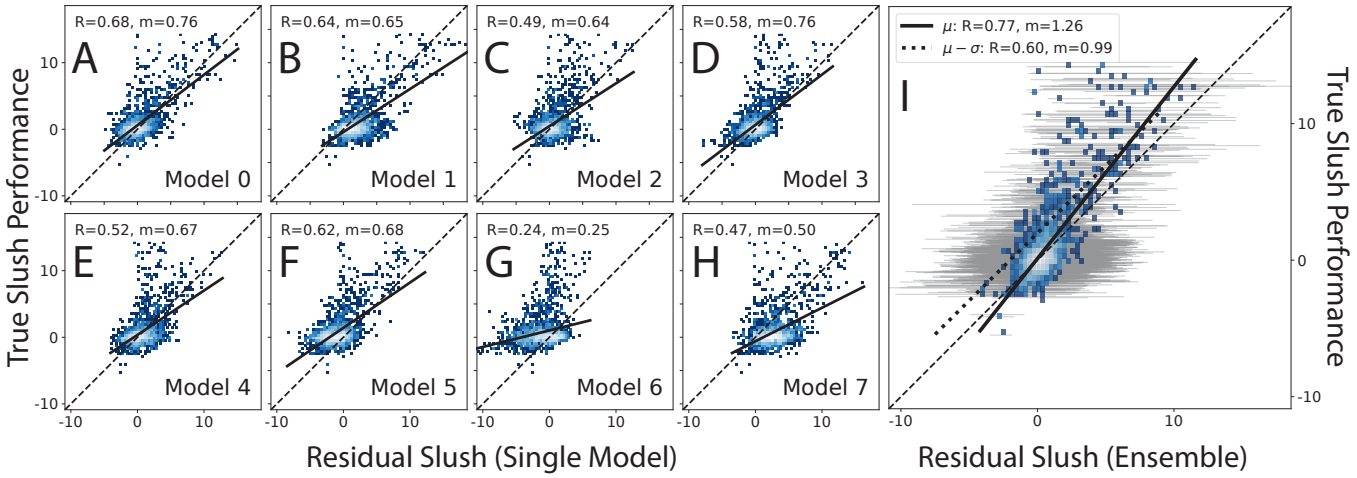

Figure 3: **An ensemble remediates hallucinated dynamics.** A subset of robot designs (2376) discovered across all generations of evolution in the True Slush and Dry Land environments were sampled and resimulated in True Slush and once in each of the 8 Residual Slush ensemble submodels. Linear regressions were performed between True Slush performance and performance as evaluated under each submodel (**A-H**). Model quality varied considerably across each ensemble submodel, with regression slopes (m in each subpanel) ranging from 0.25 to 0.76 and with Pearson correlation coefficient (R in each subpanel) ranging from 0.24 and 0.68. Slopes that deviate from the ideal of 1 (dashed lines) indicate dynamics inaccuracies in the neural augmented environment relative to True Slush. With an ensemble (**I**), prediction quality improves. Directly averaging fitness across submodel evaluations produces a correlation coefficient of 0.77 and a slope of 1.26. However, ensemble disagreement (min/max ensemble component fitnesses are denoted with grey horizontal lines) represents differences in learned dynamics. Penalizing ensemble fitness by fitness disagreement across submodel evaluations helps to ensure that designs are not exploiting hallucinated dynamics which only exist in some submodels. Using this aggregation ($\mu - \sigma$) of submodel evaluations across the ensemble results in a nearly ideal regression slope of 0.99 although with a slightly reduced correlation coefficient of 0.6.

what, the improved regression slope is desirable. For our goal of zero-shot morphology design, overestimating fitness is substantially worse than underestimating fitness. Underestimating fitness will at worst cause optimization to overlook potentially good designs if/when other search areas are perceived to be more attractive. In contrast, overestimating performance in the neural environment can distort the fitness landscape such that an optimizer converges to deceptive designs which are non-performant in the test environment.

Evolution

As discussed above, the True Slush test environment (E) converges to the unaugmented base simulation training environment (Dry Land; E') as the simulated slush puddle approaches a depth of $d = 0$ and diverges as puddle depth increases. With this adjustable dynamics gap between E and E', 10 independent evolutionary trials were conducted in each of the three simulation environments (E, E', and $\hat{E}$), for each of 10 puddle depths $d \in \{0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 9\}$. At the end of each trial, the best design discovered by evolution (the "run champion") was zero-shot evaluated in True Slush.

With very deep slush (9m, Fig. 4 A) the dynamics gap is maximized since any dynamics of the hard ground are out of reach. In this setting, an upper bound on performance was established by directly evolving designs in True Slush (blue bar in Fig. 4A; mean displacement 13.6 m), which allows evolution to fully exploit the dynamics resident in the test environment. Designs evolved in the base simulation environment, Dry Land (dark green bar in Fig. 4A), fail to zero-shot transfer into the True Slush (mean -0.8 m).

This degradation in fitness could be because the morphology or the controller or both are ill-suited for True Slush. In order to isolate the potential fitness of a design, we froze the evolved morphology and retrained the control policy from scratch (1000 generations of evolution, with morphology parameters frozen) thereby adapting behavior to the True Slush (light green, hatched in Fig. 4A). With adapted control policies the mean displacement in True Slush of designs evolved on Dry Land increases from -0.8 to 4.8 m, but still only achieves 35% of mean fitness of the designs evolved directly in the True Slush (blue). Using our ensemble of neural residual simulations, evolved designs recover 67% of True Slush dynamics with their evolved controllers (dark orange in Fig. 4A) and 87% with controller adaptation (light orange, hatched in Fig. 4A). Even without controller adaptation, these designs are significantly more performant in True Slush than those evolved on Dry Land with controllers adapted for True Slush (Student's t-test, $p < 0.0001$; n=10).

These results also show that the ensemble was neces-

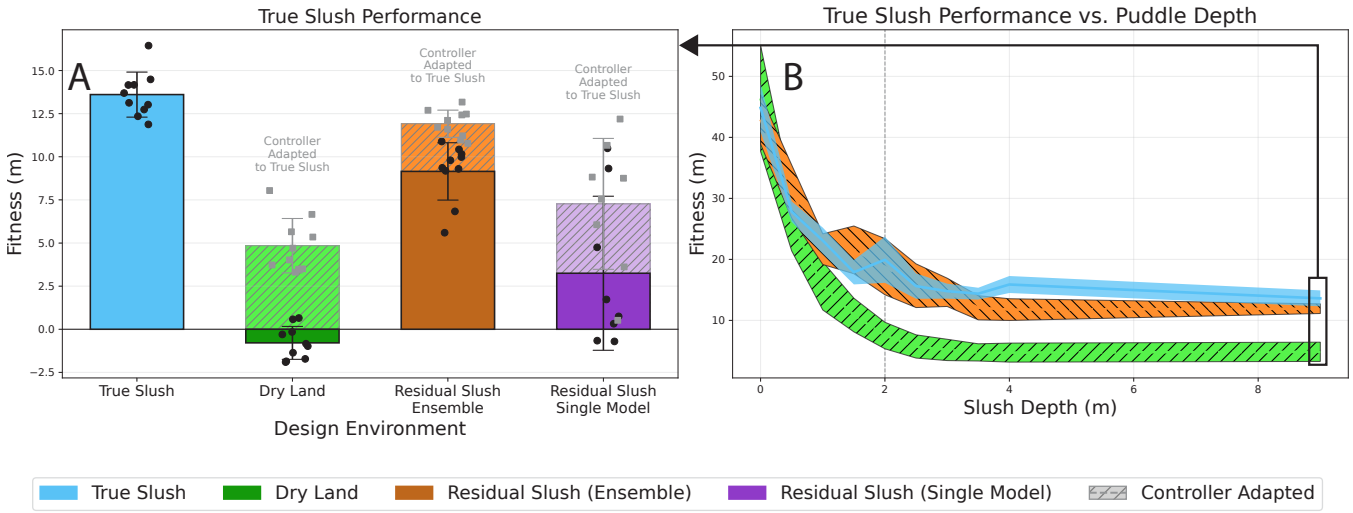

Figure 4: **Zero-shot robot design.** Body plans and control policies were co-optimized for forward locomotion in a simulated environment and zero-shot tested in the True Slush environment, which consists of a simulated puddle of slush. True Slush is a simulation, but it contains "unknown" dynamics that are not present in the training simulators. **A:** In deep slush (9m) where dynamics differences between training and testing environments are maximal, an upper bound on task performance is established by directly co-optimizing robot morphologies and controllers in the True Slush environment (blue). Robots that evolved on Dry Land achieve near zero fitness when transferred to True Slush (dark green). The poor fitness of these robots was due to their morphology, as evidenced by the lack of fitness recovered when their controllers are retrained in True Slush (hatched light green). Robots that evolved in our Residual Slush Ensemble exhibited significantly higher fitness in True Slush (orange) than those that evolved on Dry Land without residual physics; after retraining their controllers for True Slush (hatched light orange) these designs achieved 87% of maximal fitness (blue). These fitness gains were due to both the morphology of the evolved robots and the neural ensemble used to augment the base simulator. Replacing the ensemble with a single neural net (purple) induces hallucinated dynamics that become exploited by evolution, yielding designs that are ill-suited for True Slush. This holds true even after retraining all controllers in True Slush (hatched areas). **B:** In shallow slush ($<$ 1m) morphologies evolved on Dry Land (hatched green) are nearly equally capable of locomotion as morphologies evolved with residuals (hatched orange) or directly in True Slush (blue). In deeper slush ($>$ 1m), the dynamics gap between Dry Land and True Slush increases, and morphologies evolved on Dry Land have limited capacity for locomotion. In contrast, across all slush depths, morphologies evolved with residuals are nearly as capable as those evolved directly in True Slush. Vertical dotted line at 2m corresponds to the slush depth used for evolving the physical robot. Uncertainty ranges correspond to $\pm$ 1 std dev across 10 independent trials.

sary: When using a single model (purple in Fig. 4A), there was very high variation in fitness across independent trials. This high variance across independent trials is only observed when optimizing with individual neural models, and is likely attributable to model prediction quality as distinct single-model optimization trials use independent neural models (c.f. Fig. 3A-H, where each ensemble submodel is evaluated). This suggests that the designs that evolved in a single neural residual model frequently exploit hallucinated dynamics that are not available in True Slush—and that the ensemble successfully mitigated these hallucinations by forcing the submodels to agree on the residual physics.

Sweeping from shallow to deep slush depths (Fig. 4 B), we observe that at low slush depths ($<$ 1 m), designs that evolved on Dry Land achieve near optimal fitness in True Slush (Student's t-test measures insignificant ($p > 0.01$) difference between Dry Land designs and True Slush for 0.5m of slush depth). In deeper slush, however, designs that evolved on Dry Land significantly under-performed both those evolved in Residual Slush and those evolved directly in True Slush (Student's t-test, $p < 0.0001$ for all slush depths > 0.5m). The fitness of designs evolved in our residual ensemble are insignificantly different from those evolved in True Slush up until 3m of slush (Student's t-test, $p > 0.01$), and in very deep slush (9 m) they recover most (87%) of the fitness achieved by designs evolved in True Slush.

Exemplar designs evolved with and without learned residual physics at various slush depths are shown in Fig. 1. At increasing slush depths, robots evolved with learned residual physics exhibit distinct morphologies to walk with legs (Fig. 1A), leap out of the water (Fig. 1C) in a manner reminiscent of skittering locomotion (Weiss et al., 2024), paddle with intermittent ground contact to prevent backward locomotion while prepping for the next paddle stroke (Fig. 1E), and paddle without ground contact (Fig. 1G). Robots evolved in the base simulator without learned residual dynamics (Fig. 1B,D,F,H) all learn legged locomotion during training. When adapting their controllers for the true

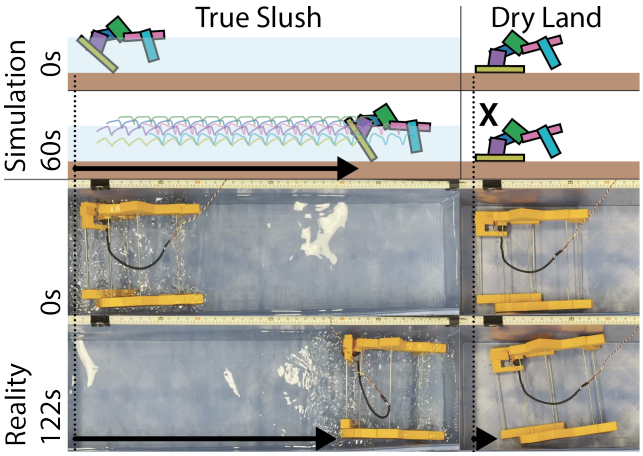

Figure 5: **Physical validation.** In this paper, we learned the residual dynamics between two simulations: Dry Land and True Slush. In this sense, the True Slush testing environment is meant as a cheap substitute reality. To understand how well True Slush imitates hidden physical dynamics, physical analogues of True Slush and Dry Land were constructed. A representative design evolved in the Residual Slush environment ($d = 2$) successfully locomotes across the True Slush simulation, however its motors have insufficient torque to move across the Dry Land simulation. This design was built and tested in physical slush and on physical dry land. The performance of the physical design matched that of its simulated counterpart, locomoting in the physical slush (crossing the tank in all 8 trials) but due to insufficient torque was unable to lift its body above the dry surface and could only move a few centimeters during the 150s evaluation period (in all 6 trials). This suggests that our synthetic dynamics gap provides a reasonable testbed for exploring how residual physics may be learned from real world data.

slush environment, legged locomotion remains effective at shallow depths (B,D), however in deeper slush environments some of the links are repurposed for paddling (Fig. 1F,H) with limited success.

Physical Validation

For our physical validation experiments, 16 independent evolutionary trials were conducted using our Residual Slush Ensemble with a slush depth of $d = 2$. The best design (run champion) discovered by evolution in each trial was zero-shot tested in True Slush. A run champion with average test performance was selected for manufacture (Fig. 6).

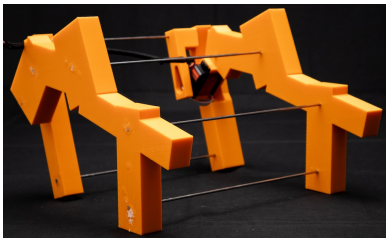

Figure 6: The evolved physical design used for validation.

The robot's behavior in simulated and physical slush were very similar: the front body links float near the slush surface level, and infrequently touch the hard floor bottom of the tank; the rear leg pushes against the floor propelling the body forward (Fig. 5). On dry land, the simulated behavior also matches the physical behavior quite well with both struggling to lift their own body weight resulting in almost no forward motion (Fig. 5). Hence the buoyancy provided by the physical and simulated slush was critical to the locomotive ability of both the physical and simulated design.

Discussion

In this paper, we showed for the first time the evolution of robots within a neural residual augmented training environment. The designs that evolved with residuals possessed morphological features that better exploited the hidden dynamics of the test environment compared to designs that evolved without residuals (Fig. 1). By retraining policies in the test environment, we showed that the fitness gains provided by neural residuals were due to morphological innovations rather than better control of behavior (Fig. 4). We also showed how evolving designs in a single neural augmented simulation can produce hallucinated dynamics that are mitigated through an ensemble approach (Fig. 3).

Instead of collecting data from the real world, residual models were here trained from data collected entirely in simulation (Fig. 2). We chose to investigate a sim-to-sim dynamics gap as it provides clarity into the specific dynamics that could be learned and exploited, and because it is a convenient testbed for rapid algorithm development. We validated the relevance of our simulation environments and the dynamics gap between them by transferring a design discovered with residual physics (Fig. 6) to analogous environments in the real world and demonstrate that, like its simulated counterpart, it exploits dynamics of the physical slush to locomote through it (Fig. 5). Because these dynamics are not available outside the slush, the physical robot was unable to locomote on dry land.

In future work we will extend our approach to directly model sim-to-real dynamics gaps in order to evolve designs in neural residual physics that exploit more exotic, perhaps previously unknown dynamics in the physical world. However, we plan to begin by learning design-relevant residuals from a physical environment (e.g. fluid) where analytical models are well studied and importantly, the limitations of which are well understood. If successful, we will begin to consider physical environments that demand more careful dynamics data collection Berrueta et al. (2024). In more complex environments, dynamics are dependent not only on the robot's state but also on the state of the environment. In these settings, it may be necessary to train on temporal observation data and multi-modal data.

Acknowledgments

This research was supported by NSF awards 2331581 and 2440412, and TWCF award 20650. DM was supported by NSF GRFP DGE-2234667. Portions of the code in this study were written with the assistance of an AI coding tool.

References

Ajay, A., Wu, J., Fazeli, N., Bauza, M., Kaelbling, L. P., Tenenbaum, J. B., and Rodriguez, A. (2018). Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3066–3073. IEEE.

Allen, K., Lopez-Guevara, T., Stachenfeld, K. L., Sanchez Gonzalez, A., Battaglia, P., Hamrick, J. B., and Pfaff, T. (2022). Inverse design for fluid-structure interactions using graph network simulators. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 13759–13774. Curran Associates, Inc.

Bauersfeld, L., Kaufmann, E., Foehn, P., Sun, S., and Scaramuzza, D. (2021). NeuroBEM: Hybrid Aerodynamic Quadrotor Model. In *Proceedings of Robotics: Science and Systems*, Virtual.

Berrueta, T. A., Pinosky, A., and Murphey, T. D. (2024). Maximum diffusion reinforcement learning. *Nature Machine Intelligence*, 6(5):504–514.

Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121.

Cellucci, D., MacCurdy, R., Lipson, H., and Risi, S. (2017). 1d printing of recyclable robots. *IEEE Robotics and Automation Letters*, 2(4):1964–1971.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, 521:503–507.

Gao, J., Michelis, M. Y., Spielberg, A., and Katzschmann, R. K. (2024). Sim-to-real of soft robots with learned residual physics. *IEEE Robotics and Automation Letters*.

Golemo, F., Taiga, A. A., Courville, A., and Oudeyer, P.-Y. (2018). Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*, pages 817–828. PMLR.

Heiden, E., Millard, D., Coumans, E., Sheng, Y., and Sukhatme, G. S. (2021). Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, page 9474–9481. IEEE.

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

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872.

Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720.

Ji, G., Mun, J., Kim, H., and Hwangbo, J. (2022). Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7(2):4630–4637.

Jiang, H., Hsu, H.-Y., Zhang, K., Yu, H.-N., Wang, S., and Li, Y. (2025). Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos. *ICCV*.

Jing, C., Bandi, J. K., Ye, J., Duan, Y., Abbeel, P., Wang, X., and Yi, S. (2026). Contact-aware neural dynamics.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. (2023). Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987.

Kobayashi, H., Gholami, F., Montgomery, S. M., Tanaka, M., Yue, L., Yuhn, C., Sato, Y., Kawamoto, A., Qi, H. J., and Nomura, T. (2024). Computational synthesis of locomotive soft robots by topology optimization. *Science Advances*, 10(30):eadn6129.

Koos, S., Mouret, J.-B., and Doncieux, S. (2010). Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, page 119–126, New York, NY, USA. Association for Computing Machinery.

Kriegman, S., Blackiston, D., Levin, M., and Bongard, J. (2020a). A scalable pipeline for designing reconfigurable organisms. *Proceedings of the National Academy of Sciences*, 117(4):1853–1859.

Kriegman, S., Blackiston, D., Levin, M., and Bongard, J. (2021). Kinematic self-replication in reconfigurable organisms. *Proceedings of the National Academy of Sciences*, 118(49):e2112672118.

Kriegman, S., Nasab, A. M., Shah, D., Steele, H., Branin, G., Levin, M., Bongard, J., and Kramer-Bottiglio, R. (2020b). Scalable sim-to-real transfer of soft robot designs. In *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 359–366.

Kumar, A., Fu, Z., Pathak, D., and Malik, J. (2021). Rma: Rapid motor adaptation for legged robots. In *Proceedings of Robotics: Science and Systems (RSS)*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Matthews, D., Spielberg, A., Rus, D., Kriegman, S., and Bongard, J. (2023). Efficient automatic design of robots. *Proceedings of the National Academy of Sciences*, 120(41):e2305180120.

O'Connell, M., Shi, G., Shi, X., Azizzadenesheli, K., Anandkumar, A., Yue, Y., and Chung, S.-J. (2022). Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597.

Schaff, C., Sedal, A., and Walter, M. (2022). Soft Robots Learn to Crawl: Jointly Optimizing Design and Control with Sim-to-Real Transfer. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA.

Schmidt, M. and Lipson, H. (2011). Age-fitness pareto optimization. In *Genetic Programming Theory and Practice VIII*, pages 129–146. Springer.

Serifi, A., Knoop, E., Schumacher, C., Kumar, N., Gross, M., and Bächer, M. (2023). Transformer-based neural augmentation of robot simulation representations. *IEEE Robotics and Automation Letters*, 8(6):3748–3755.

Sontakke, N., Chae, H., Lee, S., Huang, T., Hong, D. W., and Hal, S. (2023). Residual physics learning and system identification for sim-to-real transfer of policies on buoyancy assisted legged robots. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 392–399. IEEE.

Strgar, L., Matthews, D., Hummer, T., and Kriegman, S. (2024). Evolution and learning in differentiable robots. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands.

Weiss, T., Gillis, G. B., Van Mullekom, J., and Socha, J. J. (2024). Skittering locomotion in cricket frogs: a form of porpoising. *Journal of Experimental Biology*, 227(22).

Yi, S., Bai, X., Singh, A., Ye, J., Tolley, M. T., and Wang, X. (2025). Co-design of soft gripper with neural physics. In *Conference on Robot Learning (CoRL)*.

Yu, C., Matthews, D., Wang, J., Gu, J., Blackiston, D., Rubenstein, M., and Kriegman, S. (2026). Agile legged locomotion in reconfigurable modular robots. *Proceedings of the National Academy of Sciences*, 123(10):e2519129123.

Zeng, A., Song, S., Lee, J., Rodriguez, A., and Funkhouser, T. (2020). Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319.