Text2Robot: Evolutionary Robot Design from Text Descriptions

Ryan P. Ringel*, Zachary S. Charlick*, Jiaxun Liu*, Boxi Xia and Boyuan Chen

***Abstract*— Robot design has traditionally been costly and labor-intensive. Despite advancements in automated processes, it remains challenging to navigate a vast design space while producing physically manufacturable robots. We introduce Text2Robot, a framework that converts user text specifications and performance preferences into physical quadrupedal robots. Within minutes, Text2Robot can use text-to-3D models to provide strong initializations of diverse morphologies. Within a day, our geometric processing algorithms and body-control co-optimization produce a walking robot by explicitly considering real-world electronics and manufacturability. Text2Robot enables rapid prototyping and opens new opportunities for robot design with generative models. Our website is at `http://generalroboticslab.com/Text2Robot/`.**

I. Introduction

For over half a century, robot design has been a costly and labor-intensive process, requiring extensive human efforts from initial sketches to detailed modeling, prototyping, controller design, manufacturing, and testing. This traditional approach has significant limitations, such as prohibitive costs, lengthy development cycles, and constraints on innovation bounded by human imagination and manual capabilities. However, advancements in automated robot design [1–4] promise to revolutionize this landscape. By automating key aspects of the design process, we can drastically reduce development time and costs, allowing industries to rapidly produce specialized robots and enabling engineers to establish efficient manufacturing processes. Researchers also benefit by quickly innovating desired hardware platforms. Ultimately, automating robot design not only enables rapid prototyping but also expands the realm of possible innovations, surpassing the boundaries of what human designers can envision and create.

One major challenge in automating robot design is navigating the vast and intricate design space. Traditional engineering design is time-intensive and demands considerable technical expertise. While advancements in control engineering [5–9] and machine learning [10–14] have enabled automatic training of robot policies, designing morphologies remains laborious. Human designers typically spend months conceptualizing, designing, and fabricating a robot, balancing cost, manufacturability, and performance. Previous automation attempts often simplify the design space by using large repeating modules [1, 15–17] or voxel representations [18, 19]. Though innovative, these designs are slow due to the need to search within a vast design space and do not consider real-world fabrication, resulting in theoretically sound but impractical designs to produce.

This work is supported by DARPA FoundSci program under award HR00112490372, by ARL STRONG program under awards W911NF2320182 and W911NF2220113. All authors are from Duke University. *Equal contribution.

Fig. 1: **Text2Robot** creates physical robots from user-specified text prompts and performance preferences while considering real-world electronics and manufacturability.

Automated methods for robot design predominately involve Evolutionary Algorithms (EAs) inspired by natural evolution [20]. However, EA-based approaches are inherently slow, starting with random solutions and iterating through hundreds of generations. Moreover, existing solutions do not scale well with increasing design complexity [21] and face significant challenges in balancing multiple objectives, such as control and morphology co-optimization [17, 22]. Consequently, while these solutions may excel in simulations, they frequently fail to address practical issues such as sim2real transfer [16–18] and manufacturability. Problems like high current draw from unrealistic degrees of freedom [16, 17] or complex morphologies that are difficult to manufacture [16–18] hinder the transition from theoretical designs to practical and producible robots.

We present Text2Robot (Fig. 2), an "A-to-Z" framework from user text specifications to physical walking robots. Our approach utilizes recent advancements in text-to-3D generative models to create initial mesh designs, which are subsequently converted into kinetic robot models through our geometric processing algorithms. Within minutes we can generate a design, within an hour, a robot trained in simulation, and within a day, a fabricated walking robot. Our system not only fulfills users' aesthetic preferences but further optimizes the designs using an evolutionary algorithm to incorporate other performance preferences. Our key insight is that text-to-3D generative models can provide a much stronger starting point for the evolutionary algorithm, significantly accelerating the optimization process. Experiments in both simulation and the physical world demonstrate our ability to specify both aesthetic qualities and performance

metrics, such as velocity tracking and energy efficiency. Overall, our approach introduces the creative and artistic nature of generative models to automated robot design with fast prototyping from a text prompt, and has the potential to open up novel opportunities in rapid design and manufacturing.

II. Related Work

Generative Models for Design Generative AI aims to enhance design efficiency [23, 24] and reduce labor costs [25, 26] by automating tasks [27] or portions of the workflows [28–30]. Recent advances in generative models have led to the integration of generative adversarial networks (GANs) [31, 32], diffusion models [33, 34], and transformers [35, 36] into the generative design process. Previous research has utilized generative models for domain-specific code creation [37, 38] or parameter-based physical designs [39, 40]. Such systems often heavily rely on human feedback and tuning during the design process. In contrast, our work focuses on directly enabling artificial intelligence-generated content (AIGC) for physically embodied and functional robot design.

Text-to-3D Models Recent advancements in text-to-3D generative models [41] utilize pre-trained text-to-image diffusion models to optimize Neural Radiance Fields (NeRF) [42–44], text-to-3D shape embeddings in a GANs framework [45], or explicit and hybrid scene representations [46] to create 3D generated content. Despite their growing capabilities, current text-to-3D models are primarily used for visualization in graphics and not for the creation of functioning machines. Our work incorporates 3D AIGC into physical robot design while considering electrical components and manufacturability constraints.

Automated Robot Design Automated robot design often involves co-optimizing control and morphology using classical methods and analytical dynamics [47–50]. These approaches rely on highly parameterized designs and specified models, which limit creativity and practicality in real-world environments. Recent studies employ reinforcement learning for co-optimization through optimizing the distribution of design parameters[51, 52] or auto-differentiable hardware policies [53–55], but their designs focus on limited design space and often overlook real-world manufacturability and electronics. Evolutionary Algorithms (EA) have been widely used [56] to explore vast design spaces by combining modular building blocks [1, 15, 16, 57] or voxels [18, 19, 22] through genetic operators to co-evolve morphology and control to produce complex and unexpected [58] designs. However, EA-based approaches are computationally expensive, typically starting from random solutions and requiring numerous generations to produce effective designs. Our work leverages the open-ended nature of EA-based approaches for design evolution but drastically accelerates the process through strong initializations by incorporating generative models.

III. Approach

Text2Robot (Fig. 2) generates a physical walking quadrupedal robot that caters to a user's text description and performance priorities, such as energy efficiency or velocity tracking accuracy. There are four major components in Text2Robot: (1) A generative model to create static 3D meshes of robots given user-specified texts. (2) A set of geometric processing algorithms to convert the static meshes into kinetic models, including the necessary components for fabrication. (3) An optimization process based on evolutionary algorithms and reinforcement learning to further optimize the robot morphologies and walking policies according to the user's performance preferences. (4) A final optimized robot is quickly 3D-printed and assembled.

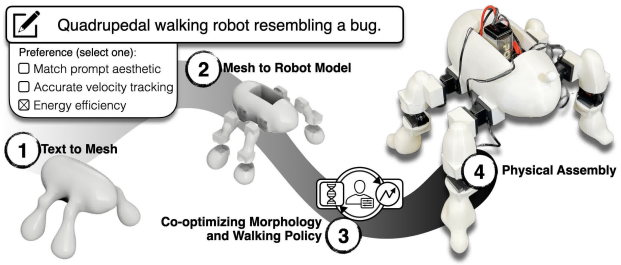

Fig. 2: Overview of the four steps in Text2Robot framework.

A. Mesh Generation from Text Prompts

Given a text prompt specifying the aesthetic of the robot, we generate a 3D mesh of the robot using a text-to-3D model. In this paper, we used one of the state-of-the-art models, Meshy [59], that takes in the text prompt and produces several candidate meshes. One high-level assumption in this paper is to demonstrate our framework by automating the design of quadrupedal robots with eight motors. Having such constraints mimics the typical real-world design requirements without the loss of generality of our framework. We implemented a structured prompt design based on specified user descriptions to ensure Meshy consistently outputs quadrupedal meshes.

The user provides a text description of their desired robot in one to three words, which we incorporate into the following prompt format: `<Quadrupedal walking robot resembling a "User-Provided Description">`. The generated candidate meshes are then manually filtered according to the following constraints: (1) the mesh must be continuous without disjoint bodies; (2) the mesh must exhibit bilateral symmetry; and (3) the mesh must include four legs.

B. Kinetic Robot Model from Static Meshes

Current text-to-3D models only produce static meshes for visualization purposes. Our key challenge is to automatically convert such static meshes into kinetic robot models. Importantly, unlike most simulated robot models that are simplified for fast simulation, to automatically transfer our designs to physical functioning robots, our generated robot model should also consider real-world manufacturing factors such as the placement of electronic components, wire connections, physical collisions at joints, limits of the number of motors, and manufacturability.

Mesh Repair and Preprocessing Due to the lack of realistic

constraints on the text-to-3D models, the generated mesh can have errors such as being non-watertight. We first call the mesh repair API through Fusion 360 [60] to repair the mesh for the downstream workflow. We then leverage the mesh conversion operation to convert the mesh to an organic BREP (Boundary Representation) body. We scale the BREP body to a volume of 6300cm^3 to unify the initial mass of all robots.

Joint Allocation Deciding a set of feasible joint positions for a quadrupedal robot with eight motors purely from the mesh model is difficult. Inspired by natural quadrupedal animals, we assume that our robots have four legs and two movable joints for each leg. In other words, our robots have four shoulder joints and four knee joints, dividing each leg into an upper and lower leg.

We determine the position and orientation of each joint based on the mesh model's geometric features (Fig. 3 A and B). The body's origin is defined as the center of mass, assuming uniform mass distribution. Starting from the origin along the $+y$ and $-y$ direction, we create vertical slices parallel to the xz plane and record the cross-sectional area at each step. The slice closest to the origin with a local minimum in cross-sectional area is mirrored to the other side of the origin. The two slicing planes separate the mesh into four legs and one body, defined as the base link. The origins of the four shoulder joints correspond to the centroids of their intersection profiles, with z-axes perpendicular to the slicing plane and y-axes pointing downwards. Similarly, knee joints are located at the slice planes with maximum cross-sectional area, traversing along the xy plane for each leg. We traverse from the bottom of the base link and stop 2cm from the ground to ensure space for motor placement. The origins of the knee joints correspond to the centroids of each intersection profile, with z-axes perpendicular to their slicing planes and x-axes parallel to the slicing planes pointing in the robot's facing direction. We follow the right-hand system.

Electronic Component Placement To create robots for real-world manufacturing and walking, we need to consider the placements of electronic components, including actuators, batteries, and controllers. As in Fig. 3C, servo motors are housed in a 3D-printed box with snap-in pegs inspired by toy building blocks. Motors can be rotated in their casings to achieve the desired axis of rotation during assembly. A larger box encases other necessary electronics, such as the motor controller, Raspberry Pi, and battery, and can be easily slid into a central channel in the base link. To avoid self-collision and reserve space for motor and electronics modules, we offset the eight limbs by 4cm and cut extrusions.

By this step, the robot's kinetic model is automatically defined by our algorithms, and our pipeline exports the model to Unified Robotics Description Format (URDF) [61]. To enlarge the design space for optimization in Sec. III-C, we scale each leg in 0.5cm increments to generate nine additional models. We further augment these models to thirty variants by defining each joint's rotational axis along the $x, y,$ or z axis of the joint coordinate.

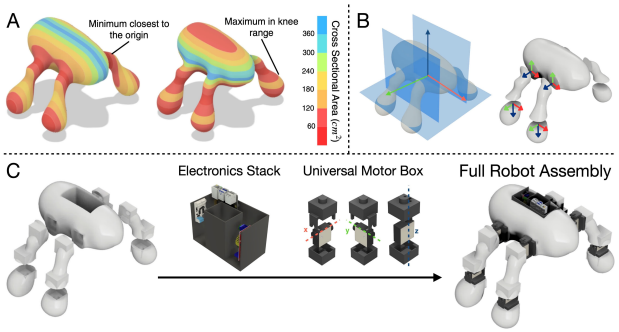

Fig. 3: **Geometric Processing.** (A) Heat maps to visualize the cross-section area from the XZ (left) and XY plane (right). (B) The selected planes for slicing the mesh model and the coordinate of the center of mass and the resulting robot components and their joint coordinates. (C) The final robot model with extruded boxes for electronics and motors.

C. Co-Optimization of Morphology and Policy

We propose an evolutionary algorithm (Fig. 4A) with a dual-loop architecture to optimize both robot morphology and control policy while simultaneously incorporating user preferences. The inner loop employs reinforcement learning to assess a robot's capacity for acquiring a walking policy. The outer loop utilizes genetic operators to evolve the robot morphology from a design repository.

An initial population of robots will be trained for locomotion to track changing velocities along $x-y-yaw$ directions. The robot observations include the previous actions, base linear and angular velocities, joint positions and velocities, the gravity vector projected onto the robot's coordinate system, and the target body velocity commands. The policy outputs the joint position offsets that will be converted to torques with a PD controller. The reward function consists of a baseline reward (r_{baseline}) and optional user-adjustable reward terms based on preference. r_{baseline} minimizes velocity tracking error, maintains a plausible robot pose, and penalizes excessive joint torque, accelerations, frequent stepping behavior, and abrupt action changes. The user-specified components are the linear velocity tracking reward and joint power penalty. The per-step reward can be defined as:

$$r = \underbrace{\alpha_1 e^{-0.25\left\|\mathbf{v}^*_{xy}-\mathbf{v}_{xy}\right\|^2}}_{\text{linear velocity tracking}} + \underbrace{\alpha_2 \sum_{i=0}^{n} \left\|\dot{q}_i \tau_i\right\|}_{\text{joint power}} + r_{\text{baseline}} \quad (1)$$

where $\mathbf{v}^*_{xy}$ is the commanded base linear velocity in xy direction, and $\mathbf{v}_{xy}$ is the actual base linear velocity, $\dot{\mathbf{q}}$ is the joint velocities, τ is the joint torques, n is the number of joints and α_1 and α_2 are the weights of the respective reward terms. r_{baseline} are listed in the supplementary material.

We select the top 100 robots for each generation and create another 100 new robots with genetic operators. 50 of the new robots are created through mutations, and the other 50 are created through crossovers. Crossover is achieved by duplicating a random robot in the current generation and choosing with a 50% chance to swap a joint or a limb with another random robot in the current generation. For the mutations, the

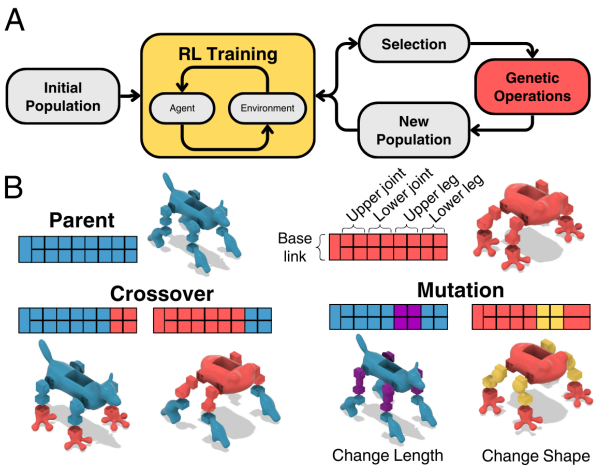

Fig. 4: **Morphology and Walking Policy Co-optimization.** (A) The inner loop implements reinforcement learning to optimize the robot control policy, and the outer loop optimizes the robot morphologies through genetic operations. (B) Our genetic representation and examples of crossover and mutation operation.

robot can undergo a change in limb length, limb shape, body shape, joint axis or remain the same with the possibility of 15%, 15%, 25%, 40%, and 5%, respectively. A constraint of symmetry is imposed on the robot based on the observation of natural animal characteristics and the practicality of our implementation. To achieve this, the same change is applied to all legs or joints to unify the joint type and length or shape of the limb within the same body level. Although a larger design space can be achieved without this constraint, this ensures our robot has a stable starting pose at the beginning of the inner loop training and improves the training quality.

By defining the fitness score based on different evaluation metrics, our outer loop can prioritize different performances for robot selection and optimize both morphology and walking policy toward user preference. We design two criteria for users to prioritize: (1) Velocity tracking. We scale the velocity tracking reward by 20 and add it to the total reward. (2) Energy efficiency. We scale the energy penalty by 10 and add it to the total reward.

Implementation Details We trained our multi-directional walking policy using the Proximal Policy Optimization (PPO) algorithm [62], following the implementation of parallel reinforcement learning described in [63, 64]. Each robot was trained for 2.46×10^7 steps (a few minutes) before evaluation. We extended the open-source IsaacGymEnvs simulation environment [65] for our parallel training with 4096 environments per robot. Both our actor and critic networks employ three fully connected layers with dimensions of [512, 256, 128] and ELU activation functions. All training were performed on servers with NVIDIA RTX A6000, NVIDIA A100, and NVIDIA GeForce RTX 3090 GPUs. Comprehensive details on the specific training hyperparameters are in the supplementary material.

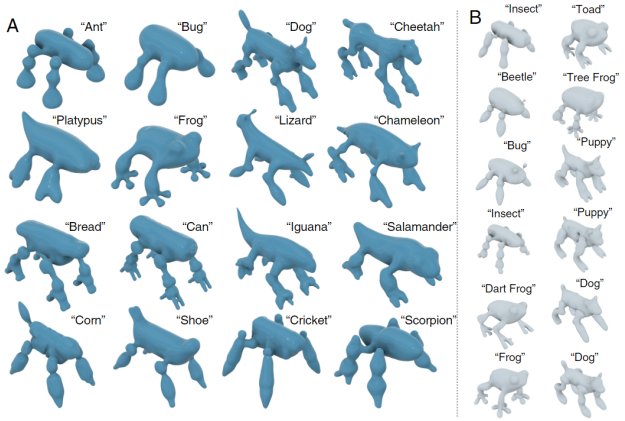

Fig. 5: **Generated Meshes and Corresponding User Descriptions.** (A) Sixteen robot mesh models generated from our structured prompt with diverse user descriptions. (B) We used the same or similar descriptions to generate four other morphology variants for bug, frog, and dog robots.

D. Physical Assembly

All robots were printed on the Creality CR-10 Smart Pro 3D printer due to its large printing bed. We use the Hiwonder HTD-45H High Voltage Serial Bus Servo to actuate our robots. We list detailed information on other electric components in supplementary material. The resulting robot assembly can be completed in **minutes** due to our modular design and careful considerations of electronic component placement and manufacturability.

IV. Experiment

In this section, we aim to evaluate Text2Robot from various aspects. First, we assess its ability to generate robot designs that align with diverse user-specified aesthetics. Second, we highlight the key advantages of integrating generative models into the automated robot design workflow. Third, we evaluate the performance of co-optimizing body morphology and control policy while prioritizing different performance metrics as specified by users. Finally, we demonstrate the real-world applicability of Text2Robot by fabricating and showcasing the physical robots.

A. Aesthetic Specification Matching

To assess the effectiveness of Text2Robot in generating robot designs that align with diverse user-specified aesthetics, we evaluated the initial stages of the framework using a variety of input text prompts. This evaluation aimed to demonstrate the capabilities of our chosen generative model in the context of robot design. We tested sixteen diverse prompts, encompassing both animal-inspired descriptions (e.g., “dog”, “frog”) and more creative concepts (e.g., “bread”, “can”, “shoe”). The qualitative results in Fig. 5A demonstrate that the generated designs can capture the essence of the input descriptions while adhering to the fundamental structure of a quadrupedal robot. We further investigated achieving subtle variations in generated morphologies by tweaking prompts, producing unique robot designs for specific species like “Bug”, “Dog”, and “Frog”, as shown in Fig. 5B. For each

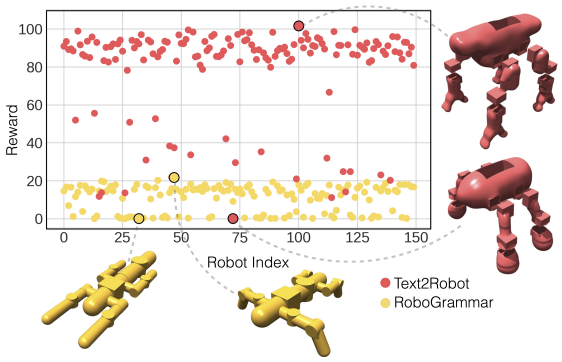

Fig. 6: Reward comparison of the Text2Robot and RoboGrammar robots and visualization of the best-performing and worst-performing robots.

of the above designs, we exported 30 robots in total by our geometric processing stage with various leg lengths and joint orientations for subsequent experiments. Text2Robot can generate a robot mesh within $1 \sim 2$ mins and convert it into a URDF model in 30 secs.

B. Advantages of Incorporating Text-to-3D Models

We hypothesize that one key advantage of integrating text-to-3D models is to provide a much stronger initialization for evolutionary design. Therefore, we compared with RoboGrammar [16] which enables a wide range of symmetrical robot designs based on simple primitive geometries as a baseline. To ensure a fair comparison, we filtered robots generated with their recursive graph grammar to only include quadrupedal robots with two joints per leg and with similar body lengths. We created URDF files for their robot and assigned the weights to match the weights of our robots. We then trained 150 RoboGrammar robots and 150 of our generated designs for the same amount of steps for a walking policy. Our 150 robots are randomly selected from 600 robots augmented from the sixteen prompts (Fig. 5A) and another four prompts in the bug species (Fig. 5B). Fig. 6 shows that our initialized robots outperformed the baseline initialized robots by a large margin.

We found most RoboGrammar robots suffer from unnatural placement and orientation of joints and links. Limbs on the same side are often too close together which compromises balance, and links protrude in the same direction as the joint axis, making the limb ineffective for locomotion. In contrast, Text2Robot leverages the knowledge of the physical world embedded within a text-to-3D generative model to produce quadrupedal robots with appropriate leg lengths, link and body proportions, and stable static initial postures.

C. Morphology and Walking Policy Co-optimization

Single Species Optimization We first co-optimized morphology and control with similar morphologies to "Bug", "Frog" and "Dog" as shown in Fig 5. Each species contains five variants of morphology from similar prompts. In each of the three trials, we used the 150 robots augmented from the five robots as the initial population in our EA, and all robots were optimized based on the general reward in Eq. 1. Fig. 7

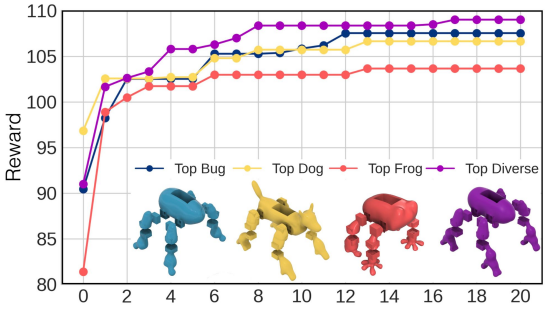

Fig. 7: **General Reward Optimization.** The reward of the best robot per generation and the morphology of the best robot in the last generation.

shows the emergence of successful locomotion within only a few generations, indicating the effectiveness of our method in generating walking quadrupedal robots. We continued to evolve the robots for 20 generations, and the performance was further improved with evolved morphologies.

Increasing Diversity We then investigated the effect of increasing diversity in the robot bank. We used the 600 robots augmented from the sixteen prompts and an additional four prompts in the bug species. The final selected robot achieved higher rewards compared to the robot optimized with similar morphologies (Fig. 7). Text2Robot enables higher-quality designs simply by expanding the diversity of text descriptions. This demonstrates the potential of our method to scale up and achieve better performance with more diverse and creative text prompts.

D. Co-optimization with Preferences

We evaluate Text2Robot to optimize robot designs according to user-supplied priorities: energy or velocity tracking. We used all robots with their augmented sets from Fig. 5. We adjusted the calculation of the fitness score by scaling the reward with additional amplified energy or velocity contribution. Fig. 8 shows the results of diverse species based on each preference under 50 generations of evolution. The results of a single species are listed in the supplementary material.

In the optimization for robots from single species or diverse species, our method is able to optimize the robot performance per generation while considering the performance priorities. The results show a strong correlation between the top-performing robot and the targeted performance criteria, and the other performance criteria, which were not being optimized, appeared more random and sporadic. These results demonstrate that our robot is optimized to meet user preferences. We found the robots selected from velocity optimization generally have longer and wider bodies, while the robots selected from energy optimization have lower body weights. We also find that the performance of the diverse robot bank is better than that of single-species banks, indicating the scalability of our method.

E. Co-optimization for Rough Terrain

We applied Text2Robot to rough terrain and observed that it effectively informed evolved robot morphologies with

higher-performing foot shapes. We also aim to highlight the impact of input texts on informing details of the robot designs, such as foot shapes, beyond just the large body components as shown above. We used the robots from the diverse bank in Fig. 5 and applied a VHACD decomposition to increase the simulated realism of foot contact. Following prior curriculum design [65], robots are trained to progress through increasingly challenging terrains (smooth slope, rough slope, stairs, discrete, and stepping stones).

Analysis of selected robot morphologies reveals a correlation between foot shape and terrain type, as illustrated in Fig. 9. Arched feet, favored in flat terrain trials, provide enhanced stability, speed, and efficient energy transfer due to their curvature. This advantage, however, hinges on predictable surface contact dynamics. We observed that the large foot dimensions of arched feet increase the risk of snagging on uneven terrain. Conversely, simple rounded feet, chosen for rough terrain, demonstrated superior adaptability to unpredictable surfaces, promoting stability and balance.

F. Physical Walking Robot

We selected the highest-performing "Bug" and "Frog" from the single-species optimization and the two robots from the diverse bank optimized for velocity tracking and energy efficiency to demonstrate our ability to fabricate the generated designs. Each robot required approximately a day to manufacture, but assembly (Fig. 10) was completed within minutes. We show a primitive Sim2Real transfer by simply playing the trained locomotion in simulation and directly executing the joint positions on the real robot. As shown in Fig. 10, our real robots successfully transfer the walking policy learned in simulation to the real world and achieve sufficient performance in locomotion and speed. This further

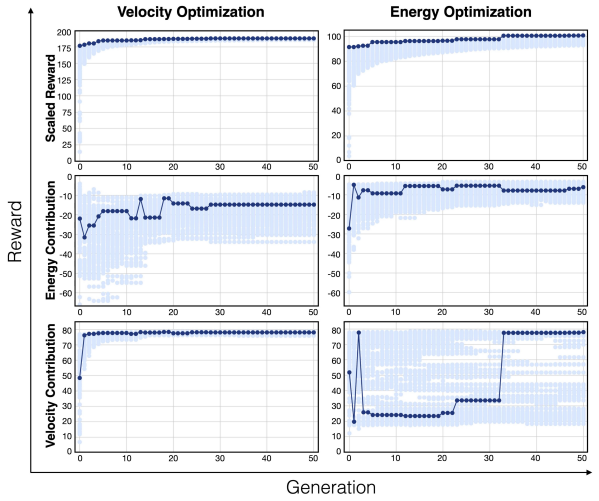

Fig. 8: **Optimization Based on Preference with Diverse Robot Species.** The scaled reward represents the fitness score evolutionary loop. It consists of the original reward received by the robots during inner loop training and the scaled contribution based on energy or velocity, depending on the user's preference. The best-performing robots are marked with dark blue, and the rest of the top 100 robots are marked with light blue in the figure.

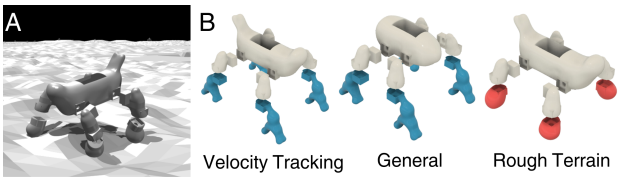

Fig. 9: **Rough Terrain Optimization.** (A) The selected robot traversing rough terrain. (B) Robots optimized for flat terrain (left and middle) evolve to have larger arcs for feet, while the robot evolved for rough terrain (right) has smaller, simple rounded feet.

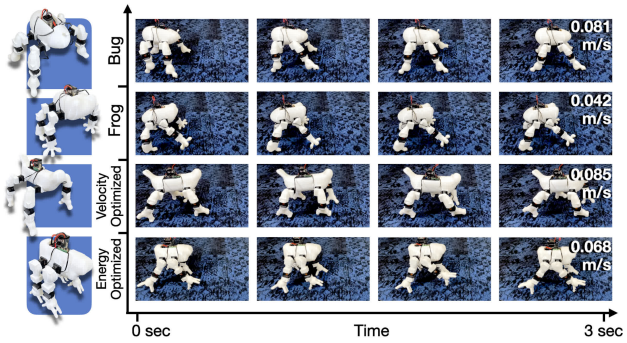

Fig. 10: **Real Robot Performance.** We play the best robot policy in simulation with a goal speed of $0.1\ \mathrm{m/s}$ in a straight direction, and the real robot executes the same position command in the real world.

validates the practicability and robustness of our design from our Text2Robot pipeline. More examples can be found in our supplementary videos.

V. Conclusion, Limitation and Future Work

We introduce Text2Robot, which generates a physical quadrupedal walking robot from text prompts to match user-specified aesthetic and performance preferences. Text2Robot leverages generative models for stronger initialization than traditional methods, while converting visual meshes to movable robots with considerations in electronics and real-world manufacturability. Both simulated and physical experiments show Text2Robot's ability to co-optimize morphology and control to produce physically functional machines.

Limitations and Future Work Text2Robot presents several opportunities for improvement in future research. While our current focus is on quadrupedal robots which are already challenging to design for humans, future work can extend the scope to robots with varying numbers of joints or other types of electromechanical machines. One possible solution is to combine the strong initialization of Text2Robot with existing robot design methods such as RoboGrammer. Additionally, the current framework still requires manual assembly. Integrating our method with automated assembly algorithms to construct physical robots would be a significant advancement. Furthermore, our text prompts remain fixed once optimization begins. Exploring a feedback mechanism to refine mesh generation from the text-to-3D model based on reward signals could offer greater flexibility.

References

[1] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, no. 6799, pp. 974–978, 2000.

[2] 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*, vol. 19, no. 4, pp. 703–719, 2003.

[3] D. Matthews, A. Spielberg, D. Rus, S. Kriegman, and J. Bongard, "Efficient automatic design of robots," *Proceedings of the National Academy of Sciences*, vol. 120, no. 41, p. e2305180120, 2023.

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

[5] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of dynamic gaits for a quadrupedal robot," in *2013 IEEE international conference on Robotics and automation*. IEEE, 2013, pp. 3287–3292.

[6] J. Carpentier and P.-B. Wieber, "Recent progress in legged robots locomotion control," *Current Robotics Reports*, vol. 2, no. 3, pp. 231–238, 2021.

[7] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1154–1171, 2021.

[8] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[9] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 93–100.

[10] S. Wang, W. Chaovalitwongse, and R. Babuska, "Machine learning algorithms in bipedal robot control," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 5, pp. 728–743, 2012.

[11] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[12] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[13] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.

[14] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[15] L. Strgar, D. Matthews, T. Hummer, and S. Kriegman, "Evolution and learning in differentiable robots," *arXiv preprint arXiv:2405.14712*, 2024.

[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)*, vol. 39, no. 6, pp. 1–16, 2020.

[17] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei, "Embodied intelligence via learning and evolution," *Nature Communications*, vol. 12, no. 1, p. 5721, 2021. [Online]. Available: https://doi.org/10.1038/s41467-021-25874-z

[18] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding," *SIGEVOLUTION*, vol. 7, pp. 11–23, 2014.

[19] N. Cheney, J. Bongard, and H. Lipson, "Evolving soft robots in tight spaces," in *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, 2015, pp. 935–942.

[20] T. Bäck and H. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[21] D. Thierens, "Scalability problems of simple genetic algorithms," *Evolutionary Computation*, vol. 7, no. 4, pp. 331–352, 1999.

[22] N. Cheney, V. Sunspiral, J. Bongard, and H. Lipson, "On the difficulty of co-optimizing morphology and control in evolved virtual creatures," *Artificial Life Conference Proceedings*, vol. 13, pp. 226–233, 2016.

[23] L. Regenwetter, A. H. Nobari, and F. Ahmed, "Deep generative models in engineering design: A review," *Journal of Mechanical Design*, vol. 144, no. 7, p. 071704, 2022.

[24] T. O. Akande, O. O. Alabi, and J. B. Oyinloye, "A review of generative models for 3d vehicle wheel generation and synthesis," *Journal of Computing Theories and Applications*, vol. 1, no. 4, pp. 368–385, 2024.

[25] L. Makatura, M. Foshey, B. Wang, F. HähnLein, P. Ma, B. Deng, M. Tjandrasuwita, A. Spielberg, C. E. Owens, P. Y. Chen, *et al.*, "How can large language models help humans in design and manufacturing?" *arXiv preprint arXiv:2307.14377*, 2023.

[26] S. Feuerriegel, J. Hartmann, C. Janiesch, and P. Zschech, "Generative ai," *Business & Information Systems Engineering*, vol. 66, no. 1, pp. 111–126, 2024.

[27] R. H. Kazi, T. Grossman, H. Cheong, A. Hashemi, and G. W. Fitzmaurice, "Dreamsketch: Early stage 3d design explorations with sketching and generative design." in *UIST*, vol. 14, 2017, pp. 401–414.

[28] B. Sanchez-Lengeling and A. Aspuru-Guzik, "Inverse

molecular design using machine learning: Generative models for matter engineering," *Science*, vol. 361, no. 6400, pp. 360–365, 2018.

[29] F. Buonamici, M. Carfagni, R. Furferi, Y. Volpe, L. Governi, *et al.*, "Generative design: an explorative study," *Computer-Aided Design and Applications*, vol. 18, no. 1, pp. 144–155, 2020.

[30] V. Liu, J. Vermeulen, G. Fitzmaurice, and J. Matejka, "3dall-e: Integrating text-to-image ai in 3d design workflows," *ACM Transactions on Graphics*, vol. 41, pp. 1955–1977, 2023.

[31] Z. Liu, D. Zhu, S. P. Rodrigues, K.-T. Lee, and W. Cai, "Generative model for the inverse design of metasurfaces," *Nano letters*, vol. 18, no. 10, pp. 6570–6576, 2018.

[32] S. Oh, Y. Jung, S. Kim, I. Lee, and N. Kang, "Deep generative design: Integration of topology optimization and generative models," *Journal of Mechanical Design*, vol. 141, no. 11, p. 111405, 2019.

[33] S. Luo, Y. Su, X. Peng, S. Wang, J. Peng, and J. Ma, "Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9754–9767, 2022.

[34] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–39, 2023.

[35] R. Wu, C. Xiao, and C. Zheng, "Deepcad: A deep generative network for computer-aided design models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6772–6782.

[36] Y. Siddiqui, A. Alliegro, A. Artemov, T. Tommasi, D. Sirigatti, V. Rosov, A. Dai, and M. Nießner, "Meshgpt: Generating triangle meshes with decoder-only transformers," *arXiv preprint arXiv:2311.15475*, 2023.

[37] A. Nordmann, N. Hochgeschwender, and S. Wrede, "A survey on domain-specific languages in robotics," in *International conference on simulation, modeling, and programming for autonomous robots*. Springer, 2014, pp. 195–206.

[38] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[39] H. Abdullah and J. Kamara, "Parametric design procedures: a new approach to generative-form in the conceptual design phase," in *AEI 2013: Building Solutions for Architectural Engineering*, 2013, pp. 334–343.

[40] G. S. Hornby and J. B. Pollack, "The advantages of generative grammatical encodings for physical design," in *Proceedings of the 2001 congress on evolutionary computation (ieee cat. no. 01th8546)*, vol. 1. IEEE, 2001, pp. 600–607.

[41] J. Liu, X. Huang, T. Huang, L. Chen, Y. Hou, S. Tang, Z. Liu, W. Ouyang, W. Zuo, J. Jiang, *et al.*, "A comprehensive survey on 3d content generation," *arXiv preprint arXiv:2402.01166*, 2024.

[42] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-to-3d content creation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 300–309.

[43] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv preprint arXiv:2209.14988*, 2022.

[44] S. Babu, R. Liu, A. Zhou, M. Maire, G. Shakhnarovich, and R. Hanocka, "Hyperfields: Towards zero-shot generation of nerfs from text," *arXiv preprint arXiv:2310.17075*, 2023.

[45] K. Chen, C. B. Choy, M. Savva, A. X. Chang, T. Funkhouser, and S. Savarese, "Text2shape: Generating shapes from natural language by learning joint embeddings," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*. Springer, 2019, pp. 100–116.

[46] R. Chen, Y. Chen, N. Jiao, and K. Jia, "Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 246–22 256.

[47] 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*, vol. 116, no. 3, pp. 344–356, 1994.

[48] C. Paul and J. Bongard, "The road less travelled: morphology in the optimization of biped robot locomotion," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 1, 2001, pp. 226–232 vol.1.

[49] T. Geijtenbeek, M. Van De Panne, and A. F. Van Der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, pp. 1–11, 2013.

[50] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Computational co-optimization of design parameters and motion trajectories for robotic systems," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1521–1536, 2018.

[51] 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.

[52] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly learning to construct and control agents using deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 9798–9805.

[53] 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.

[54] 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," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[55] H. Zhanpeng and C. Matei, "Morph: Design co-optimization with reinforcement learning via a differentiable hardware model proxy," *IEEE Robotics and Automation*, 2024.

[56] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. E. Eiben, "Evolutionary robotics: what, why, and where to," *Frontiers in Robotics and AI*, vol. 2, p. 4, 2015.

[57] R. J. Alattas, S. Patel, and T. M. Sobh, "Evolutionary modular robotics: Survey and analysis," *Journal of Intelligent & Robotic Systems*, vol. 95, pp. 815–828, 2019.

[58] J. Lehman, J. Clune, D. Misevic, C. Ofria, K. O. Ellefsen, J.-B. Mouret, and A. Bernatskiy, "The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities," *Artificial Life*, vol. 26, pp. 274–306, 2020.

[59] Meshy. [Online]. Available: https://www.meshy.ai/

[60] Autodesk fusion 360. [Online]. Available: https://www.autodesk.com/products/fusion-360

[61] T. Kitamura, "Fusion2urdf," https://github.com/syuntoku14/fusion2urdf, 2020.

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

[63] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[64] D. Makoviichuk and V. Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," https://github.com/Denys88/rl_games, May 2021.

[65] 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.

SUPPLEMENTARY MATERIAL

A. Single Species Optimization

We show performance optimization results for single-species robot banks, 'Bug', 'Frog', and 'Dog,' and compare performance to that of the diverse bank. As in our diverse bank performance optimization experiment, we adjusted the fitness score calculation with an additional velocity reward or energy cost through 50 generations of evolution to demonstrate the effect of an input performance preference on final designs. As shown in Fig. 12, our results show a large correlation between the selected bot and the prioritized performance criteria. Fig. 11 shows the visualization of the selected robots, their average final rewards, and physical characteristics.

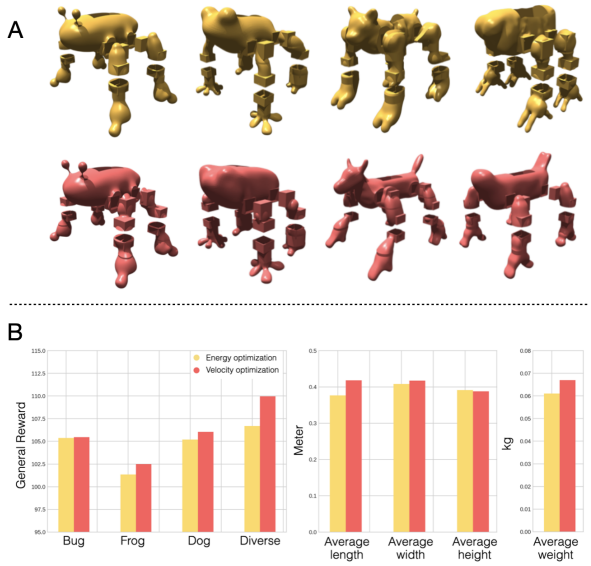

Fig. 11: (**A**)The morphology of the best robot in the last generation from the eight experiments with energy or velocity prioritize. Yellow color represents the robots that prioritize energy contribution and red color represents the robots that prioritize velocity contributions. (**B**) The unscaled reward of the eight best robots. The average length, width, height and weight of the eight robots.

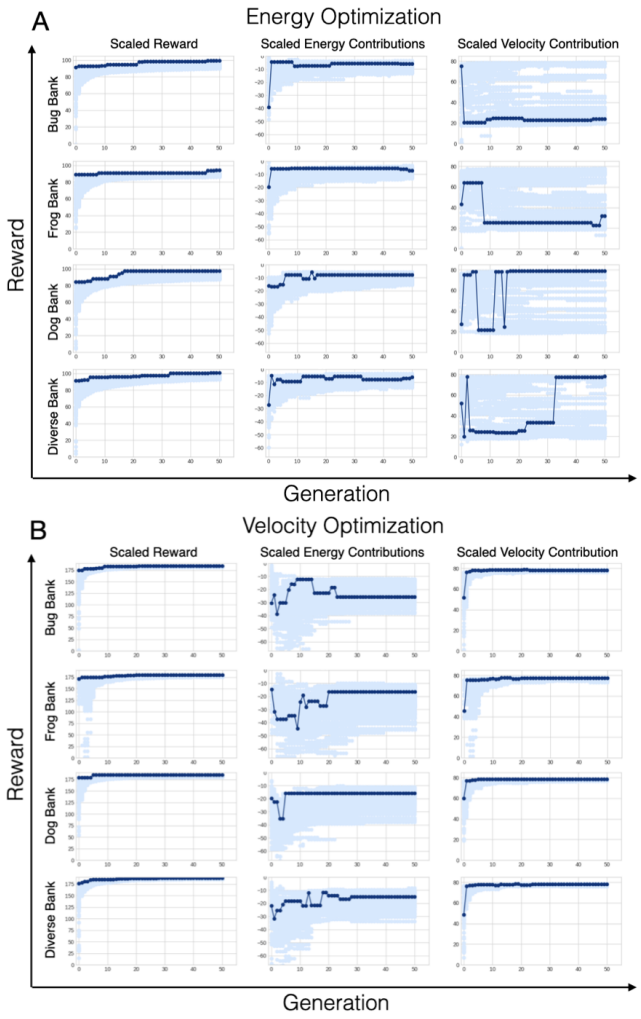

Fig. 12: The scaled reward is used as the fitness metric in the EA loop. It consists of the original reward received by the robots during inner loop training summed with the requested priority: (**A**) Scaled energy contribution, (**B**) Scaled velocity contribution. The best-performing robots are marked with dark blue, and the rest of the top 100 robots are marked with light blue in the figure.

B. Reinforcement Learning Details

We show the detailed training parameters of the inner reinforcement learning loop in Tab. I , and the definition of symbols and baseline reward in Tab. II and Tab. III.

C. Full Hardware Specifications

We manufacture our robots using Creality CR-10 Smart Pro 3D printers. Robots are assembled using the 3D printed parts, as well as various electronic components which are easily inserted in the design. A comprehensive list of materials used in the construction of our robot is outlined in Table IV.

Hyper-parameters	Values
Dense network shape	[512, 256, 128]
Dense network activation	elu
Discount factor	0.99
GAE discount factor	0.95
PPO loss clip range	0.2
Entropy coefficient	0.001
Learning rate α	adaptive
Batch size	98304 (4096x24)
Mini-batch size	16384 (4096x4)
Mini epochs	5
Critic loss coefficient	2
KL-divergence threshold	0.008

TABLE I: PPO hyper parameters

Base linear velocity	$\mathbf{v}$
Base angular velocity	ω
Commanded base linear velocity	$\mathbf{v}^*$
Commanded base angular velocity	ω^*
Joint positions	$\mathbf{q}$
Joint velocities	$\dot{\mathbf{q}}$
Joint accelerations	$\ddot{\mathbf{q}}$
Target joint positions	$\mathbf{q}^*$
Joint torques	τ
Number of joints	n
Number of feet	n_f
Feet air time	t_{air}
Feet stance time	t_{stance}
Base gravity	$\mathbf{g}_b$
Environment time step	dt

TABLE II: Definition of symbols.

baseline reward terms	definition	weight $[*dt]$
Linear velocity tracking	$e^{-0.25\left\|\mathbf{v}^*_{xy}-\mathbf{v}_{xy}\right\|^2}$	1
Angular velocity tracking	$e^{-0.25\left\|\omega^*_z-\omega_z\right\|^2}$	0.5
Linear velocity penalty	v_z^2	-4
Angular velocity penalty	$\left\|\omega_{xy}\right\|^2$	-0.05
Joint acceleration penalty	$\left\|\ddot{\mathbf{q}}\right\|^2$	-5^{-7}
Joint torque penalty	$\left\|\tau\right\|^2$	-2^{-5}
Action rate penalty	$\left\|\dot{\mathbf{a}}\right\|^2$	-5^{-5}
orientation	$\left\|\mathbf{g}_{b,xy}\right\|^2$	-0.5
Feet air time	$\sum_{k=0}^{n_f}(t_{air,k}-0.5)$	0.1
Feet stance time	$\sum_{k=0}^{n_f}(t_{stance,k}-0.5)$	0.1

TABLE III: **Definition of baseline reward ($r_{\textbf{baseline}}$) terms.** The baseline reward contains base velocity and orientation tracking terms, action rate penalty, and joint torque penalty; the air time and stance time reward encourages longer air time and stand time to promote a more natural and fluid walking gait.

Component	Model	Quantity per robot
Microcontroller	Raspberry Pi 4 Model B	1
Battery	Povway 5200 mA Lipo $3S$ $11.1V$ $50C$	1
Servo Motor	Hiwonder $HTD-45H$ High Voltage Serial Bus Servo 45 KG	8
DC to DC Power Converter	DROK $10A$ Synchronous Step-Down Voltage Regulator DC-DC $4-30V$ to $1.2-30V$ $12V$	1
PLA Filament	Ender PLA 3D Printer PLA Filament 1.75 mm $1KG$ (2.2 lbs) Spool PLA White	1
Motor Controller	Hiwonder TTL / USB Debugging Board	1

TABLE IV: **List of hardware component.**