TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Chen, G., Shi, M., Xing, Y., Popović, M., Alonso-Mora, J., Zhang, L., & Pang, J. (2026). Automatic personalized limbed robot design from media inputs. *npj Robotics*, *4*(1), Article 42. https://doi.org/10.1038/s44182-026-00101-3

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

https://doi.org/10.1038/s44182-026-00101-3

Automatic personalized limbed robot design from media inputs

Check for updates

Gang Chen[1,2,6] ✉, Moji Shi[1,3,6], Yu Xing[1], Marija Popović[3], Javier Alonso-Mora[2], Lei Zhang[4,5] & Jiangmiao Pang[1] ✉

Designing limbed robots is a complex, multidisciplinary task that typically requires substantial effort from experienced engineers. In this paper, we present a novel automatic robot design framework based on Decomposition-Optimization-Assembling (DOA) to address this challenge. Our framework enables non-experts to create personalized limbed robot designs from media inputs, such as text and images, within minutes to a few hours. Our system leverages recent advances in generative AI and 3D printing to produce designs that match the descriptions provided in the input media. The output consists of selected motors and 3D-printable mechanical components that can be assembled into a limbed robot. To handle the large design space and intricate details in fabrication and assembly, we formulate and solve a series of optimization problems involving actuators, geometry, and structural density. We validate the proposed system by designing and fabricating a Centaur robot based on an image input. Furthermore, we demonstrate the system's versatility and effectiveness through the generation of a wide variety of limbed robot designs.

Limbed robots have drawn significant attention in recent years. These robots include popular designs such as quadrupeds[1–4] and humanoids[5,6], as well as other bio-inspired designs like rat[7], salamander[8], spider[9], and kangaroo[10] robots. Numerous designs have been developed to support diverse applications. However, designing limbed robots remains a complex, time-intensive, and multidisciplinary challenge that demands expertise in fields such as mechanical engineering, electrical engineering, industrial design, and materials science, which makes it challenging for non-experts to create functional limbed robots.

Recent advancements in generative artificial intelligence have enabled the creation of 3D models of any object from media inputs such as text and images[11–16], which are more accessible to non-experts and convey personalized design information. These advancements allow individuals without expertise in 3D modeling to generate high-quality 3D models by providing a text description or an image. This development prompts the question: *Can the robot design process be made as simple as providing media inputs to a computer program?* Specifically, an automatic robot design system capable of generating limbed robot designs from media inputs like text and images would make the design process accessible to non-experts, empowering everyone to create personalized limbed robots and opening new possibilities in education and entertainment with robotics.

The automatic design of robots, situated within the domain of computational design, remains in its early stages. Existing methods typically fall into two categories:

The first category focuses on improving key parameters of a robot to achieve a target motion or maximize locomotion performance. Early works in this category aim at realizing a desired motion trajectory or space for joints or end-effectors in a mechanical structure. These methods typically adjust linkage topology[17,18], actuator placement[19,20], elasticity[21] and mechanical component selection and arrangement[22–24] to produce the desired motion trajectory or space. A recent focus has been the co-optimization of morphology parameters and control to enhance task performance, especially in locomotion. This includes works on quadruped robots[25–28], general limbed robots[29,30], and soft robots[31], where parameters such as limb length and joint configuration are optimized to reach the best performance when a reinforcement learning-based or model predictive controller is used. In addition, David et al.[32] optimized the placement of voids and muscle patches in a soft robot to enable motion driven by a pneumatic actuator. While these methods can yield effective designs for specific tasks, their design space is limited to one or several key parameters of a manually designed robot or a simple mechanical structure, limiting their ability to explore new and personalized robot designs.

[1]Shanghai AI Laboratory, Shanghai, China. [2]Department of Cognitive Robotics, Delft University of Technology, Delft, The Netherlands. [3]MAVLab, Delft University of Technology, Delft, The Netherlands. [4]Meta Robotics Institute, Shanghai Jiao Tong University, Shanghai, China. [5]State Key Laboratory of Mechanical System and Vibration, Shanghai Jiao Tong University, Shanghai, China. [6]These authors contributed equally: Gang Chen, Moji Shi. ✉e-mail: g.chen-5@tudelft.nl; pangjiangmiao@pjlab.org.cn

The second category emphasizes the composition and arrangement of predesigned modular components to realize personalized robot designs. These approaches often provide a library of actuators and manually designed mechanical structures, and use optimization or user input to configure them into functional robots. For example, Zhao et al.[33] used graph grammars to represent robot designs and employed heuristic search for optimal component arrangements. Megaro et al.[34] developed a user interface (UI) that allows users to edit the skeleton structure and high-level motion of a limbed robot. The robot is then designed using a composition of off-the-shelf servo motors and predefined mechanical components. Later, Desai et al.[35] created an expanded library with mounting brackets. Users can drag and drop components from the library to complete a robot design. A search-based semi-automatic completion mode was also introduced to enable design based on functional characteristics. Schulz et al.[36] further increased design flexibility with print-and-fold components and a user guidance system. In addition, Desai et al.[37] considered emotional expressions in robot behaviors, and Feng et al.[38] focused on skinned quadrupeds. The work of Du et al.[39] developed a system tailored to the design of multicopters, providing a database of motors, body frames, and carbon fiber rods for user selection. Some key component parameters were further optimized to ensure a stable flight. Although these methods facilitate user-driven customization, they are inherently limited by the scope and granularity of predefined mechanical components, which require significant manual design effort.

Our approach aims at generating personalized robot designs, including mechanical structures and actuator systems, directly from media inputs. These media inputs, while abstract and lacking technical design details, emphasize the robot's artistic and morphological aspects. Their expressiveness and diversity stem from the ease with which individuals can describe or visually depict their envisioned robots through language or imagery. Despite advances in the aforementioned works, designing robots directly from media inputs remains an underexplored and challenging problem. Specifically, two major challenges must be addressed:

The first challenge lies in the large design space. Existing computational design systems typically restrict the design space to a few key parameters of manually designed mechanical components (e.g., lengths of legs[25–28] or rods[39]), the selection and placement of actuators or joints[29,30,34,40], or the arrangement of predefined modular components in a "composition" paradigm[33,36–39,41,42]. However, these manually designed mechanical components heavily limit the diversity of the resulting robot designs. To design a personalized robot that resembles the descriptions in the media inputs, existing works still require experienced designers to manually build the Computer-Aided Design (CAD) model of the entire robot or its components at the beginning. The main challenge lies in determining actuation and geometry under fabrication constraints, a process that remains labor-intensive and difficult to scale. We aim to fully automate the robot design process from media inputs to a functional robot, thereby significantly reducing manual labor and design time. Consequently, the geometry shape and material of the mechanical components and the actuator system should all be designed to generate a robot that aligns with the descriptions in the media inputs, which introduces a much larger design space.

The second challenge arises from the intricate fabrication and assembly details. Fabrication details are typically overlooked in existing computational design systems. To ensure the feasibility of a design for a functional robot, expert designers must consider many intricate details, such as the mechanical properties of each component, processability, interference between components during motion, and assembly interference. Existing methods require designers to manually build the CAD model of the robot[25–28,43] or its modular components[33,36–39,41,42] to address these details. However, since we design our system to automatically generate diverse mechanical structures, it must also address these fabrication and assembly details for each robot design. This requirement also poses a significant new challenge for the system.

This paper proposes a novel "decomposition-optimization-assembling" (DOA) based automatic robot design system to address the aforementioned challenges. The system automatically generates fabricable limbed robot designs from unstructured meshes produced by text/image-to-3D generative models, advancing toward text/image-to-robot design.

An overview of the system is shown in Fig. 1. Given a text or image input, a preprocessing step, composed of a text/image to 3D model generation tool[44] and pose estimation model[45], is first used to generate an initial mesh that resembles the descriptions in the input and the initial joint configuration. The joint configuration specifies joint positions, DoF (currently one or two), and motion axes, with all joints modeled as revolute. We adopt Bite[45] to estimate initial joint configurations for quadruped-type animals. For cases where joint configurations cannot be reliably inferred (e.g., the Lynel character in our experiments), we provide a UI that enables interactive joint addition and modification. Subsequently, the system automatically decomposes the initial mesh into initial mechanical components, i.e., links such as the thighs, shanks, arms, and bodies of a limbed robot. Three optimization procedures are then performed automatically: actuator optimization, geometry optimization, and density optimization.

The actuator optimization aims to select the proper actuators and optimize their configuration. A library of off-the-shelf motors with varying sizes and torques is used to define the search space for actuator selection. The goal is to optimize the type and pose of each motor to (1) keep close to the initial joint position and align the motion axes, (2) bring less change to the original shape of the robot, (3) provide enough but not excessive torque to actuate the links, and (4) avoid interference between each other. These goals are encoded as kinematic alignment cost, mesh coverage cost, torque constraints, and inference constraints, respectively, and we formulate a mixed-integer and non-linear optimization problem based on them and solve by evolutionary algorithm. More details are provided in the "Methods" section.

The geometry optimization aims to adjust the mesh generated from the preprocessing step so that the motors can be mounted on the links and the interference between the links during motion can be avoided, while the shape of the robot is kept close to the initial mesh. Considering the high complexity of this problem, we first voxelize the initial mesh and turn the problem into three sub-problems: (1) mesh expansion, (2) key voxel searching, and (3) motion interference removal. These sub-problems reallocate the free or occupied status of the voxels for each link to meet the aforementioned goals. Algorithms are proposed to solve these sub-problems in the "Methods" section. The mesh is then reconstructed from the voxels using the marching cubes algorithm[46].

At this step, the system can already output a rough Unified Robot Description Format (URDF) file of the robot, which can be used in simulations. However, the robot is not feasible for real-world fabrication, as many fabrication details, such as the materials and mechanical properties of the links and their assembly methods, have not yet been specified.

Considering recent advancements in 3D printing and metamaterials, we propose optimizing link density to ensure they are suitable for 3D printing while achieving the desired mechanical properties. Since the density of the meta-material is the primary variable in this process, we call this step density optimization. The mechanical properties are evaluated by Finite Element Analysis (FEA) in a programmatic approach. For each link, the density is optimized to balance the weight and Von Mises Stress. The mesh is then filled with a six-fold plate lattice structure[47], with the skin preserved.

The final step of our system addresses the assembly details. Since screws are not compatible with our top-down decomposition design and 3D printing approach, we propose using a mortise-and-tenon joint to connect the motors and links. We search for the optimal orientation of the mortise-and-tenon joint to minimize interference during assembly. Further assembling interference is tackled by mesh subtraction operations.

The system's output is a list of motors and a set of 3D-printable mechanical components that can be assembled into a limbed robot. Details of the system are described in the "Methods" section. Our system addresses the two aforementioned challenges as follows. First, starting from a new "decomposition" based automatic design paradigm, our system uses text/image to 3D model to generate an initial mesh, decomposes the mesh into different mechanical components, and then optimizes them. Therefore, our system is not fixed to a set of predefined modular components. Instead, it

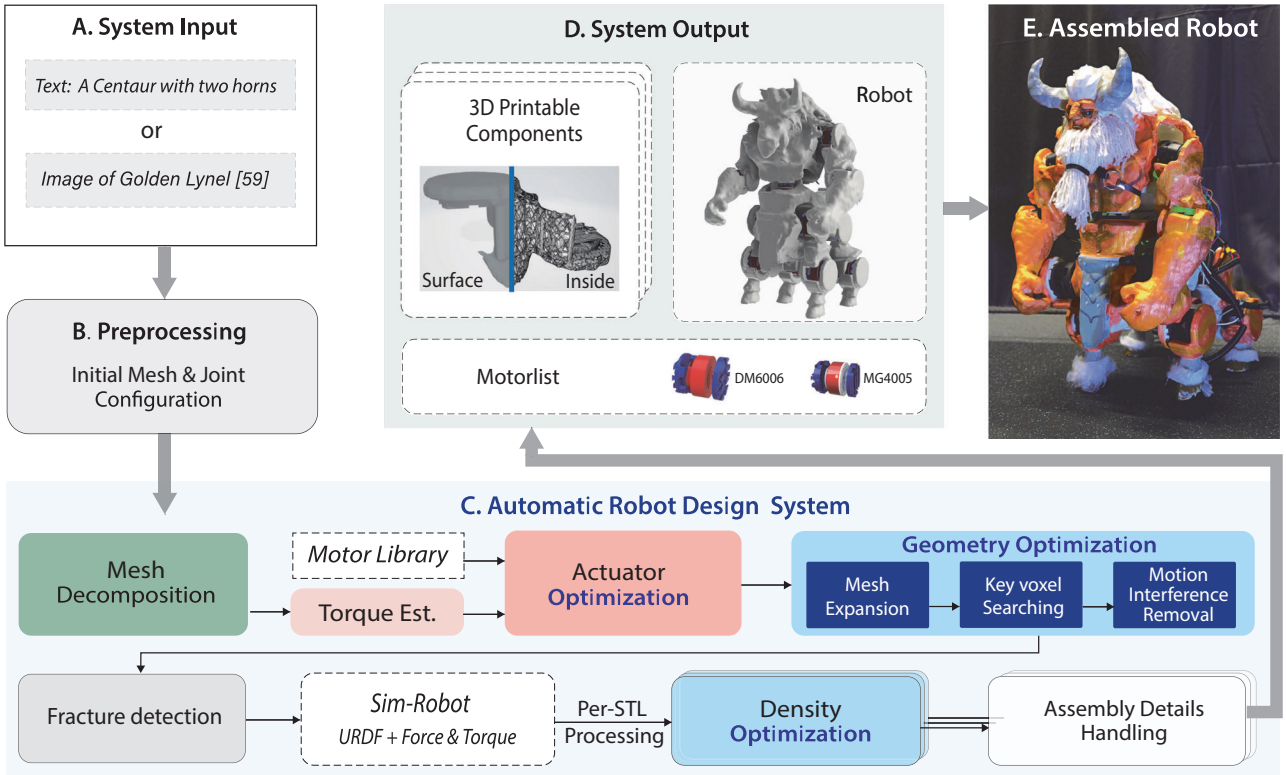

Fig. 1 | Proposed system overview. A System input, which can be a text or an image (the used image can be found in ref. 56. **B** Preprocessing step that generates the mesh and joints of the text or image. **C** Automatic limbed robot design system. Black arrows indicate the system's operational flow. Solid boxes represent subprograms within the system, while dashed boxes represent data. **D** System output composed of 3D printable mechanical components and a motor list. **E** Assembled robot, which is a Centaur named "Lynel" designed from the image input. "Lynel" is a character from the video game "The Legend of Zelda: Breath of the Wild".

can generate diverse-shaped mechanical components that match the descriptions in the media inputs.

Second, leveraging recent advancements in 3D printing and metamaterials, our system can handle fabrication details such as component materials, mechanical properties, and processability. In addition, interferences between components during assembly and motion are considered in the geometry optimization and assembling detail handling steps using voxel- and mesh-based operations, respectively. The final output consists of 3D-printable mesh files that can be assembled directly into a functional, limbed robot.

Using our system, non-experts can create personalized, diverse-legged robot designs in several minutes to a few hours without needing to handle the technical details of the design process or intricate fabrication. The system keeps production costs low while automating the transformation of text and images into functional, fabricable robots. This streamlines the transition from conceptual modeling to physically realizable robotic systems.

Results

This section presents the results of our system in three parts. First, we present the design result of a Centaur generated from an image input. The designed robot is fabricated, assembled, and tested in the real world. Next, we show quantitative automatic design results with the Stanford Dogs dataset[48]. Finally, additional results with different input modalities, including text, image, and 3D model, are presented.

Design result of a Centaur

This subsection presents the result of a Centaur designed from an image input. The input image is shown in Fig. 1(**A**), a character named "Lynel" from the video game *The Legend of Zelda: Breath of the Wild*. The initial mesh is generated from the image using Meshy[44]. Since this character has a non-standard joint configuration (unlike the quadrupeds discussed in the next subsection), we specified the initial joint positions, DoFs and motion axes using our UI during preprocessing. Subsequently, the mesh decomposition and the remaining steps of the system are automatically performed to produce the final design result. Recent work[49] has introduced a method capable of generating joints for such characters, which could be integrated into our system in the future.

The joints include two DOFs for each hip and shoulder, one DOF for each knee and elbow, and one for the waist, totaling 19 DOFs. We set the rotational range of each joint to ± 1 rad and the voxel resolution to 0.5 cm. The automatic design process takes about 8 h on a computer equipped with an AMD Ryzen 9 5900X CPU. Intermediate results of each step are shown in the "Methods" section. We show the final robot designed and assembled in Fig. 2.

Figure 2(A) shows the real 3D-printed mechanical components of the system's mesh output. These components include the lower and upper bodies, thighs, shanks, forearms, and upper arms. Each component is filled with a six-fold plate[47] lattice structure with the optimized density. The inner structure of one component is illustrated in the bottom center of (A). Each component is also integrated with a dovetail connector in the design to interface with the motors. We used a Bamboo Lab P1s printer and low-cost Polylactic acid (PLA) material to print the components. All mechanical components are directly printed without further modification.

The selected motors and the corresponding dovetail connectors on the motor side are shown in Fig. 2(B). Twelve DM6006 motors and seven MG4005 motors are selected from a motor library consisting of five motors with different sizes and torques. Specifications of the motors are provided in Table S1 of the supplementary material. In the assembly process, the dovetail connectors on the motor side are first attached to the motors using screws compatible with each motor. Then the motors with dovetail connectors and the mechanical components are assembled together using the optimized dovetail direction without the need for additional screws. The assembled robot is shown in Fig. 2(E). To control the robot, we use an Up Board computing board with an SPI to CAN converter to calculate and send

Fig. 2 | Result of a Centaur. A 3D-printed mechanical component of the robot. The initial joint configurations are specified via our UI during preprocessing. Mesh decomposition and the subsequent steps of our system (Fig. 1C) are performed automatically. **B** Selected motors in the actuator optimization step and the dovetail connectors. **A**, **B** constitute the result of the automatic robot design system. The original input image is shown in Fig. 1(**A**). **C** Decorating tools to color the robot and add more facial details. **D** Control board of the robot. **E** Assembled robot with no decoration. **F** Assembled robot with decorations in motion.

control commands to the motors. The lower body, with twelve motors, is controlled using Champ[50], while the upper body, with seven motors, is operated via a joystick.

We additionally apply color and decorations to enhance the robot's resemblance to the character. The final robot is shown in Fig. 2(F). The total weight is about 15 kg. It can walk and move joints to realize various poses. The first two rows depict the snapshots of the robot in various poses during motion. The third row shows a profile view to better illustrate the lower-body gait and the motion. Additional motion results are available in the supplementary video.

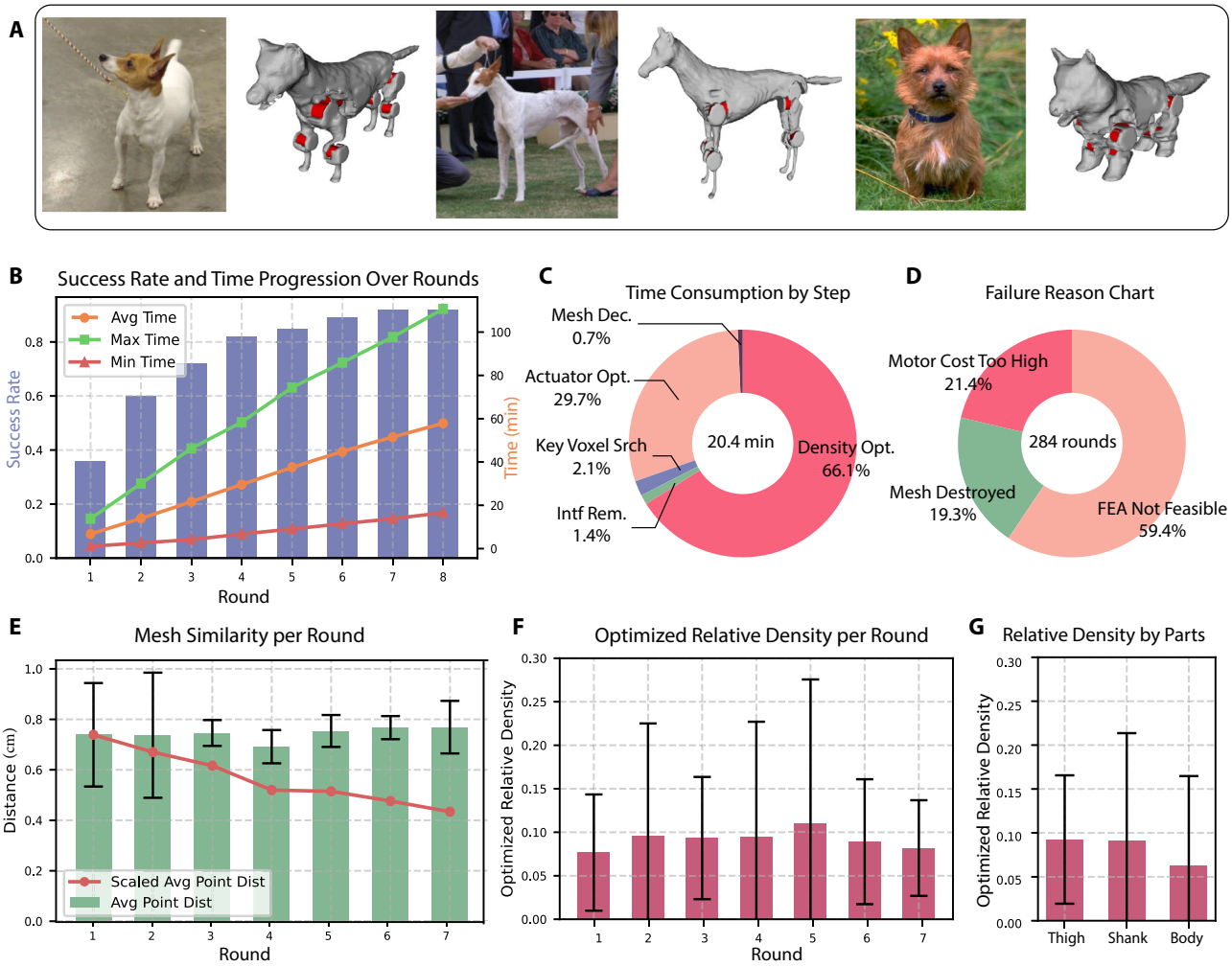

Fig. 3 | Results with the Stanford Dogs dataset[48]. A presents the resulting URDFs of three different dog pictures. **B** illustrates a plot of the conditioned success rate and time consumption increasing with the number of design rounds. **C**, **D** further reveal the distribution of the time consumption and the causes of failure. **E** shows the similarity between the mesh of the final components and the initial mesh in the preprocessing step. **F**, **G** present the average relative density categorized by the design rounds and the link types, respectively.

Quantitative results with quadrupeds

This subsection presents the quantitative results with image inputs from the Stanford Dogs dataset[48]. One hundred randomly selected images from the dataset are used as input to our system. Since dogs have standard joint templates, a 3D pose estimation model Bite[45] is used to generate both the initial mesh and joint positions. The neutral pose mesh and joint positions are used to keep the limbs straight and avoid intersections. The joint degrees of freedom follow the 12-DOF configuration of a standard quadruped robot[1–4]. The motors used in the motor library are listed in the supplementary material. These motors are popular off-the-shelf modules available on the market. Since the actual sizes of the dogs are not known from the image, we normalize the width (usually from the leftmost point on the left shoulder to the rightmost point on the right shoulder) to 20 cm. If the design fails, the mesh is scaled up by 10%, and a new design round begins, continuing until success or until the scaling limit is reached (up to eight times). The voxel size is set to 1 cm. The results are shown in Fig. 3.

In Fig. 3(A), the results of three dogs with different body shapes are shown in URDF form. A neutral pose of each designed robot is illustrated. The red meshes indicate the motors with connectors.

In subplot (B), we report the conditioned success rate and time consumption of the design process as functions of the number of design rounds. We introduce the term "conditioned" to indicate that these designs are not physically fabricated or assembled, but are instead evaluated automatically based on the following conditions: (1) available motors in the motor library can be assigned for all links; (2) each optimized mesh can accommodate its assigned motors without motion or assembly interference. In the Geometry Optimization step, the motor mesh has been removed from the structure, and a motor-holding shell has been fused to ensure proper motor integration. Thus, our verification focuses on ensuring that the mesh remains structurally intact, i.e., watertight, after voxel-level interference removal in both the Geometry Optimization step and the Assembly Details Handling step described in the "Methods" section. (3) Each mesh passes FEA validation, i.e., the Von Mises stress remains below a density-dependent material threshold under the maximum motor torques, which verifies the absence of structurally weak regions.

From round 1 to round 8, the conditioned success rate increases from 36% to 92%. The conditioned success rate does not increase from round 7 to 8. For this reason, round 8 is not included in subplots (E) and (F). Each design that passes the conditions requires 2.39 rounds on average. The average time consumption of a successful design increases from about 9 min to 60 min, from round 1 to round 8. The minimum time consumption is about 5 min, while the maximum is about 1.5 h. On average, each round takes about only 8.5 min, and the successful design of a robot takes 20.4 min. Increasing the voxel resolution to achieve finer mesh detail results in higher time consumption, as more voxels must be processed during motor and geometry optimization. When the voxel resolution is increased to 0.5 cm, it takes about 22.5 min for one round.

The pie plots in (C) and (D) further illustrate the distribution of the time consumption and the causes of failure. From the pie charts, it is evident that time consumption is primarily attributed to the actuator optimization and density optimization steps, where the evolutionary algorithm and FEA are the main time consumers. The majority (59.4%) of the failure rounds are caused by the FEA constraint not being satisfied, typically due to the initial mesh containing overly thin parts or interference removal between links creating excessively thin parts, which weakens the link and prevents it from withstanding torque, even with maximum density. The second cause of failure is the inability to find a suitable motor from the library for some links. The third cause is that interference removal damaged the mesh of a link, preventing it from properly connecting to the motors.

In addition to successfully generating a robot design from the media input, another key objective is to ensure the robot resembles the description in the media input. Since we use existing 3D generative models to obtain a 3D mesh when the input is text or image in the preprocessing step, similarity is evaluated only between the final meshes and the initial mesh from the preprocessing step. A more advanced 3D generative model can always be adopted in the future. Subplot (E) shows the average point distance between the original mesh given by Bite[45] and the final designed meshes. The average point distance is calculated by measuring the distance of densely sampled points on one mesh to the nearest point on the other mesh and then averaging the distances. The orange line indicates the average distance normalized by the size of the dog, i.e., dividing the average distance by the scaling factor of each round. The plot shows that the original average distance does not change much with the number of rounds, while the normalized distance decreases, suggesting that larger dogs exhibit greater shape similarity to the input. The reason is that larger shapes offer more spatial flexibility to integrate motors within the mesh, resulting in fewer geometric modifications. Overall, the average point distances across all rounds are below 1 cm.

In subplot (F), we show the average relative density of the links after the density optimization step with a bar plot that has standard deviations. PLA is used as the material for the links, and the six-fold plate[47] lattice structure is used as the meta-material. In the successful designs, the average relative density is below 0.15, while the average relative density plus one standard deviation is below 0.3. This indicates that the density optimization step can effectively reduce the density of the links to make them lightweight and less costly to print. We further categorize the links into thigh, shank, and body, and draw a subplot (G). Since the body usually contains a thicker mesh than the shank and thigh, the average relative density of the body is smaller than that of the shank and thigh after optimization.

Overall, the designed quadrupeds show a high conditioned success rate, low time consumption, and a lightweight design that resembles the input, indicating the effectiveness of our system. More results for the motor selection are presented in Fig. S3 of the supplementary material.

Results with different inputs

We further tested our system with diverse personalized inputs across different modalities–including text, images, and 3D meshes. While our system is primarily designed for image and text inputs, 3D mesh models are also compatible by bypassing the image/text-to-3D model conversion step during preprocessing. This flexibility enables users with 3D modeling expertise to design personalized robots directly from their own geometric models. The results are presented in Fig. 4. The initial meshes are generated by Meshy[44]. The voxel size in these designs is set to 0.5 cm to achieve a fine mesh. The system successfully generates the designs in about 30 min to 5 h. We use a teach-and-repeat controller to control the URDF model of the robots in simulation. Results can be found in the attached video.

Discussion

The presented automatic robot design system generates a 3D printable limbed robot design from a user-specified text, image, or 3D model input. While expert-designed robots can achieve higher task-specific performance, our goal is fundamentally different. We aim to enable automatic, accessible and scalable robot design from unstructured mesh inputs, a setting not addressed by existing methods. Quantitative results show that the system can generate a design that resembles the input with a high success rate and low time consumption. Real-world tests with a Centaur robot designed from an image input further demonstrate the system's effectiveness. With the system, non-expert users can create a personalized limbed robot in a few fes to a few hours.

Despite the strong generalization ability demonstrated across diverse inputs, several challenges remain. First, the system depends on the quality of the initial mesh generated from text or image inputs, which may introduce geometric artifacts and propagate errors to subsequent design stages. Second, the current formulation only supports rotational joints, limiting the ability to model mechanisms requiring linear or more complex joint types. Third, structural constraints, such as thin components failing under FEA or interference removal, can lead to unsuccessful designs.

In addition to addressing the challenges above, we identify three main directions for future improvements of the system.

First, controller design can be integrated into the system to realize co-design. Our current system uses either a controller designed for quadrupeds or a simple teach-and-repeat controller. However, this can only achieve simple motions. An advanced learning-based controller can be trained for each designed robot to realize more complex motions. Furthermore, the controller can be optimized with the robot design to make the designed robot more robust and adaptive.

Second, the design process can be made differentiable to achieve a more efficient design. Modern GPUs and AI accelerators can provide a large amount of computing power to accelerate the design if the design process is differentiable. This acceleration can help the system explore the design space more efficiently. To make the design process differentiable, a continuous representation, such as neural representations and 3D Gaussian, could be used. The FEA process should also be differentiable to realize error backpropagation.

Finally, video input can be explored. Video inputs provide multi-view and multi-time information about an object and, thus, can be used to enhance the geometry quality, detect desired joints, and capture motions of the object. A controller that mimics the motion in the video can be designed to make the designed robot perform the motion, making the designed robot more personalized and interesting.

Methods

An overview of the system has been presented in Fig. 1. This section further presents the detailed method of the system. We first present the preprocessing and mesh decomposition, where existing algorithms are used to generate proper representations suitable for the following optimization steps. Our core methods are in the actuator, geometry, and density optimization steps, which select the actuators and generate strong, lightweight, non-interfering, and 3D printable mechanical components for the robot. We present the problem formulations and solutions of each step in this section. Finally, the assembly details that handle the connections of the motors and mechanical components are described.

Preprocessing

This step is to generate the initial mesh and joint configuration from the input text or image. Thanks to the recent advancement in text/image to 3D generation[11–16], pose estimation[45,51] and auto-rigging methods[49,52,53], mesh and joints of a limbed character can be generated from the input text or image with one of these methods or the combination of two. To make the joints adjustable by the user, we designed a UI to modify the joint positions and orientations interactively (shown in Fig. S2 of the supplementary material). In our experiments, the initial mesh and joints of the quadrupeds are estimated by Bite[45] with the neutral pose applied. The initial meshes of the rest of the characters are estimated by Meshy[44], while their joints are adjusted by the designed UI. Figure 5(A) shows the initial mesh and joints of the Lynel. Red arrows indicate the directions of each joint, which can be either one DOF or two DOFs.

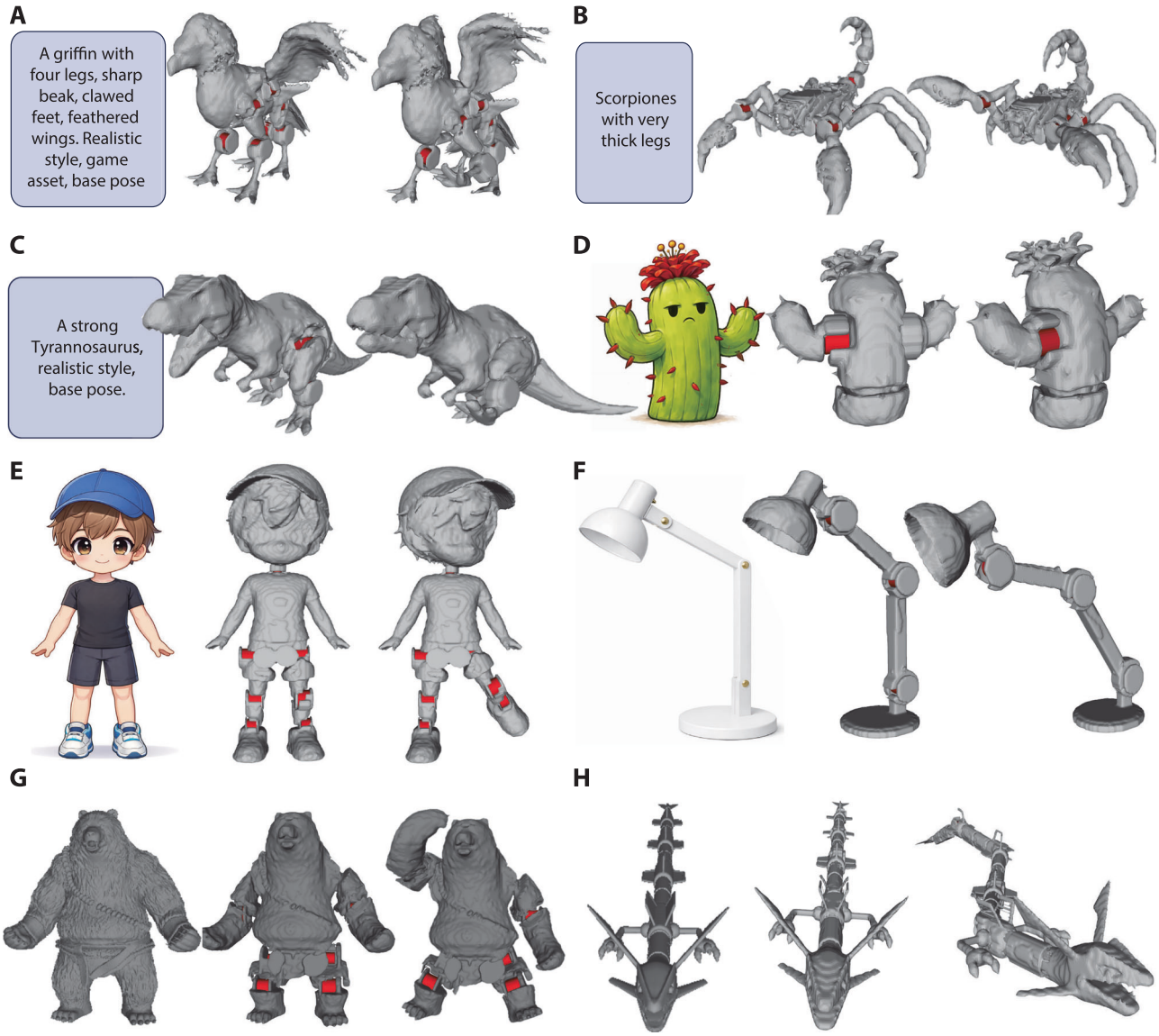

Fig. 4 | Results with different input modalities. A–C are designed from text inputs. **D–F** are designed from image inputs. **G**, **H** are designed from 3D model inputs. In each subplot, the left figure is the input (input figures are generated by the OpenAI image generation tool), the middle figure is the automatically designed robot with a neutral pose, and the right figure is the robot with a motion. Red meshes indicate the motors.

Mesh decomposition

To process the mesh more efficiently, we voxelize the mesh and then decompose it into voxel clusters that belong to different mechanical components, i.e., links of a limbed robot. Since the number of original joints from pose estimation[45,51] and auto-rigging methods[52,53] are usually much more than the number of joints that can be constructed in a real limbed robot, we reduced the number of joints by (1) for quadrupeds, we only keep the joints of the front and rear legs and use a typical 12 DOF model[1–4]; (2) for other characters, users can select the movable joints and define the rotation directions of the joints in the UI. The rest of the original joints are considered fixed as feature points of each link, denoted as black points in Fig. 5(A). We define the joints that connect link i and link j as joint (i, j), the position of this joint as $\widehat{\boldsymbol{p}}_{i,j} \in \mathbb{R}^3$, and the direction vector of the joint as $\widehat{\boldsymbol{n}}^k_{i,j} \in \mathbb{R}^3$ with k indicating the degrees of freedom.

The mesh is then decomposed into individual links based on the joint positions and feature points described above. Specifically, we iterate through each occupied voxel and assign it to the link whose joints, feature points, or the lines connecting them are closest to the voxel, based on Euclidean distance. This spatial proximity-based assignment ensures that each voxel is grouped with the most geometrically relevant link. Note that, although mesh decomposition can also be completed with some pose estimation models and auto-rigging methods, distance-based clustering is adopted here to keep the decomposition efficient and one unified method for different preprocessing methods. After the decomposition, a voxel map $\mathbb{G}$ with each voxel $g \in \mathbb{G}$, either occupied and assigned to a link or unoccupied, is acquired.

Actuator optimization

The actuator optimization aims to select and place actuators on the links. For simplicity, we consider that each actuator is placed at the joint of two links without using power transmission components, such as a belt or a chain. Gears are integrated in the actuator; thus, existing off-the-shelf brushless joint motors can be used here. We consider the following four requirements in motor placement and selection: (1) Torque constraint: The motors should provide sufficient torque to actuate the corresponding link; (2) Interference Constraint: The motors should not interfere with each other; (3) Kinematic alignment: Each motor should be placed close to its corresponding joint position $\widehat{\boldsymbol{p}}_{i,j}$, and its orientation should align with the joint direction $\widehat{\boldsymbol{n}}^k_{i,j}$; (4) Mesh coverage: The motors should be covered by the mesh as much as possible, so that the generated robot resembles the input geometry.

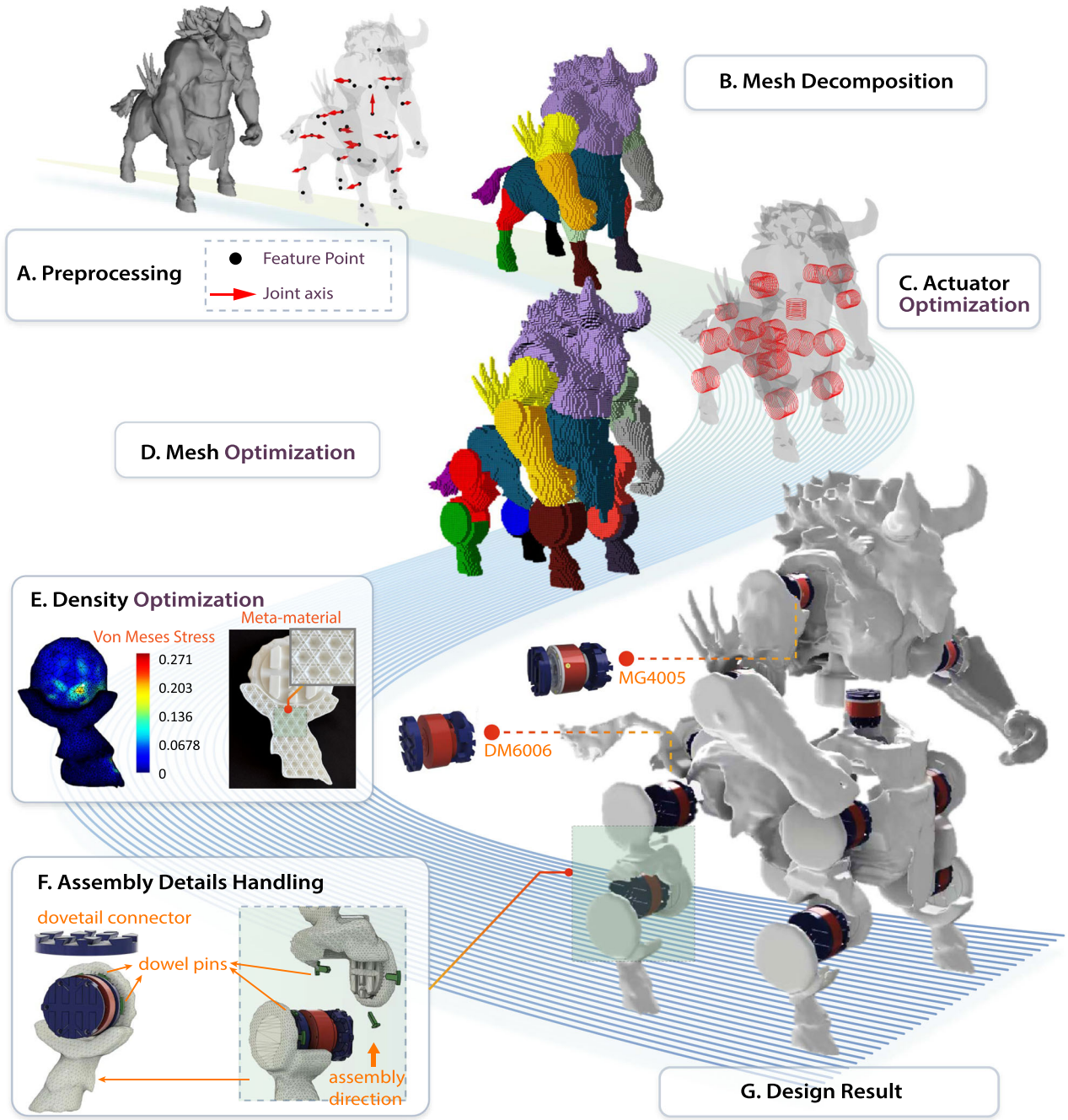

Fig. 5 | Illustration of the intermediate results of each step. A Initial mesh and joints of the Lynel from the preprocessing step. **B** Voxelized mesh and the decomposed voxel clusters from the mesh decomposition step. **C** The selected motors and their positions and directions in the actuator optimization step. **D** The optimized mesh from the geometry optimization step in voxel form. **E** An example of the density optimization step, including an image of the FEA result of a shank and an image of the result meta-material filling. **F** Illustration of the dovetail connector and the assembly process. **G** The final designed robot in an exploded view.

We formulate this problem as an optimization problem. In our current system, all joints are modeled as revolute joints. A joint may have either one or two rotational DoFs. We assign one actuator variable to each rotational DoF of joint (i, j). Therefore, a one-DoF joint is realized by one revolute motor and has one actuator variable, while a two-DoF joint is realized by two orthogonally arranged revolute motors connected in series and has two actuator variables. The actuator corresponding to the k-th DoF of joint (i, j) is denoted by $\boldsymbol{a}_{i,j}^k \in \mathbb{R}^7$, where $k \in \{1, 2\}$:

$$\boldsymbol{a}_{i,j}^k = \left[\boldsymbol{p}_{i,j}^{k\,T} \boldsymbol{n}_{i,j}^{k\,T} t_{i,j}^k \right]^T \tag{1}$$

where $\boldsymbol{p}_{i,j}^k$ is the 3D position of the motor, $\boldsymbol{n}_{i,j}^k$ is the orientation vector of the motor, and $t_{i,j}^k$ is an integer indicating the selected motor type from the motor library. The shape of the motor is simplified as a cylinder with radius $r(t_{i,j}^k)$ and height $h(t_{i,j}^k)$, and the torque of the motor is $m(t_{i,j}^k)$. We use a motor library composed of off-the-shelf motors with different sizes and torques to define the search space for the motor selection. For a one-DoF joint, only one actuator variable is instantiated. For a two-DoF joint, two actuator variables, corresponding to the two rotational DoFs, are instantiated. These two revolute motors are connected in series by a fixed, predesigned connector. Therefore, their relative position and orientation are fixed by the connector design, rather than optimized independently.

The general optimization problem is formulated as:

$$\begin{aligned} \min_{\boldsymbol{a}_{i,j}^k} \sum_{(i,j)\in\mathbb{J}} \sum_k \left[f_{\mathrm{K}}(\boldsymbol{a}_{i,j}^k) + f_{\mathrm{C}}(\boldsymbol{a}_{i,j}^k) \right] \\ \text{s.t. } g_{\mathrm{I}}(\boldsymbol{a}_{i,j}^k) \le 0,\ \forall (i,j) \in \mathbb{J}, \forall k \\ g_{\mathrm{T}}(\boldsymbol{a}_{i,j}^k) \le 0,\ \forall (i,j) \in \mathbb{J}, \forall k \end{aligned} \tag{2}$$

where $\mathbb{J}$ denotes the set of joints, f_{K} is the kinematic alignment cost, f_{C} is the mesh coverage cost, g_{I} is the interference constraint and g_{T} is the torque constraint.

The kinematic alignment cost f_{K} is defined as follows:

$$f_{\mathrm{K}}(\boldsymbol{a}_{i,j}^k) = w_1 \left\| \boldsymbol{p}_{i,j}^k - \widehat{\boldsymbol{p}}_{i,j} \right\|_2 + w_2 \left(1 - \frac{\| \boldsymbol{n}_{i,j}^k \cdot \widehat{\boldsymbol{n}}_{i,j}^k \|}{\left\| \boldsymbol{n}_{i,j}^k \right\|_2 \left\| \widehat{\boldsymbol{n}}_{i,j}^k \right\|_2} \right) \tag{3}$$

where w_1 and w_2 are the weights for the position and direction constraints, respectively.

To calculate the mesh coverage cost $f_{\mathrm{C}}(\boldsymbol{a}_{i,j}^k)$, we sample points uniformly on the surface of each motor and check whether they are covered by the original mesh:

$$f_{\mathrm{C}}(\boldsymbol{a}_{i,j}^k) = w_3 \sum_{l=1}^{N_s} \min\left(\mathrm{SDF}\left(s_l(\boldsymbol{a}_{i,j}^k) \right), 0 \right) \tag{4}$$

where $s_l(\boldsymbol{a}_{i,j}^k)$ represents the l-th sampling point on the surface of the cylindrical motor defined by position $\boldsymbol{p}_{i,j}^k$, direction $\boldsymbol{n}_{i,j}^k$, radius $r(t_{i,j}^k)$ and height $h(t_{i,j}^k)$. N_s is the total number of sampling points. SDF($\cdot$) calculates the signed distance from a point to the original surface mesh, with positive values indicating points outside the mesh and negative values for points inside. The minimum operation and summation penalize sampling points outside the mesh, encouraging the motor to be covered by the initial mesh given by preprocessing, so that the generated robot resembles the original mesh.

The interference constraint $g_{\mathrm{I}}(\boldsymbol{a}_{i,j}^k)$ is formulated as the analytical collision check between cylinders introduced in ref. 54.

The torque constraint $g_{\mathrm{T}}(\boldsymbol{a}_{i,j}^k)$ is defined as follows:

$$g_{\mathrm{T}}(\boldsymbol{a}_{i,j}^k) = \tau_{\mathrm{est}}(i,j) - m(t_{i,j}^k) \tag{5}$$

where $\tau_{\mathrm{est}}(i,j)$ is the estimated torque of the joint (i,j) calculated from inverse dynamics[55] and $m(t_{i,j}^k)$ is the maximum torque of the motor.

We note that the optimization problem is a mixed-integer and non-convex program, due to the introduction of integer variable $t_{i,j}^k$ in Equation (1) and the use of the SDF($\cdot$) function introduced in Equation (4). To solve this problem, we leverage the genetic algorithm to find the solution. Details can be found in the first subsection of the supplementary material.

Geometry optimization

The geometry optimization aims to modify the mesh of each link so that the following requirements can be satisfied: (1) The optimized mesh supports the mounting of the motors; (2) The interference between meshes of different links during motion can be avoided. These modifications include the addition and reduction of some parts of the meshes. At the voxel level, this leads to a change of free and occupied status of all possible voxels on the robot and in the space near the robot. Since the number of voxels can be up to a million, the optimization space is as huge as two raised to the one millionth power. Meanwhile, it is hard to directly express the requirements as mathematical constraints or objective functions. Considering the complexity of the problem, we break down the geometry optimization into three individual steps: (1) Mesh Expansion, (2) Key Voxel Searching, and (3) Mesh Interference Removal.

In Mesh Expansion, we first add cylinder-shaped occupied shell voxels around each motor to ensure proper mounting. Let $\mathrm{G}_{i,j} \subset \mathbb{G}$ denote the set of shell voxels around motor $\boldsymbol{a}_{i,j}$. The shell voxels are generated by dilating the cylindrical motor shape with a fixed thickness. These shell voxels should be further assigned to two links that the motor connects to.

Intuitively, voxels on the top and bottom surfaces of the motor cylinder model, corresponding to the output shaft and the mounting base of the motor, should be assigned to the link i and j, respectively, for rotational motion between links. Voxels on the lateral surface of the motor, however, should be assigned partially to i and the rest to j, to make the shell voxels well connected to the original voxels of each link. To guide this assignment, we use a Support Vector Machine (SVM) classifier to find a separating plane based on the distribution of original voxels belonging to each link. This encourages voxels on the lateral surface to be assigned to the nearby link. Details are explained in the supplementary material with Fig. S1.

However, we observe that the shell voxels assigned to a link are not always connected to the original voxels of the link. This happens because, during the actuator optimization step, to satisfy the constraints, the motor's final position may be placed away from the links' interfacial surface defined by the mesh decomposition, resulting in a discontinuity. Even if connectivity is established at this stage, it may still be disrupted in subsequent steps, such as interference removal, where some connecting voxels may be removed. To further ensure structural connectivity, we perform Key Voxel Searching to identify the key voxels that bridge the shell voxels and the original voxels of the assigned link. Since this problem is equivalent to determining the connection between two voxel clusters, we reformulate it as a path-finding problem and solve it with the A* algorithm. For each shell voxel $g \in G_{i,j}$, we find the shortest path $\mathcal{P}(g) \subset G$ to the nearest voxel in its assigned link $A(g)$. All voxels along the path $\mathcal{P}(g)$ are also added to the corresponding link to ensure structural connectivity. Since these voxels are important for the connectivity, they are marked as unremovable key voxels. Note that the path search is done within already occupied voxels. The path will not pass through unoccupied voxels, so we will only change the voxel type (e.g., from parent link to child link) and will not add new voxels to the model.

We then execute the Mesh Interference Removal operation to remove the interference between the links during motion. For each joint angle $\theta_{i,j,k}$ between links i and j, we sample N_θ angles uniformly within its motion range $[\theta_{i,j,k}^{\min}, \theta_{i,j,k}^{\max}]$. The sampling starts at $\theta_{i,j,k} = 0$ and expands bidirectionally toward the joint limits with a step size determined by N_θ, ensuring symmetric coverage of the motion range. For each sampled angle $\theta_{i,j,k}^s$, we apply the corresponding homogeneous transformation matrix $T_{i,j,k}(\theta_{i,j,k}^s)$ to rotate the voxel grid of link j and all its child links. The rotation is performed around the axis defined by the actuator optimization result. If any voxels from link j overlap with voxels from link i after transformation, we resolve the collision based on the removability of the involved voxels determined by Key Voxel Searching. If both voxels are removable, we remove the voxel from link i. If one voxel is non-removable, we remove the other one. If both voxels are non-removable, the configuration is considered invalid, and we restart the design loop. This process ensures that the links can move freely without collision throughout their range of motion.

Eventually, the mesh of each link is reconstructed from the voxel grid G using the marching cubes algorithm[46].

Density optimization

The mesh of each link, i.e., the mechanical structure of the robot, is to be 3D-printed for fast and complex shape fabrication. Direct printing of fully dense structural components is not feasible because the weight of the link is usually too heavy, while the strength is usually redundant. To solve this problem, we fill the mesh with metamaterial and optimize the relative density of the metamaterial of each link to make the link lightweight and yet strong enough.

Let ρ denote the relative density of a link. The density optimization problem for the link is formulated as follows:

$$\begin{aligned} &\min_{\rho} \|\rho\|_2 \\ &\text{s.t.} \sigma^{\text{vm}}(\rho, \mathrm{E}^*, \epsilon, \boldsymbol{\tau}, \boldsymbol{f}) \leq \sigma^{\text{vm}}_{\max}(\rho) \\ &\rho \in (0, 1] \end{aligned} \tag{6}$$

where $\sigma^{vm}(\rho, E^*, \epsilon, \boldsymbol{\tau}, \boldsymbol{f})$ is the maximum von Mises stress of the link under the applied torque $\boldsymbol{\tau}$ and external force $\boldsymbol{f}$. E^* and ϵ are the effective Young's modulus and the Poisson's ratio of the meta-material, respectively. $\sigma^{vm}_{\max}(\rho)$ is the maximum allowed von Mises stress under the given relative density ρ.

In practice, we use the six-fold plate[47] lattice structure as the meta-material. The effective Young's modulus is a function of the relative density ρ and the Young's modulus of the material. Detailed expressions of E^* and $\sigma^{vm}_{\max}(\rho)$ can be found in ref. 47. To calculate σ^{vm}, we use CGAL to generate the tetrahedral mesh of each structure and then use PyAnsys to perform the FEA automatically. Since torques are not directly accepted by FEA, we convert the torques to external forces applied on the link connection points.

To quickly find a solution for the density optimization problem, we use a gradient descent-based searching method to find the best ρ that satisfies the constraint. The plot on the left of Fig. 5E shows an FEA result of a shank link with a relative density of 0.15. With optimized density, we use SolidPython and OpenSCAD to generate the plates of the metamaterial and fill the mesh with Boolean operations at the mesh level. The plot on the right of Fig. 5E shows the inner metamaterial structure of the shank after meta-material filling. A skin of 1 mm thickness is kept to retain the original appearance and further enhance structural strength. More details about the density optimization can be found in the supplementary material.

Assembly details handling

Given the optimized motor type and position, and the mesh of each mechanical structure, this step adds connectors to the links to connect the motors and mechanical components and further removes the interference in assembly. Since the links are 3D-printed mechanical structures and the skin of the mesh is thin, e.g., 1 mm, to keep the structure lightweight, using screws or bolts to connect the links and motors will cause high stress concentrations and cause the structure to break. Additionally, the arbitrary shapes complicate the assembly process of screws and bolts: sometimes an extremely deep hole is required, and in other cases, the screw path may be obstructed by another component.

Inspired by the tenon and mortise connections commonly used in woodworking, we design a special dovetail connector that can connect the links and motors firmly without causing high stress concentrations. An illustration of the dovetail connector is shown in Fig. 5H. These connectors are designed as motor-specific standardized components, where each connector is paired with a particular motor type. These connectors are manually designed, 3D printed (FDM) and empirically validated to ensure reliable assembly and sufficient structural integrity under the maximum torque of the corresponding motor.

Each connector consists of one part on the motor and the other part on the link. For the part on the motor, screws holes are added to fit the off-the-shelf motors. For the part on the link, we directly fuse it to the link mesh by mesh Boolean operations. Multiple dovetail grooves that fit each other are added to each part of the connector to ensure a strong and stable connection. Up to six dowel pins can be added to prevent motion along the grooves. In practice, two pins on the most accessible side are sufficient to ensure a stable connection. The overall thickness of the connector is about 15 mm. We add this thickness to the height of the motor so that the thickness of the connector is considered in the optimization steps. The placement of the connector on the link is decided by the optimized motor position.

While the dovetail connectors are currently designed manually to interface with the screw holes of off-the-shelf motors, this step is not part of our core automatic design pipeline. Ideally, such connectors should be standardized or provided by the motor manufacturers themselves as part of a modular interface, replacing the traditional screw-based connection. Therefore, we do not regard this manual adaptation as a limitation of our automatic robot design system.

The two parts of the connector need to be assembled by sliding along the direction of the grooves. To prevent possible interference during assembly, we search for a direction that minimizes mesh interference on the link mesh and allows smooth sliding engagement by uniform sampling. In case there is still interference along the best direction, Boolean operations are used to remove the interference. Finally, with the motors, link meshes, and connectors, the robot can be assembled.

Data availability

All data, materials, and codes needed to evaluate the paper's conclusions are publicly available at: https://github.com/g-ch/anything2robot.

Received: 16 November 2025; Accepted: 26 May 2026;
Published online: 09 June 2026

References

1. Katz, B., Di Carlo, J. & Kim, S. Mini cheetah: a platform for pushing the limits of dynamic quadruped control. *IEEE Int. Conf. Robot. Autom*. 6295–6301 https://doi.org/10.1109/ICRA.2019.8793865 (2019).
2. Grimminger, F. et al. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robot. Autom. Lett.* **5**, 3650–3657 (2020).
3. Biswal, P. & Mohanty, P. K. Development of quadruped walking robots: a review. *Ain Shams Eng. J.* **12**, 2017–2031 (2021).
4. Hoeller, D., Rudin, N., Sako, D. & Hutter, M. Anymal parkour: learning agile navigation for quadrupedal robots. *Sci. Robot.* **9**, eadi7566 (2024).
5. Chignoli, M., Kim, D., Stanger-Jones, E. & Kim, S. The MIT humanoid robot: design, motion planning, and control for acrobatic behaviors. *IEEE-RAS Int. Conf. Humanoid Robots*1–8 https://doi.org/10.48550/arXiv.2104.09025 (2021).
6. Tong, Y., Liu, H. & Zhang, Z. Advancements in humanoid robots: a comprehensive review and future prospects. *IEEE/CAA J. Autom. Sinica* **11**, 301–328 (2024).
7. Shi, Q. et al. Development of a small-sized quadruped robotic rat capable of multimodal motions. *IEEE Trans. Robot.* **38**, 3027–3043 (2022).
8. Crespi, A. & Ijspeert, A. J. in *Artificial Life Models in Hardware*. 35–64 (Springer, 2009).
9. Vidoni, R. & Gasparetto, A. Efficient force distribution and leg posture for a bio-inspired spider robot. *Robot. Auton. Syst.* **59**, 142–150 (2011).
10. Graichen, K. et al. Control design for a bionic kangaroo. *Control Eng. Pract.* **42**, 106–117 (2015).
11. Liu, M. et al. One-2-3-45++: Fast single image to 3D objects with consistent multi-view generation and 3D diffusion. *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. 10072–10083* (IEEE, 2024).
12. Tang, J. et al. Make-It-3D: High-fidelity 3D creation from a single image with diffusion prior. *Proc. IEEE/CVF Int. Conf. Comput. Vis*. 22762–22772 (IEEE, 2023).
13. Raj, A. et al. Dreambooth3d: Subject-driven text-to-3D generation. *Proc. IEEE/CVF Int. Conf. Comput. Vis*. 2349–2359 (IEEE, 2023).
14. Poole, B. et al. DreamFusion: Text-to-3D using 2D diffusion. *Int. Conf. Learn. Represent*. (2023).
15. Shi, Y. et al. MVDream: Multi-view diffusion for 3D generation. *Int. Conf. Learn. Represent.* (2024).
16. Long, X. et al. Wonder3D: Single image to 3D using cross-domain diffusion. *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit*. 9970–9980 (IEEE, 2024).
17. Thomaszewski, B. et al. Computational design of linkage-based characters. *ACM Trans. Graph.* **33**, 1–9 (2014).

18. Maloisel, G. et al. Singularity-aware design optimization for multi-degree-of-freedom spatial linkages. *IEEE Robot. Autom. Lett.* **6**, 6585–6592 (2021).
19. Skouras, M. et al. Computational design of actuated deformable characters. *ACM Trans. Graph.* **32**, 1–10 (2013).
20. Huber, S., Poranne, R. & Coros, S. Designing actuation systems for animatronic figures via globally optimal discrete search. *ACM Trans. Graph.* **40**, 1–10 (2021).
21. Xu, H. et al. Bend-it: Design and fabrication of kinetic wire characters. *ACM Trans. Graph.* **37**, 1–15 (2018).
22. Coros, S. et al. Computational design of mechanical characters. *ACM Trans. Graph.* **32**, 1–12 (2013).
23. Song, P. et al. Computational design of wind-up toys. *ACM Trans. Graph.* **36**, 1–13 (2017).
24. Roussel, R. et al. Exploratory design of mechanical devices with motion constraints. *Comput. Graph.* **74**, 244–256 (2018).
25. Ha, S.et al. Task-based limb optimization for legged robots. *IEEE/RSJ Int. Conf. Intell. Robots Syst*. 2062–2068 (IEEE, 2016).
26. De Vincenti, F.et al. Control-aware design optimization for bio-inspired quadruped robots. *IEEE/RSJ Int. Conf. Intell. Robots Syst*. 1354–1361 (IEEE, 2021).
27. Belmonte-Baeza, Á et al. Meta reinforcement learning for optimal design of legged robots. *IEEE Robot. Autom. Lett.* **7**, 12134–12141 (2022).
28. Fadini, G.et al. Computational design of energy-efficient legged robots: Optimizing for size and actuators. *IEEE Int. Conf. Robot. Autom*. 9898–9904 (IEEE, 2021).
29. Yuan, Y. et al. Transform2Act: Learning a transform-and-control policy for efficient agent design. *Int. Conf. Learn. Represent*. (2022).
30. Schaff, C. B. & Walter, M. R. N-LIMB: Neural limb optimization for efficient morphological design. Preprint at http://arxiv.org/abs/2207.11773 (2022).
31. Schaff, C. et al. Soft robots learn to crawl: Jointly optimizing design and control with sim-to-real transfer. Proc. Robotics: Science and Systems. (2022).
32. Matthews, D. et al. Efficient automatic design of robots. *Proc. Natl Acad. Sci. USA* **120**, https://doi.org/10.48550/arXiv.2306.03263 (2023).
33. Zhao, A. et al. RoboGrammar: Graph grammar for terrain-optimized robot design. *ACM Trans. Graph.* **39**, 188 (2020).
34. Megaro, V. et al. Interactive design of 3D-printable robotic creatures. *ACM Trans. Graph.* **34**, 216 (2015).
35. Desai, R., Yuan, Y. & Coros, S. Computational abstractions for interactive design of robotic devices. *IEEE Int. Conf. Robot. Autom*. 1196–1203 (IEEE, 2017).
36. Schulz, A. et al. Interactive robogami: an end-to-end system for design of robots with ground locomotion. *Int. J. Robot. Res.* **36**, 1131–1147 (2017).
37. Desai, R.et al. Geppetto: Enabling semantic design of expressive robot behaviors. *Proc. CHI Conf. Hum. Factors Comput. Syst*. 369 (ACM, 2019).
38. Feng, X. et al. Computational design of skinned quad-robots. *IEEE Trans. Vis. Comput. Graph.* **27**, 2881–2895 (2021).
39. Du, T. et al. Computational multicopter design. *ACM Trans. Graph.* **35**, 227 (2016).
40. Maloisel, G. et al. Optimal design of robotic character kinematics. *ACM Trans. Graph.* **42**, 1–15 (2023).
41. Ha, S. et al. Computational co-optimization of design parameters and motion trajectories for robotic systems. *Int. J. Robot. Res.* **37**, 1521–1536 (2018).
42. Ha, S. et al. Computational design of robotic devices from high-level motion specifications. *IEEE Trans. Robot.* **34**, 1240–1251 (2018).
43. Grandia, R. et al. Design and control of a bipedal robotic character. *Robot. Sci. Syst*. https://doi.org/10.48550/arXiv.2501.05204 (2024).
44. Meshy AI. *Meshy: AI-Powered Text-and-image-to-3D Model Generator*. https://www.meshy.ai/ (2025).
45. Ruegg, N.et al. BITE: Beyond priors for improved three-dimensional dog pose estimation. *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit*. 8867–8876 (IEEE, 2023).
46. Lorensen, W. E. & Cline, H. E. Marching cubes: a high resolution 3D surface construction algorithm. in *Seminal Graphics: Pioneering Efforts That Shaped the Field*. 347–353 https://doi.org/10.1145/37402.37422 (1998).
47. Wang, Y. & Sigmund, O. Quasiperiodic mechanical metamaterials with extreme isotropic stiffness. *Extreme Mech. Lett.* **34**, 100596 (ACM, 2020).
48. Dataset, E. Novel datasets for fine-grained image categorization. *IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* **5**, 2 (2011).
49. Song, C. et al. Magicarticulate: Make your 3D models articulation-ready. *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit*. 15998–16007 (IEEE, 2025).
50. Lee, J. *Hierarchical Controller For Highly Dynamic Locomotion Utilizing Pattern Modulation And Impedance Control: Implementation on the MIT Cheetah robot*. PhD thesis, Massachusetts Institute of Technology (2013).
51. OpenMMLab Contributors. OpenMMLab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose (2020).
52. Xu, Z. et al. Rignet: Neural rigging for articulated characters. *ACM Trans. Graph.* **39**, 4 (2020).
53. Liu, I. et al. RigAnything: Template-free autoregressive rigging for diverse 3D assets. *ACM Trans. Graph.* **44**, 122 (2025).
54. Ketchel, J. & Larochelle, P. Collision detection of cylindrical rigid bodies for motion planning. *IEEE Int. Conf. Robot. Autom*. 1530–1535 (IEEE, 2006).
55. Carpentier, J. et al. The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. *IEEE Int. Symp. Syst. Integr*. (IEEE, 2019).
56. Zelda Wiki Golden Lynel. https://zelda.fandom.com/wiki/Golden_Lynel (2018).

Acknowledgements

No funding was received for this research.

Author contributions

G.C. proposed the initial idea. G.C. and M.S. implemented the method, conducted the experiments, and wrote the manuscript. Y.X. and L.Z. helped handle the 3D printing for the robot. J.P. supervised the project. L.Z., M.P., and J.A. provided comments and suggestions. All authors reviewed the final manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s44182-026-00101-3.

Correspondence and requests for materials should be addressed to Gang Chen or Jiangmiao Pang.

Reprints and permissions information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2026