2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Computational Abstractions for Interactive Design of Robotic Devices

Ruta Desai, Ye Yuan and Stelian Coros

Abstract— **We present a computational design system that allows novices and experts alike to easily create custom robotic devices using modular electromechanical components. The core of our work consists of a design abstraction that models the way in which these components can be combined to form complex robotic systems. We use this abstraction to develop a visual design environment that enables an intuitive exploration of the space of robots that can be created using a given set of actuators, mounting brackets and 3d-printable components. Our computational system also provides support for design auto-completion operations, which further simplifies the task of creating robotic devices. Once robot designs are finished, they can be tested in physical simulations and iteratively improved until they meet the individual needs of their users. We demonstrate the versatility of our computational design system by creating an assortment of legged and wheeled robotic devices. To test the physical feasibility of our designs, we fabricate a wheeled device equipped with a 5-DOF arm and a quadrupedal robot.**

I. Introduction

In the not-so-distant future, a rich ecosystem of robotic devices will be tightly integrated into the fabric of our daily lives. To best serve their purpose, many of these robots will need to be custom-designed for different tasks or according to the individual requirements of their users. Unfortunately, with current design methodologies, the process of creating new robotic systems is notoriously challenging, time-consuming and resource-intensive [1].

We present a computational design system for interactive, on-demand generation of custom robotic devices. To make it easy to configure and deploy robots, our design system makes use of a library of standard building blocks such as actuators and mounting brackets. We assume that these modular components are either off-the-shelf, mass-produced parts or that they can be 3D printed. The task of designing a new robot therefore amounts to choosing which components to use, how to combine them to form a functional system, and how to control the resulting device in order to achieve a desirable set of motions or behaviors. Our computational design system assists users with each of these tasks.

One of the main goals of our work is to make design processes more efficient by appropriately modeling the way in which modular building blocks can be combined to form complex robotic systems. To this end, we formalize a design abstraction whereby each component is represented in terms of geometric features and virtual pins. The virtual pins establish compatibilities between different components and define the set of possible physical connections between them. We leverage this design abstraction in two ways. First, we develop an easy-to-use visual design environment that allows users to create robotic devices through familiar drag-and-drop interactions. The virtual pins are used to suggest valid placements for each new component, and thus enable an intuitive exploratory design process. Our computational system also provides support for design auto-completion operations. Here, the design abstraction we propose is used to automatically generate assemblies of structural components that connect pairs of actuators whose desired relative placement is user-specified. These intermediate designs are obtained by efficiently searching through the space of possible arrangements of modular components.

Our design system also provides a physical simulation environment where robots can be tested before fabrication. In particular, we employ an existing optimal control method to generate physically-valid motions for legged robots of arbitrary designs [2]. The resulting motions are tracked in simulation using PD control. By providing feedback at interactive rates, users of our system can therefore iteratively adjust their design to best meet their individual needs and preferences.

To evaluate the scalability of our approach, we employ a library of modular components consisting of Dynamixel actuators [3], off-the-shelf brackets and 3D printable connectors. We demonstrate the effectiveness of our computational design system by creating an assortment of legged and wheeled robotic devices. We validate the physical feasibilty of our results by fabricating two of the robots generated with our design system – a wheeled robot with a manipulator arm and a quadruped robot.

II. Related Work

Recent research efforts in the robotics community highlight the need for new design approaches that support on-demand generation of robotic devices [1], [4]. Our work aims to contribute to this exciting emerging area. More specifically, we are inspired by the ease of use and flexibility of the Spore Creature Creator [5], which comes from the computer graphics literature. To date, close to 200 million customized animated characters have been created with this system by non-expert users from around the world [6]. As a long term goal, we wish to make the process of creating customized robotic devices equally powerful and accessible.

Sharing a goal that is similar to ours, several methods that enable the design of origami-inspired robots have been proposed recently [7], [8], [9], [10], [11]. This body of

This work was supported by the National Science Foundation. The authors are with the Robotics Institute, Carnegie Mellon University, USA. {rutad, yyuan2, scoros}@andrew.cmu.edu

work relies on a set of expert-designed foldable building blocks that are hierarchically chained together using a custom scripting language. Although we use a similar abstraction to enable modular composition, rather than defining new robots through a scripting language, we develop an intuitive *visual design environment* that provides guiding suggestions and automatic design completion operations.

We pair our design framework with a physical simulation environment that allows users to test their robots before fabrication. To this end, we build on the work of Megaro et al. [2]. However, instead of relying entirely on 3D printed parts, our design system is capable of also employing standard, off-the-shelf components such as mounting brackets. We can therefore harness the best of both worlds – cost-effectiveness and durability of mass-produced components, and customization of 3D printing. While [2] focused solely on walking motions for 3D-printable robots, our goal is to enable the design of a much larger variety of devices leveraging modular, off-the-shelf components. We also propose an algorithm to search the rich space of component assemblies based on high-level specifications. Our work is also related to specialized design systems such as Tinkerplay [12] and VEX assembler [13]. We take inspiration from these applications, which are developed for specific types of physical artifacts such as articulated 3d-printable figurines. The goal of our design system, however, is to be more general and provide users with the ability to create robotic devices using different libraries of electromechanical components.

A long-standing goal in robotics is the synthesis of robots from high-level functional specifications. To this end, inspired by Sims' work on evolving virtual creatures [14], a variety of evolutionary methods that aim to co-design a robot's structure and control inputs have been investigated [15], [16], and this effort continues today [17]. However, evolutionary methods rely on stochastic algorithms that are slow, notorious for exploiting unwanted loopholes in the problem specification, and lead to results that are often not repeatable [15]. Rather than relying on stochastic algorithms, our design system puts *the user in the loop* and assists them with suggestions and semi-automatic design completion.

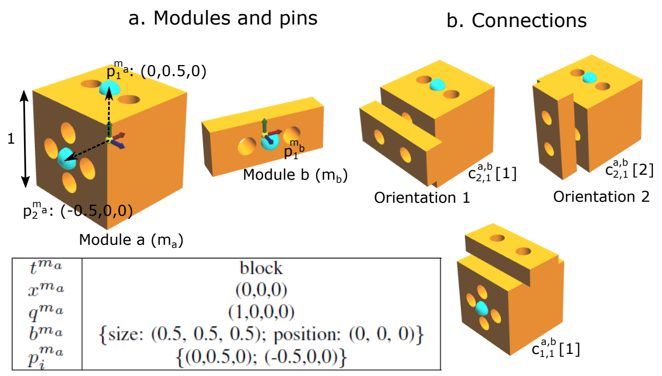
III. Design Abstraction

The goal of our work is to formalize a computational method that enables efficient processes for designing highly customizable robots. To achieve this goal, we develop a design abstraction based on the observation that robotic devices are simply collections of inter-connected electromechanical components. These modular components and the way they connect to each other, therefore, constitute the main elements of our design abstraction.

Modules and pins: Each electromechanical component (henceforth called a module) of a robot is modeled as a rigid body with 6 degrees of freedom (DOF). Each module also has an associated bounding box and a set of virtual pins (henceforth called as pins). Bounding boxes are used during the design process for collision detection, while pins model physically compatible connections between modules. A module m is formally defined as:

$$m = (t^m, x^m, q^m, b^m, \{p_1^m, \ldots, p_n^m\}) \ , \qquad (1)$$

where t^m denotes the module type (e.g. motors), x^m and the quaternion q^m store the module's position and orientation in a global coordinate frame, b^m represents its axis aligned bounding box, and $\{p_i^m\}_{i=1}^n$ is a set of n pins. Each pin p_i^m represents a pre-set location on module m where another module can attach (e.g. the horn of a servomotor). We use local coordinates to represent the location of each pin. Axis aligned bounding boxes for each module are also defined in local coordinates. Fig. 1(a) illustrates two hypothetical modules together with their virtual pins.



t^{m_a}	block
x^{m_a}	(0,0,0)
q^{m_a}	(1,0,0,0)
b^{m_a}	{size: (0.5, 0.5, 0.5); position: (0, 0, 0)}
$p_i^{m_a}$	{(0,0.5,0); (-0.5,0,0)}

Fig. 1. (a) Design abstraction example: Two hypothetical modules (a and b) are shown here. Module a is a block of 1 unit, with two pins (shown in cyan) and module b is a type of connector with one pin. Their local coordinate frames are shown at their center with a triad. The pin positions and bounding box are defined with respect to this frame. The table enumerates the parameters that define module a. (b) All possible connections between modules a and b are shown here. Modules a and b connect with each other at $p_2^{m_a}$ and $p_1^{m_b}$ in two different orientations while they can connect in only one orientation at pins $p_1^{m_a}$ and $p_1^{m_b}$.

We associate with each module a semantic label t^m, which can take on a value in the set $\{motor, connector, body\_plate, end\_effector\}$. Modules with type $motor$ are used to represent the actuated degrees of freedom of a robot, while the $end\_effector$ type indicates that the module is a foot, wheel or manipulator. Modules labeled as $connector$ denote structural components, such as mounting brackets, and $body\_plate$ modules specify locations on the robot's main body where motors can be attached to start new kinematic chains (e.g. a new leg). The geometry of the robot's body is generated as the convex hull of all $body\_plate$ modules in a design.

Connections: Connections encapsulate the conceptual design rules that specify how individual modules can be combined to form complex robotic systems. A connection between two modules m_a and m_b is defined through a pair of compatible pins, as well as a set of physically-valid relative orientations (e.g. due to screw mounting holes):

$$c_{i,j}^{a,b} = \left(p_i^{m_a}, p_j^{m_b}, \{q_1^{m_a,m_b}, \ldots, q_n^{m_a,m_b}\}\right) \ , \qquad (2)$$

where $c_{i,j}^{a,b}$ denotes a connection between modules m_a and m_b using the i^{th} pin of m_a ($p_i^{m_a}$) and j^{th} pin of m_b

$(p_j^{m_b})$. We use $c_{i,j}^{a,b}[k]$ to denote the k^{th} feasible relative orientation (i.e. the quaternion $q_k^{m_a,m_b}$) between the two modules. Connecting two modules through $c_{i,j}^{a,b}[k]$ amounts to positioning them relative to each other such that the world positions of pins $p_i^{m_a}$ and $p_j^{m_b}$ line up and the relative orientation between them, $q_w^{m_a-1} q_w^{m_b}$, is $q_k^{m_a,m_b}$. As shown in Fig. 1(b), modules can be connected to each other both using different relative orientations, and through different pairs of pins altogether. We store the library of modules, the virtual pins and all connectivity information in a configuration file, as exemplified below. We note that the configuration file does not explicitly store the position x^m and orientation q^m of the modules. These are determined automatically during the design process.

Configuration file example:

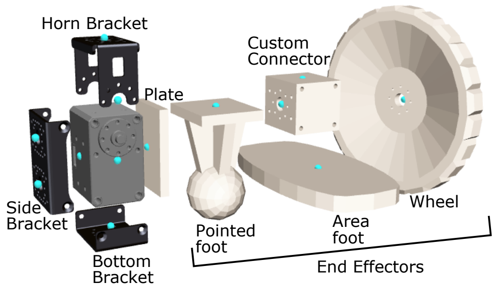
```
Module
        Name a
        Type Block
        BoundingBox
                Size 0.5 0.5 0.5
                Position 0 0 0
        Pin
                Name a-pin1
                Position 0 0.5 0
        Pin
                Name a-pin2
                Position -0.5 0 0
Module
        Name b
        Type Connector
        BoundingBox
                Size 0.75 0.3 0.1
                Position 0 0 0
        Pin
                Name b-pin1
                Position 0 0 0.1
ConnectionMap
        PinPair a-pin2 b-pin1
        RelativeOrientation 0.7 0 0.7 0

        PinPair a-pin2 b-pin1
        RelativeOrientation 0.49 0.49 0.49 -0.49

        PinPair a-pin1 b-pin1
        RelativeOrientation 0.49 0.49 0.49 0.49
```

A complete testbed: To test our computational design approach, we defined a complete library of modular components consisting of Dynamixel XM430 actuators, off-the-shelf mounting brackets and 3D printable end effectors. As illustrated in Fig. 2(a), the library features four types of connectors: *horn bracket*, *side bracket*, *bottom bracket*, and a *custom connector* that can be attached to each type of bracket in multiple ways. The library also includes two types of end-effectors that can be used as point or area feet for legged robots, as well a wheel for mobile robotic vehicles.

IV. Designing Robotic Devices

Our design abstraction enables an interactive process for creating robotic devices. In addition to a manual mode that leverages a graphical user interface capable of providing meaningful suggestions, we also develop a search-based algorithm for design auto-completion. Once a design is finished, our system provides a physically-simulated environment where robots can be tested before being assembled. In particular, for legged robots, we use our previous work to efficiently generate stable full-body motions [2]. By providing feedback at interactive rates, users of our system can iteratively adjust their design until it best meets their needs.

a.



b.

	Motor	Horn Bracket	Bottom Bracket	Side Bracket	End Effector	Plate	Custom Connector
Motor		13	9	18		40	
Horn Bracket	13	8	3	8	5		10
Bottom Bracket	9	3	3	4	5		4
Side Bracket	18	8	4	3	10		
End Effector		5	5	10			
Plate	40						
Custom Connector		10	4				

Fig. 2. (a) A design testbed with Dynamixel modules: We use custom designed modules such as end-effectors, plates and a custom connector (shown in white) as well as commercially available Dynamixel modules such as motors. Their pins are shown in cyan. (b) Table describes connection compatibility between these modules. Each shaded entry indicates the total number of connections between two modules. Wheel can only connect to the motor in one way and hence end effectors entries refer only to the feet.

A. System-guided manual design

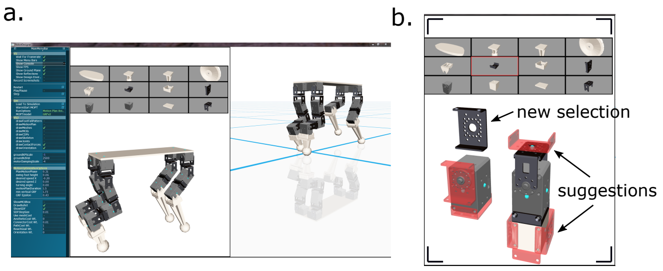


Fig. 3. (a) The design interface consists of two workspaces– the left workspace allows for designing the robot while the right workspace runs a physics simulation of the robot designed by the user for its feasibility. The left workspace displays a list of various modules at the top. The leftmost menu provides various functions that allow users to define preferences for the search process, visualization as well as for physical simulation. (b) When the user selects a new module from the list, our system makes visual suggestions (shown in red) about possible connections for this module, based on the current design.

Using our design abstraction as the technical core, we developed a visual design system for interactive, on-demand generation of custom robotic devices. As Fig. 3(a) illustrates, users of our system are presented with a design workspace (left) and a simulation workspace (right). The design workspace allows them to browse through the list of available modules, which can be dragged and dropped into

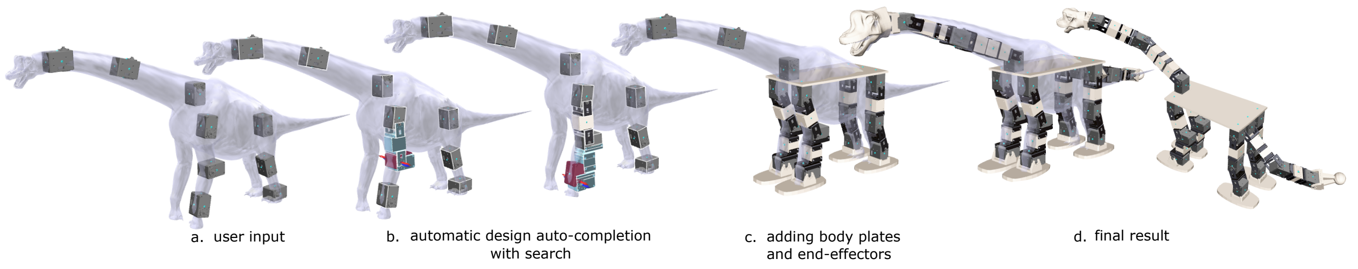

a. user input
b. automatic design auto-completion with search
c. adding body plates and end-effectors
d. final result

Fig. 4. Automatic design with search: (a) Users can start with a guiding mesh for the robot they want to make. Then, they specify the positions and orientations of motors for this robot, using the drag and drop interface. (b) Our search process searches for possible designs that connect a given pair of motors in user-defined locations, according to user-defined preferences. The user can reject the solution and re-do the search with different preferences any-time. A proposed search solution connecting the root motor to the target motor (highlighted in dark red) is shown in light blue. The design is only created if the user accepts the proposed solution. The process is repeated by the user for each pair of motors. (c) Since the legs are symmetric, the users only need to use search process for two legs. Our interface allows them to create the other pair of legs by simple editing operations. Finally, the users can attach end-effectors of their choice and create a body plate to complete the robot design. (d) shows the final design (with and without the guiding mesh). The dinosaur head mesh was manually added after the design, for aesthetic appeal.

the scene at any time. Once a specific module is selected, our system visualizes the ways in which it can be connected to the design the user is currently working on (see Fig. 3(b)). This is achieved by iterating through all unused pins in the current design and checking if they are compatible with the pins of the new module. As the user positions the module in the scene sufficiently close to any of the suggested placements, it is automatically snapped into place. The user can then cycle through the set of possible relative orientations (i.e. $c_{i,j}^{a,b}[k]$) associated with the connection that was selected. Our design system also supports other editing operations that increase productivity. For example, different parts of a design can be marked as symmetric (e.g. left and right limbs), with edits made to one part being automatically propagated to the other. Different parts of an existing design can also be copied, pasted and mixed with other designs. To further guide the design process, users can load a 3D mesh and overlay it into the workspace, as seen in Fig. 4. We illustrate a representative design session in the accompanying video.

B. Design auto-completion

The manual mode described in the previous section affords users with full control over the design. Nevertheless, this exploratory design process can become somewhat tedious if a large number of modules is required to create a robotic device. We therefore developed a novel, semi-automatic mode that allows users to focus primarily on functional characteristics of their design. For example, if the goal is to create a robot arm with 3 actuated joints, shoulder, elbow and wrist, the user can simply specify how the joint motors should be positioned relative to each other. Our system then employs a computational algorithm to auto-complete the design. This is achieved by searching for a sequence of modules that result in appropriate mechanical structures connecting the shoulder elbow, and wrist motors.

We use a heuristic-guided tree search algorithm to auto-complete designs. As illustrated in Fig. 5, starting from a *root motor*, our computational system creates a tree of possible designs in a recursive manner. Briefly, nodes correspond to modules, edges describe how compatible modules connect to each other, and the path from each node to the root corresponds to an intermediate robot design. Leaf nodes of the search tree correspond to designs that end with a *target motor*. The goal of our design auto-completion algorithm is to find the leaf node corresponding to a relative placement between the root and target motors that is as close as possible to what the user specified (e.g. placement of elbow motor relative to shoulder motor, or wrist motor relative to elbow motor). As described in the remainder of this section, we explore several heuristic functions that guide the search process in order to make it computationally efficient, and therefore suitable for interactive design.

Search tree: The search tree T is a collection of nodes and edges, $T = (N, E)$. Each node N_i represents a module m_i and is defined as:

$$N_i = (m_i, N_i^p, \{_1N_i^c, \dots, _nN_i^c\}) \,, \tag{3}$$

where m_i is the module corresponding to node N_i (defined by eq. 1), N_i^p represents its parent node – the node it originated from, and $\{_jN_i^c\}_{j=1}^n$ represents a set of n successive nodes called children nodes. Node N_i has a child node for every way in which module m_i can connect to any other module in the library. The child nodes themselves represent the modules that are to be connected to m_i, while the way in which the two modules are attached to each other is specified by the edges of the tree. In particular, an edge connecting nodes N_a and N_b is defined as:

$$E_{i,j,k}^{a,b} = \left(N_a, N_b, c_{i,j}^{a,b}, k\right) \,, \tag{4}$$

where a is the index of the parent node, b is the index of the child node, $c_{i,j}^{a,b}$ references the connection between their modules m_a and m_b at virtual pins $p_i^{m_a}$ and $p_j^{m_b}$ respectively (defined by eq. 2), and k indicates the index of a relative orientation from the set of feasible configurations for $c_{i,j}^{a,b}$. With the parent node N_a's position and orientation kept fixed, the child node N_b's relative placement is automatically determined such that the locations of the connection's virtual pins (i.e. $p_i^{m_a}$ and $p_j^{m_b}$) coincide in world coordinates, and the relative orientation between the two is set according to

$c_{i,j}^{a,b}[k]$. We note that nodes of the search tree are expanded as needed during the search process.

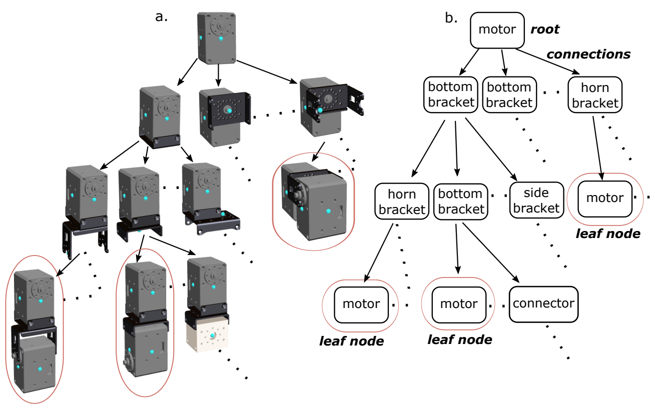


Fig. 5. The search tree originates at the root motor and enumerates all possible structural designs between the root and the target motor. The visualization of physical designs represented by the search tree and its corresponding schematic representation are shown in (a) and (b) respectively. Each branch represents a different design that results in a particular target motor configuration (encoded by its leaf node, highlighted with red oval).

Informed tree search: Since the depth of the search tree is potentially unbounded, and each node has a large branching factor (see Fig. 1(b) and Fig. 2(b)), the design space is vast. As a result, a brute force traversal of the search tree would be prohibitively expensive. We therefore propose an informed search process that accounts for the desirability and feasibility of a design. We leverage existing informed search methods in graph theory, namely the A* algorithm, and adapt it for interactive, user-driven design [18].

Informed search methods use problem specific knowledge to decide which nodes to expand while traversing the search tree. Heuristic functions that embed the target information are the most common form of domain knowledge used during the search. We define a heuristic function h that measures the promise of each branch in reaching the target motor:

$$h(N_i) = |x^{m_i} - x^{m_t}| \;, \tag{5}$$

where N_i represents current node of a branch, module m_i represents N_i's module, m_i's position $x_w^{m_i}$ is defined in eq. 1, and m_t is the target motor. h refers to the euclidean or straight-line distance between m_i and m_t. Note that h is an admissible heuristic because the shortest path between any two points is a straight line, so it cannot be an overestimate [18]. Apart from heuristics, other cost functions can be used to bias the search in finding designs with desired properties.

Desirability cost: We define a desirability cost that encodes user preferences regarding aesthetics and resource economy. The desirability of a design is measured in terms of the total number of modules used and the compactness of the resulting structure. A design that uses a lower number of modules may be more economical (and it is trivial to associate different costs to different types of modules). Similarly, a design that connects the root motor and target motor using an approximately straight structure might be more compact, and therefore more aesthetically pleasing. The desirability cost D of a branch in the search tree measures these two characteristics of its design, at its current node N_i.

$$D(N_i) = w_c * \delta(N_i) + \frac{\sum_{j=1}^{\delta(N_i)} dist(N_j, \overline{RT})}{\delta(N_i)} \;, \tag{6}$$

where w_c is a scaling weight and $\delta(N_i)$ is the depth of node N_i in the branch (with root node at depth 0). The first term therefore represents the total number of nodes (modules) between the root and the node. $\overline{RT}$ represents a line segment in 3D joining the root and the target nodes, and $dist(N_j, \overline{RT})$ measures the distance between the node N_j and $\overline{RT}$. This distance is zero if the position of N_j's module ($x_w^{m_i}$) lies on $\overline{RT}$, thereby implying a straight structure. The second term thus measures the average deviation of branch's module positions from $\overline{RT}$. The nodes with lower D are more desirable. D's role in the search is comparable to that of path cost g in A* search [18]. However, defining D allows us to present the users with intuitive handles to control the search output.

The total cost f of a node N_i is then determined as:

$$f(N_i) = h(N_i) + w_d D(N_i) \;, \tag{7}$$

where w_d is a user-defined weight, $h(N_i)$ is defined by eq. 5, and $D(N_i)$ is defined by eq. 6. h, D and f are calculated and stored for each node during the expansion process. The definition of node N_i is extended to store them.

$$N_i = (m_i, N_i^p, \{{}_1N_i^c, \ldots, {}_nN_i^c\}, h, D, f) \;, \tag{8}$$

where m_i, N_i^p, $\{{}_jN_i^c\}$ are defined as in eq. 3. h, D and f are defined as in eq. 5, 6, 7 respectively. Algorithm 1 describes our search process that uses f to determine which node to expand. The node with lowest f is expanded first. The *openset* keeps track of all the nodes yet to be expanded. Upon expansion, only the nodes free of collision are added to the *openset*, to ensure physical validity of the design. We use bullet collision engine [19] to compute collisions between modules. Expanding nodes with lowest f ensures that the branches whose structures extend towards the target motor position are expanded and traversed first in the tree. However, the function of an articulated link with multiple motors depends not only on the position of the motors, but also on their rotation axis. Hence, not all the designs that extend to the target motor position might function as desired. We therefore define an additional functionality cost.

Functionality cost: The functionality of each branch in the search tree is measured using the error between the position and orientation of the motor's axis at its leaf node and the user-specified target values. This error term determines whether the proposed design is acceptable. F is comparable to a termination criteria of conventional search methods.

$$F(N_i) = |x^{m_i} - x^{m_t}| + (1 - |\overrightarrow{a}^{m_i} \cdot \overrightarrow{a}^{m_t}|) \;, \tag{9}$$

where N_i is a leaf node, m_i represents the motor module associated with N_i, and m_t is the target motor which is placed in the design by the user. x^{m_i} and x^{m_t} denote positions of m_i and m_t respectively (eq. 1). $\overrightarrow{a}$ is a vector denoting the motor's axis of rotation. For certain applications, users might also care about the rotation of the target motor about it's rotation axis. This degree of freedom does not interfere with the functionality of a design, but it may affect its form factor. In such cases, F can be extended to account for this error in the orientation of the target motor:

$$F(N_i) = |x^{m_i} - x^{m_t}| + (1 - |\overrightarrow{a}^{m_i} \cdot \overrightarrow{a}^{m_t}|) + \Delta(q^{m_i}, q^{m_t}) \,, \tag{10}$$

where q^{m_i} and q^{m_t} are quaternions representing orientations of modules m_i and target motor m_t respectively (eq. 1). Their orientation difference $\Delta(q^{m_i}, q^{m_t}) = \cos^{-1}(scalar(q_d))$, where $q_d = (q^{m_i})^* q^{m_t}$ and q^* represents quaternion conjugate. F is defined by eq. 9 by default. In algorithm 1, as soon as a motor module node is encountered in the expansion process, instead of adding it to the *openset*, it is compared to the target motor using F to determine whether its branch represents a valid functional design. If the design is valid, the search displays it to the user and discards it otherwise.

Comparison to conventional search: When $w_d > 0$ in eq. 7, our search process (algorithm 1) becomes an A* algorithm that determines which nodes should be expanded by summing up the heuristic cost-to-goal estimate and the cost of intermediate designs [18]. Setting $w_d = 0$ converts the search into a greedy best-first search that only uses the heuristics for node expansion. Throughout the search process, we use our functionality metric F to keep track of the current best node. Instead of waiting for the search to find an optimal mechanical assembly, the design corresponding to the current best node is displayed and updated as the search progresses. This strategy produces various alternative designs for users to choose from as they are found, and it leads to lowers wait times that promote interactive design. Users can accept any of the designs produced by the system at any point in the search process. Keeping users in the loop in this manner allows us to also account for aesthetic preferences that may not be captured by the desirability cost. If users do not choose any design, we exit the search when the number nodes in *openset* exceeds a threshold, returning the design with minimum F. Inspired by the various alternative designs proposed during the search, users are free to change the placement of the root or target motors at anytime. The search process will account for this change and will be updated immediately. Users can further influence the search process by specifying different weights for the orientation or desirability objectives (eq. 7,10).

Algorithm 1: Interactive search for robot structure

```
input : Root motor ($r$), target motor ($t$), $max_N$, $w_d$
output: Designs connecting $r$ and $t$.
// start with root node
calculateCost ($N_r$); $openset \leftarrow N_r$
// keep track of the best design
$N_{best} = N_r$
while $0 < size(openset) < max_N$ do
    // select node for expansion
    $N_{current} = N$ in $openset$ with lowest $f$
    // node expansion
    $N_{new} \leftarrow \{_j N^c_{current}\}^z_{j=1}$
    for $i \leftarrow 1$ to $z$ do
        // check feasibility
        if $N_{new}(i)$ is free of collision then
            calculateCost ($N_{new}(i)$)
            // compare leaf node to target
            if $N_{new}(i) \rightarrow m = motor$ then
                if $F(N_{new}(i)) <= F(N_{best})$ then
                    $N_{best} = N_{new}(i)$
                end
            end
            // add to openset
            else
                $openset \leftarrow N_{new}(i)$
            end
        end
    end
end
// output best node
return $N_{best}$
Function calculateCost($N$)
    calculate $N \rightarrow h$, $N \rightarrow D$, $N \rightarrow f$
end
```

V. Results

Our design system allows casual and expert users alike to easily design customized robotic devices that range from wheeled vehicles to legged robots (Figures 6, 7 and 9). In this section, we evaluate our system by analyzing the abilities of the manual and automatic design modes. Further, we verify the feasibility of the designs made with our system by fabricating two very different robots – a wheeled robot with a manipulator arm that draws called "robo-calligrapher"', and a quadruped robot which we call "puppy".

Guided by visual suggestions provided by our system, the manual mode we developed is a powerful tool that allows users to explore the design space. Fig. 7 shows several different designs for two types of quadrupeds. Each of these designs took a matter of minutes to create. When considering the number of motors that can be used for each limb, their placement and direction of rotation axes, configurations of the legs and how they attach to the body, the design space is very rich. The ability to quickly create or alter designs is therefore very important in exploring the relationship

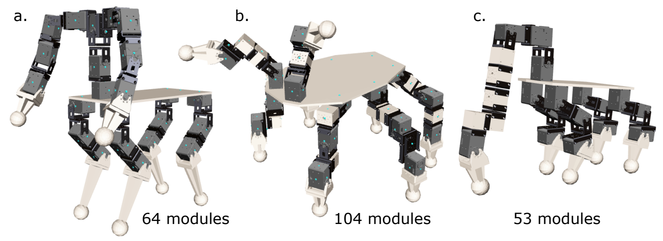


Fig. 6. Various robots designed with our system: (a) a centaur, (b) a hexapod with arms, and (c) a pentapod. (a) was designed in the manual mode, (b) was designed using the search, and (c) was designed using both modes. Its short legs were designed in manual mode while the longer limb was designed with the search process. Each design is composed of off-the-shelf modules and custom 3D printed modules (shown in white). The total number of modules used in each design (mentioned below) indicate their complexities.

between the form and function of the robot. To further speed up design processes, our computational system allows users to copy-and-paste different parts of a design in order to mix and match features from different robots. This functionality is demonstrated with the top-left design in Fig. 7, where the robot employs two legs from the spider design, and two from the mammal design. Upper bodies with an arbitrary arrangement of manipulators can also be easily designed, as demonstrated by the robots presented in Fig. 6.

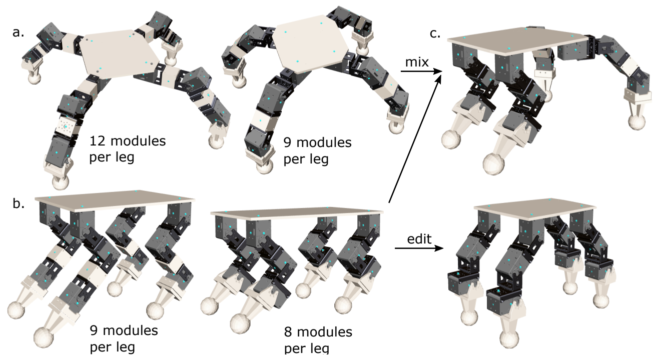


Fig. 7. Two types of quadrupeds inspired by spider-like legs and mammal-like legs are shown in (a) and (b) respectively. Each design uses 3 motors per leg, but uses different number of modules per leg (mentioned below each design). These modules are also used in different configurations resulting in diverse looks and motion behaviors. Designs in (c) were made by editing and mixing designs from (a) and (b).

Fig. 4 illustrates the process of creating a dinosaur robot using our design auto-completion mode. For designs such as this, the large number of required components can make the manual mode too tedious and time consuming. Our semi-automatic mode proposes designs that attach motors to each other through a series of modular connectors. Each design is generated to ensure that the final placement and orientation of the motors is as close as possible to user-specified configurations. Fig. 8(a), for example, shows various design alternatives suggested by our search algorithm. By specifying preferences regarding the number of modules, aesthetics (eq. 7), and motor orientations about their rotation axes (eq. 10), users can intuitively influence the result of the search process.

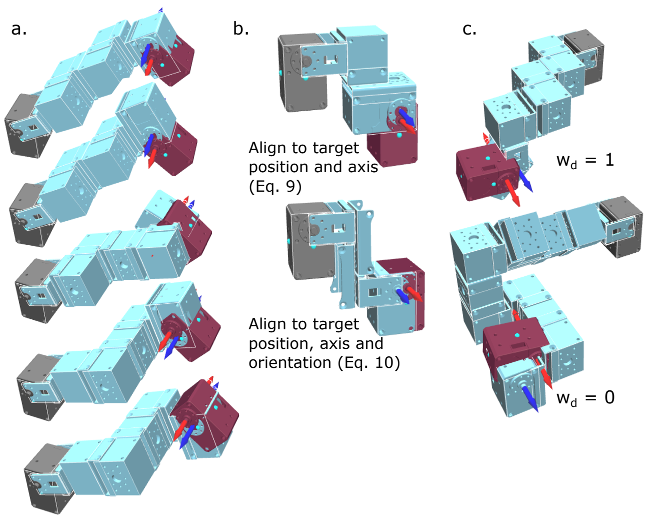

a.
b.
c.
Align to target position and axis (Eq. 9)
Align to target position, axis and orientation (Eq. 10)
$w_d = 1$
$w_d = 0$

Fig. 8. (a) The search proposes various design alternatives (shown in light blue) that connect the root motor (in gray) to the target motor (in dark red) in desired configuration as closely as possible. The motor axis alignment for the target motor (red axis) and the proposed structure (blue axis) is also illustrated. (b) The resultant designs vary based on F cost used for comparing the resultant motor configurations of the proposed designs to the target motor. (c) The importance of desirability cost is shown here. When w_d in eq. 7 is set to zero, the resultant designs end up using more number of modules and are aesthetically less appealing.

Fig. 9 shows two fabricated prototypes of robots designed with our system. The body plates, wheels and feet were 3D-printed, while all connecting brackets are off-the-shelf, aluminum parts. The robo-calligrapher robot was designed to take high-level commands from a blue-tooth device. By controlling the velocity of the motors that are attached to the wheels, it can easily be commanded to move forward, backward and to turn. We further developed an application that translates a user sketch to a sequence of motor commands for its arm. We used inverse kinematics for this purpose. Our design system allowed us to experiment with the number of actuators used in the arm. We found that with 3 motors and off-the-shelf mounting brackets, its range of motion was too limited, and therefore it was not able to reproduce a significant number of sketches we provided to it. Based on this diagnostic, we decided to add two additional motors to the design.

The puppy robot was designed to walk forward and sideways. For this robot, the feedback obtained from the simulation environment and the efficient iterative design process enabled by our system were particularly useful. As shown in the accompanying video, our system makes it easy to experiment with different body proportions, types of end effectors and motor configurations. In addition to affecting the perceived motion style, these choices also affect the robot's ability to perform different motor tasks. For example, before converging to our final design, we created a few robots that were only able to walk forward, and not sideways (e.g. bottom left design in Fig. 7). As our accompanying video shows, the motions of the physical prototype match well the simulated results. We therefore find the outcome of this experiment very encouraging, since performing an equivalent exploration of the design space directly in hardware is tedious and much more time consuming.

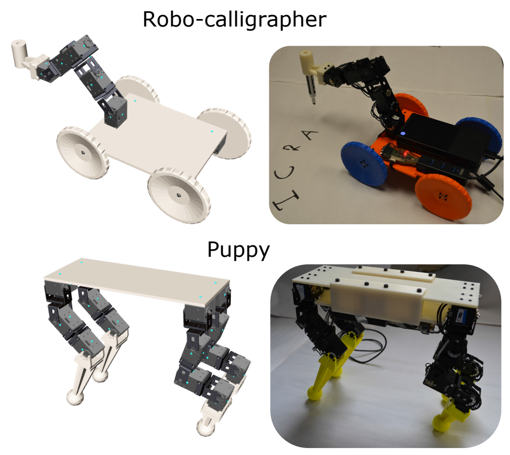


Fig. 9. The "robo-calligrapher" and "puppy" robot are shown here with their fabricated counterparts. We designed a special purpose end effector that served as a pen-holder for the robo-calligrapher. Accompanying video shows these robots in action.

VI. Conclusion and Future Work

We formalized a general and flexible design abstraction for on-demand generation of customized robotic devices. Based on this abstraction, we developed manual and semi-automatic approaches to allow users of our system to explore the space of robot designs in a visual environment. Through continuous feedback from physical simulations, our system allows users to iteratively improve their designs until individual needs and preferences are met. Our work aims to make robotics more accessible to casual users. We believe this is particularly important. As recent research shows, playing an active role in creating robotic devices for personal use increases our sense of self-agency, enhances the way we perceive them as well as the quality of our interactions with them [20], and may therefore accelerate their acceptance in our everyday life.

Our system allows users to efficiently create customized robots through design space exploration and simulation-based feedback, both of which result in faster design iterations. User design is supported through a manual mode that allows forward design, and an auto-completion mode that further speeds up the design process. However, our auto-completion approach does not currently account for dynamics, external loads, or desired motion profiles that may be important for professional design of articulated robots. Ideally, future iterations of our computational design system will account for these wider design requirements. We also note that while the simulation-based feedback immensely helps in iteratively improving the design, translating a negative outcome observed in simulation to an appropriate change in the design may be difficult for novices. In the longer term, we therefore wish to automatically translate functional and behavioral specifications into co-designed hardware, sensing and control software systems – a goal shared by other researchers [8]. At the same time, we believe it is important to find the right balance between automation and user-in-the-loop design processes. We therefore aim to perform an extensive user study to assess the needs and preferences of novice, intermediate and experienced robot designers.

References

[1] N. Bezzo, A. Mehta, C. D. Onal, and M. T. Tolley, "Robot makers: The future of digital rapid design and fabrication of robots," *IEEE Robotics & Automation Magazine*, vol. 22, no. 4, pp. 27–36, 2015.

[2] V. Megaro, B. Thomaszewski, M. Nitti, O. Hilliges, M. Gross, and S. Coros, "Interactive design of 3d printable robotic creatures," *ACM Transactions on Graphics (TOG)*, 2015.

[3] Robotis, "Dynamixel x series," 2015 (accessed on 5-September-2016). [Online]. Available: http://www.robotis.us/dynamixel-xm430-w210-r/

[4] "Robot makers workshop," RSS 2016. [Online]. Available: http://www.seas.upenn.edu/~nicbezzo/RoMa2016/

[5] C. Hecker, B. Raabe, R. W. Enslow, J. DeWeese, J. Maynard, and K. van Prooijen, "Real-time motion retargeting to highly varied user-created morphologies," *ACM Trans. on Graphics(TOG)*, p. 27, 2008.

[6] E. A. Inc., "Sporepedia," 2009 (accessed on 5-September-2016). [Online]. Available: http://www.spore.com/sporepedia

[7] A. M. Mehta and D. Rus, "An end-to-end system for designing mechanical structures for print-and-fold robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, June 2014.

[8] A. M. Mehta, J. DelPreto, B. Shaya, and D. Rus, "Cogeneration of mechanical, electrical, and software designs for printable robots from structural specifications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2014.

[9] C. Onal, M. Tolley, R. Wood, and D. Rus, "Origami-inspired printed robots," *Mechatronics, IEEE/ASME Transactions on*, vol. PP, no. 99, pp. 1–8, 2014.

[10] A. Mehta, J. DelPreto, and D. Rus, "Integrated codesign of printable robots," in *Journal of Mechanisms and Robotics*, vol. 7, 2015.

[11] C. Sung and D. Rus, "Foldable joints for foldable robots," in *Journal of Mechanisms and Robotics*, vol. 7, 2015.

[12] Autodesk, "Tinkerplay," 2015 (accessed on 5-September-2016). [Online]. Available: http://investors.autodesk.com/phoenix.zhtml?c=117861&p=irol-newsArticle&ID=2026270

[13] VEX-Robotics, "Vex assembler," 2015 (accessed on 5-September-2016). [Online]. Available: http://www.vexrobotics.com/vexiq/software/vexassembler/

[14] K. Sims, "Evolving virtual creatures," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 15–22. [Online]. Available: http://doi.acm.org/10.1145/192161.192167

[15] P. C. Leger, "Automated synthesis and optimization of robot configurations: An evolutionary approach," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1999.

[16] H. Lipson and J. B. Pollack, "Towards continuously reconfigurable self-designing robotics." in *ICRA*. IEEE, 2000, pp. 1761–1766. [Online]. Available: http://dblp.uni-trier.de/db/conf/icra/icra2000.html#LipsonP00

[17] J. Auerbach, D. Aydin, A. Maesani, P. Kornatowski, T. Cieslewski, G. Heitz, P. Fernando, I. Loshchilov, L. Daler, and D. Floreano, "RoboGen: Robot Generation through Artificial Evolution," in *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, 2014, pp. 136–137. [Online]. Available: http://www.robogen.org/

[18] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, vol. 2.

[19] "Bullet physics library," 2015 (accessed on 5-September-2016). [Online]. Available: http://bulletphysics.org/

[20] Y. Sun and S. S. Sundar, "Psychological importance of human agency how self-assembly affects user experience of robots," in *2016 11th ACM/IEEE Intl. Conf. on Human-Robot Interaction (HRI)*, 2016, pp. 189–196.