

---

# Interactive Robogami : An End-to-End System for Design of Robots with Ground Locomotion

Journal Title  
XX(X):1-16  
©The Author(s) 2015  
Reprints and permission:  
sagepub.co.uk/journalsPermissions.nav  
DOI: 10.1177/ToBeAssigned  
www.sagepub.com/



Adriana Schulz<sup>\*1</sup>, Cynthia Sung<sup>\*1</sup>, Andrew Spielberg<sup>1</sup>, Wei Zhao<sup>1</sup>, Robin Cheng<sup>1</sup>, Eitan Grinspun<sup>2</sup>, Daniela Rus<sup>1</sup>, and Wojciech Matusik<sup>1</sup>

## Abstract

This paper aims to democratize the design and fabrication of robots, enabling people of all skill levels to make robots without needing expert domain knowledge. Existing work in computational design and rapid fabrication has explored this question of customization for physical objects but so far has not been able to conquer the complexity of robot designs. We have developed Interactive Robogami, a tool for composition-based design of ground robots that can be fabricated as flat sheets and then folded into 3D structures. This rapid prototyping process enables users to create lightweight, affordable, and materially versatile robots with short turnaround time. Using Interactive Robogami, designers can compose new robot designs from a database of print and fold parts. The designs are tested for the users' functional specifications via simulation and fabricated upon user satisfaction. We present six robots designed and fabricated using a 3D printing based approach, as well as a larger robot cut from sheet metal. We have also conducted a user study that demonstrates our tool is intuitive for novice designers and expressive enough to create a wide variety of ground robot designs.

## Keywords

interactive design, digital fabrication, data-driven methods, simulation, concurrent design

## Introduction

A long-held goal in the robotics field has been to see our technologies enter the hands of the everyman. With the increasing number of retailers selling robotic kits, as well as the development of robotic household products, this goal has recently started to become a reality. Unfortunately despite these advances, customizing robotic technology to individual needs remains a challenge. Robots are complex systems that tightly integrate mechanical, electronic, and computational subsystems. As a result, customization at anything more than a superficial level often requires a nonnegligible amount of engineering skill. And yet, in order for robots to be able to address the individual needs of their users, they must allow for personalization.

Traditionally, robot development is a challenging and time-consuming process involving many iterations of design and testing, even for skilled engineers. Designers who decide to tackle this challenge must be able not only to devise and integrate physical and computational subcomponents, but also to evaluate manufacturability, usability, reliability, and other practical issues. Projects often go through multiple prototypes before converging to a final design. Recent advances in rapid fabrication Yan and Gu (1996) have made creating complex 3D physical objects easier than ever, allowing people to realize their designs in hours or days instead of weeks or years. Among these, 3D printing has emerged as a method for creating general geometries quickly. Unlike traditional manufacturing, in which complex geometries require in-depth analysis to ensure fabricability Gupta and Nau (1995), 3D printing processes are independent of the fabricated object, meaning

that designers are able to instantiate increasingly complex geometries without corresponding increases in cost or fabrication time. As a result, multiple groups have started to investigate how mechanisms and linkages can be fabricated as single prints without requiring post-fabrication assembly by printing joints Cali et al. (2012); Fuge et al. (2015), full mechanisms Thomaszewski et al. (2014)), and robots Mavroidis et al. (2000); Megaro et al. (2015). However, when it comes to evaluation and design cycles, current design tools still present users with clear limitations, and the learning curve is steep for anyone who wishes to create a design from scratch.

In this paper, we explore an intuitive design tool that complements current rapid fabrication methods, with the goal of providing designers with a framework for rapidly exploring, evaluating, and realizing their robotic designs. We focus in particular on print and fold robots, robots whose mechanical parts are fabricated as flat sheets and folded into their final 3D form. The main challenges that we tackle with this system are threefold. First, the design space is large and there is a great amount of flexibility. For any given task, there are many robot designs that could satisfy the task requirements. A robot design tool must allow

---

<sup>1</sup>Massachusetts Institute of Technology, MA, USA

<sup>2</sup>Columbia University, NY, USA

\* The first two authors contributed equally.

## Corresponding author:

Adriana Schulz, Cynthia Sung, Massachusetts Institute of Technology, 77 Mass Ave., Cambridge, MA 02139

Email: aschulz@csail.mit.edu crsung@csail.mit.edu

users to make interesting design choices while presenting the design space in an intuitive and manageable manner. A second challenge is the interdependence of subsystems in a robotic design. A robot has a mechanical body that has a geometry, has a particular kinematic structure, and is made of certain materials; it has actuators, sensors, and electronics that control the robots interactions with the environment at the low-level; and it has software that provides its high-level behaviors. Each of these components affects the other. In this paper, we address the subproblem of geometry-gait interdependence for ground robots. A third challenge is the transition from a conceptual design to a physical robot. Any design that a user creates must be manufacturable, that is, each of its pieces must be able to be fabricated, and the disparate pieces must be able to be assembled. Therefore, a robot design tool should validate not only a robots final form but also its fabrication plan.

In our work we tackle these challenges using a design by composition framework. This framework presents users with a database of usable robot parts which are independently manufacturable. Users can compose the parts together to form full designs and the validity and manufacturability of the design is maintained at every step. Components in the database can be parameterized to allow variability while keeping the space of possible designs manageable. Composition also addresses the challenge of concurrent geometry and gait design, since in addition to the geometric components, we also include gait components in the database. The tool incorporates simulations and interactive feedback with algorithms for design composition to streamline parts of the design cycle and guide the users exploration. It also includes algorithms for automatic generation of fabrication plans to provide users with full fabrication and assembly instructions for the robots mechanical body, required electronics, and control software. We have tested our methods by implementing this system, which we call Interactive Robogami, and using it to design and fabricate six robots of different forms. The robots are printed as flat fold patterns using a 3D printer and folded into their 3D shape. One was also fabricated in a larger form using sheet metal. We have also introduced the tool to eight novice users to investigate the expressiveness and usability of a design by composition framework.

## Related Work

Our work draws from a number of methods in computational design, fabrication-oriented design, and robot design.

*Computational Design* Recent works in computational design have proposed techniques to bridge the gap between aesthetic design and physical validity and functionality. For example, Whiting et al. [Whiting et al. \(2012\)](#) propose a method to optimize the geometric form of masonry buildings to shapes that are more structurally sound. Umetani et al. [Umetani et al. \(2012\)](#) describe a system for furniture design that guides users in creating physically valid solutions. Similar works have optimized the geometry of 3D printed shapes to make them stand [Prévost et al. \(2013\)](#), spin [Bächer et al. \(2014\)](#), and to control more generic mass properties [Musialski et al. \(2015\)](#). Shape optimization has also been used to allow complex functionality such as flying

capabilities [Umetani et al. \(2014\)](#). Our work is similar to these efforts since we propose a technique for designing functional robots that also guides the user in geometric modeling to optimize for ground locomotion.

*Fabrication-Oriented Design* Manufacturability has long been a concern in engineering design [Currier \(1980\)](#); [Gupta and Nau \(1995\)](#); [Wang and Sturges \(1996\)](#); [Khardekar et al. \(2006\)](#); [Patel and Campbell \(2010\)](#) and more recently has become an area of interest in the computer graphics community. Proposed fabrication-oriented systems include tools for plush toys [Mori and Igarashi \(2007\)](#), furniture [Saul et al. \(2011\)](#); [Umetani et al. \(2012\)](#); [Lau et al. \(2011\)](#); [Schulz et al. \(2014\)](#), clothes [Umetani et al. \(2011\)](#), inflatable structures [Skouras et al. \(2014\)](#), wire meshes [Garg et al. \(2014\)](#), model airplanes [Umetani et al. \(2014\)](#), twisty puzzles [Sun and Zheng \(2015\)](#), prototypes of mechanical objects [Koo et al. \(2014\)](#), and architecture-scale objects [Yoshida et al. \(2015\)](#). Our work is similar to these efforts in the sense that our tool constrains the design space to ensure fabricability. For fabrication, we use a method that combines 3D printing with origami-inspired fabrication methods: robots are 3D printed as flat faces that are then folded into their final shape [Deng and Chen \(2013\)](#). This method exploits origami-inspired fabrication which generates low-cost, lightweight structures [Sung and Rus \(2015b\)](#); [Mehta et al. \(2015\)](#); [Felton et al. \(2014\)](#); [Gong et al. \(2014\)](#); [Li et al. \(2010\)](#); [Chen et al. \(2013\)](#); [Hoover et al. \(2008\)](#); [Mitani and Suzuki \(2004\)](#); [Yim et al. \(2017\)](#); [Fitzner et al. \(2017\)](#) and combines it with the flexibility of additive manufacturing technologies to produce complex geometries and durable functional joints [Thomaszewski et al. \(2014\)](#); [Cañi et al. \(2012\)](#); [Ceylan et al. \(2013\)](#); [Bächer et al. \(2012\)](#); [Mavroidis et al. \(2000\)](#).

*Design of Robots and Functional Mechanisms* Design generation work often considers mechanical structure and task-specific movement in isolation, with mechanisms being designed independently of actuation [Zhu et al. \(2012\)](#); [Mehta and Rus \(2014\)](#); [Megaro et al. \(2015\)](#). Although fabrication plans for fully functional robots were outputted in [Mehta et al. \(2014\)](#), the designs were driven by mechanical considerations, and the electrical and software components generated in a post-processing stage. On the other side of the spectrum are work in generating driving mechanisms that match desired input motions for toy designs [Zhu et al. \(2012\)](#); [Coros et al. \(2013\)](#); [Thomaszewski et al. \(2014\)](#), robots [Song et al. \(2015\)](#); [Megaro et al. \(2015\)](#), and other models [Fuge et al. \(2015\)](#). The combined geometry-motion design challenge, although acknowledged, is not addressed by these systems.

*Concurrent Design* Concurrent design of multiple coupled subcomponents is a common challenge when developing complex systems. Proposed methods for tackling this challenge include methods for combined structure and control optimization for mechatronic systems [Park and Asada \(1994\)](#); [Pil and Asada \(1996\)](#); [Li et al. \(2001\)](#), cooptimization of dimensions and gait for a quadrupedal robot [Digumarti et al. \(2014\)](#), cogeneration of mechanics and actuation for printable robots [Lipson and Pollack \(2000\)](#); [Cheney et al. \(2013\)](#), and concurrent design of planar linkage structures and trajectories [Bächer et al. \(2015\)](#). These works search for

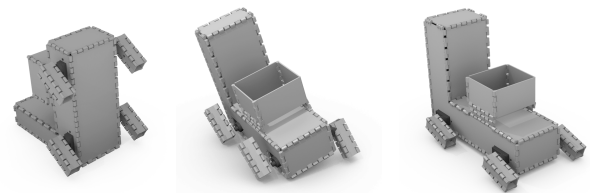
satisfactory designs under space and kinematic constraints. Our work is similar in that our robots require concurrent design of geometry and motion. However, our system handles 3D robot designs and additionally considers the stability, speed, and actuation requirements associated with a ground robot.

**Design by Composition** Compositional design tools have been shown in previous work [Funkhouser et al. \(2004\)](#) to facilitate user creativity while simplifying the design process for a wide variety of geometric objects such as furniture [Koo et al. \(2014\)](#); [Schulz et al. \(2014\)](#), clothing [McCann et al. \(2016\)](#), and electromechanical systems [Bezzo et al. \(2015\)](#); [Mehta et al. \(2015\)](#). The idea has existed at a hardware level in the robotics literature in the form of modular robots [Meng et al. \(2011\)](#); [Davey et al. \(2012\)](#); [Romanishin et al. \(2015\)](#), where individual hardware modules can be combined on-site to create new geometries and capabilities. However, since it is difficult to know what capabilities will be needed, these systems often suffer from the need to create ever more powerful modules. For robot design tools, allowing users to combine virtual modules that can then be fabricated according to their exact specifications provides designers with greater flexibility. In our approach we combine geometry and control design exploiting a component database in an interactive system.

**Origami-inspired Design** Origami inspired approaches have been widely used to allow low-cost and fast fabrication of 3D objects from 2D sheets ranging in scale from the micrometer and millimeter range [Yoon et al. \(2014\)](#); [Whitney et al. \(2011\)](#); [Ma et al. \(2016\)](#) to the decimeter and meter range [Tachi \(2011\)](#); [Sung and Rus \(2015a\)](#). Creating robots using a print and fold process accelerates the fabrication and enables the construction of strong and lightweight structures well suited for robot designs, but it also presents additional fabrication constraints that complicate the design process. Previous work has leveraged the transformation abilities of folded structures to create robots that crawl [Felton et al. \(2014\)](#); [Firouzeh and Paik \(2015\)](#), jump [Zhakypov et al. \(2015\)](#), or navigate obstacles [Lee et al. \(2013\)](#), but most of these designs are single instance designs that have been manually optimized. Work in [Mehta et al. \(2015\)](#) explored composition of print and fold robots but did not present any validity guarantees on the fold patterns. In our system, using a combination of folding and 3D printing [Deng and Chen \(2015\)](#) allows us to leverage the flexibility of an additive manufacturing process to simplify fold pattern design while maintaining the speed and material efficiency of folding.

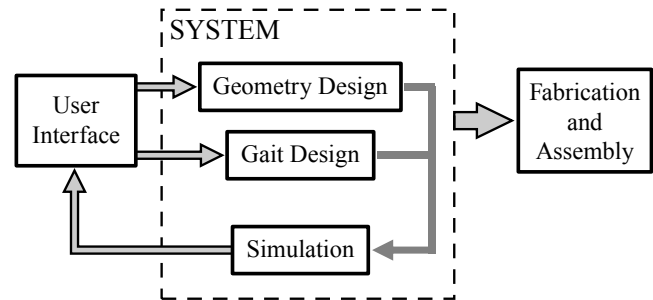
## Design Decisions

One of the main challenges in robot design is the interdependence of the geometry and motion. A typical design process for a robot starts with geometry design, followed by actuator selection, then controller design [Song et al. \(2015\)](#); [Megaro et al. \(2015\)](#). However, minor changes in geometry can drastically simplify actuator or control design. Take for example the robot in [Figure 1](#). This four-legged robot was designed by a user to deliver ice cream. In this task, speed is essential to ensure the ice cream does not melt. In addition, the robot must be steady enough that the ice cream does not spill on the way to its destination. In general,



(a) Original speed: 32.59 mm/s (b) Modified gait speed: 69.29 mm/s (c) Modified geometry speed: 99.76 mm/s

**Figure 1.** A robot design that topples while walking (a) can be modified to follow a gait that only wobbles slightly (b), but changing the geometry (c) allows the robot to move much faster and more steadily.



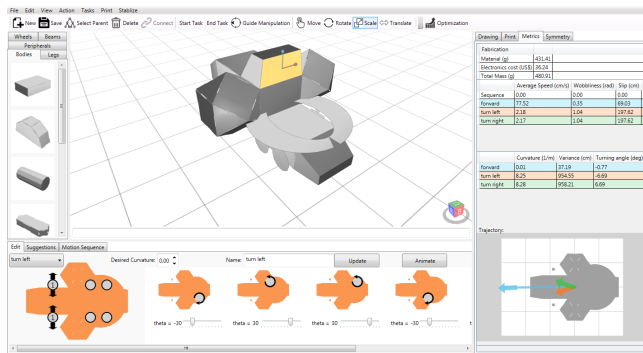
**Figure 2.** System diagram. Users interact with geometry and gait design tools. The designed models are simulated to provide feedback to the user. The user may iterate over the design before fabricating and assembling the model.

the fastest way for a [statically stable](#) four-legged robot to move is for all the legs to move simultaneously, resulting in a scooting motion. With the user's initial design ([Figure 1\(a\)](#)), though, this gait is unstable and the robot topples backwards. The user can find a gait with the left and right pairs of legs moving sequentially that allows the robot to move forward with only a slight wobble ([Figure 1\(b\)](#)). However, changing the geometry slightly allows the robot to succeed with the original gait and maintain its high speed without falling over ([Figure 1\(c\)](#)), resulting in a robot that is almost 50% faster.

We present an interactive system that allows designers to explore how both geometry and motion affect robot performance, enabling them to make these types of design choices. Our system contains tools for geometry and gait design, as well simulations for evaluating the model (ref. [Figure 2](#)). Users interact with the tool through a graphical user interface ([Figure 3](#)) in which they can visualize the models they create and receive real time feedback as to [how changes to the design affect the robot's performance](#). The system keeps track of the geometry and motion in order to output a full fabrication plan once the user is ready to build the design.

Our design choices were driven by three main objectives: 1) facilitate user creativity while maintaining wide range of accessibility, 2) enable concurrent design of motion and geometry, 3) guarantee fabricability and 4) provide real time feedback for ground locomotion design.

Since the design space of robots is too large to be explored interactively by casual users, we introduce a component library and an assembly based modeling tool to allow users to combine components in the database to create new robots.



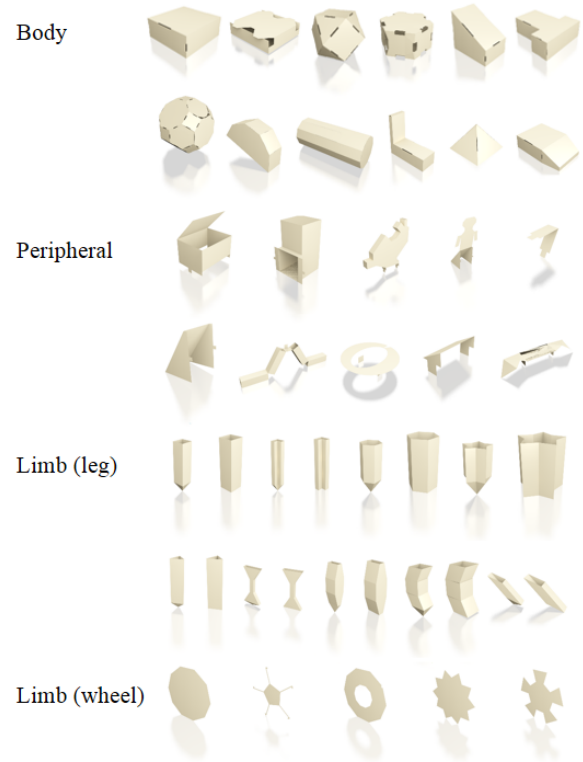
**Figure 3.** User interface. Icons that link to geometry components are displayed on the left and the gait design tool is on the bottom. Users design models by dragging components into the center canvas and editing them. Performance metrics for the design are shown on the right.

This method has been shown to facilitate user creativity while maintaining wide range accessibility [Funkhouser et al. \(2004\)](#). The components in the database are parameterized to allow variability while keeping the space of possible designs manageable. A linear parameterization simplifies the data representation and speeds up computation without compromising expressiveness.

In order to address the concurrent design problem, our database contains both geometric parts and motions and is supported by a grammar that dictates composition rules for both of these components. To this end we drew many insights from the experience of robotics engineers. We categorized parts into bodies, legs, and peripherals based on the functionality of components in existing robot designs. The joint controllers were chosen based on the physiology literature and parameterized according to when legs were expected to make contact with the ground [Ayyappa \(1997\)](#).

All the geometric components can be fabricated using our 3D print and fold method and the parameterization is designed so that this is preserved at any configuration. The composition tool uses the parametric representation to snap geometric component together constraining the composed models to a fabricable set of geometries. The composition rules are used to determine the electronic components needed in the assembly and the parametric motion representation is used for automatic generation of the control software. This allows the system to automatically generate a full fabrication plan for every composed model.

Optimization and simulation tools are used to guide users in realizing a stable ground robot [as will be detailed in a later sections](#). The parametric representation defines the design space for a fixed topology over which optimization can occur. Nine performance and fabrication metrics are computed in real time when a user creates or modifies a robot and are displayed to the user. Users can optimize designs relative to these metrics by altering geometry, gait, or both. We chose metrics that are commonly of interest to roboticists designing ground behavior. These objectives allow inexperienced users to optimize their designs to follow trajectories, minimize fabrication cost, maximize speed, and overall make more intelligent design tradeoffs than when using single-metric systems often found in the literature. During a pilot study, we found that users preferred some



**Figure 4.** Geometry components by category. Our system's database contains 45 components: 12 bodies, 23 limbs, and 10 peripherals.

metrics (e.g., speed, turning angle) and found others to be unintuitive (e.g., forward travel) and we incorporated this feedback into our user interface design.

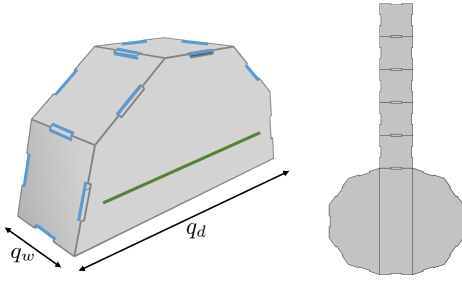
## Geometric Design

Our assembly based modeling tool allows users to create new designs by composing parts from a collection of parametric components and manipulating their shape parameters.

### Geometry Components

We have created a database of robot components to be used in design. These components are classified into three categories: robot bodies, limbs (wheels and legs), and peripherals, shown in [Figure 4](#). The database contains a total of 45 components, including 12 bodies, 23 limbs, and 10 peripherals.

All the components in the database are parameterized to allow structure-preserving variations. They were created using a system similar to that in [Mehta and Rus \(2014\)](#). Each component  $M$  contains a set of shape parameters  $\mathbf{q}$  and a corresponding feasible set  $\mathcal{Q}$ , which together describe the viable component geometries (see [Figure 5](#)). Each vertex on the geometry is written as a linear function of the shape parameters  $\mathbf{q}$ . Components are also annotated with fabrication rules for our 3D print and fold method. This includes a 2D representation of the fold pattern, consisting of a set of 2D faces and connection information between their edges that indicate where folds are located in the design. The 3D and 2D representations are coupled so that they are simultaneously updated as parameters are changed.



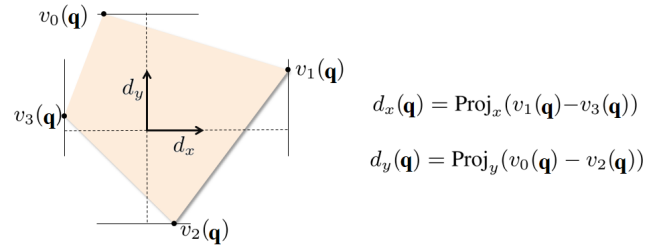
**Figure 5.** One of the geometry components in the database, with both its 3D shape and 2D unfolding. The component has parameter values of diameter  $q_d$  and width  $q_w$ . The green line indicates a functional patch, and the blue lines are static patches.

In order to allow the assembly based modeling to work with our fabrication method, each component  $M$  is also annotated with a set of connecting patches  $p_i$  that indicate where components are allowed to be connected together. Patch representations are also a function of the shape parameters  $\mathbf{q}$  and contain orientation information dictating the direction of connection. We define two types of patches. Static patches are edges where rigid attachments can be made. Because we use an origami-inspired fabrication approach, components can only be attached along folds. When components are attached along static patches, connecting hinges are added to the corresponding edges in the 2D unfolding. Functional patches indicate where articulated connections occur. These are either single points on the centers of wheels and legs or line segments along the sides of the bodies. When components are attached along functional patches, servomotors are added to actuate the articulations.

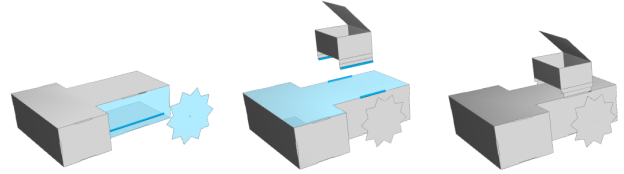
### Geometry Manipulation

Components can be modified based on the parametric representation. Allowable manipulations include translation, rotation, and dimension scaling. In order to maintain a linear representation for the geometric components, rotation is restricted to a global rotation of the model and involves modifying the linear functions that represent the vertex locations and connections.

All other manipulations correspond to an optimization of the parameter values. For example, in order to scale the dimensions of a face, users select that face and click and drag on one of the two orthonormal control axes that appear (Figure 3). The dimension in the selected direction,  $d(\mathbf{q})$ , is expressed as a function of the design parameters  $\mathbf{q}$  using the distance between the extreme vertices in this direction (see Figure 6). Resizing involves finding a feasible solution,  $\mathbf{q} \in \mathcal{Q}$ , such that  $\|d(\mathbf{q}) - (d(\mathbf{q}_{current}) + K)\|_2$  is minimized, where  $\mathbf{q}_{current}$  are the parameter values at the current configuration and  $K$  is the resizing amount defined by the user's dragging. Since there might be many feasible solutions to this problem, the system finds the one that keeps the model as close as possible to its original state. We express the distance to the current configuration as  $\|D(\mathbf{q}) - D(\mathbf{q}_{current})\|_2$ , where  $D$  is a matrix created by stacking the directional dimensions of all faces and represents the current



**Figure 6.** Given an orthonormal orientation  $(x, y)$ , the dimension of the part in each direction is written as a function of  $\mathbf{q}$  by using the parametric representation of the extreme vertices.



**Figure 7.** As the user drags in components, the UI highlights the patch pairs that will be connected. Limbs are attached to bodies along functional patches. All other components attach via static patches.

dimensions of the model. The closest model satisfying the user's input can be found by solving for

$$\mathbf{q}^* := \operatorname{argmin} \|d(\mathbf{q}) - (d(\mathbf{q}_{current}) + K)\|_2 + \alpha \|D(\mathbf{q}) - D(\mathbf{q}_{current})\|_2 \quad s.t. \quad \mathbf{q} \in \mathcal{Q} \quad (1)$$

where  $\alpha$  is a weighting parameter. Translation of parts is performed in an analogous manner.

### Geometry Composition

As users drag new components  $M^c$  onto the screen, the system proposes a connection to the working model  $M^w$ . It searches for the closest pair  $(p_i^c, p_j^w)$  of either static or functional patches and proposes to connect them by highlighting them in real time (ref. Figure 7). When the users are satisfied with a matching pair, they finalize the connection (Algorithm 1). This operation first rotates the added model so that the patches are properly aligned, then defines constraints on the parametric representation that snap the components into place and adds connection information.

The information added depends on the patch type. For static patches, constraints ensure that the pairs of patch edges remain coincident. The system also adds folds to connect the 2D unfoldings. For functional patches, the constraints enforce that patch regions intersect, i.e., that patch points are coincident when attaching limbs to limbs or that patch points lie on patch lines when attaching limbs to bodies. Joint information indicating that a servomotor should be added between these two components is also added. We define four types of joints: wheel joints (connecting wheels and bodies), single leg joints (connecting single-link legs to bodies), shoulder joints (connecting multi-link legs to bodies), and elbow joints (connecting links within the same leg). These connections define a kinematic chain that is used for gait design and simulation.

---

**Algorithm 1:**  $\text{Compose}(M^c, p_i^c, M^w, p_j^w)$

---

```

1 Rotate  $M^c$  so orientation of  $p_i^c$  matches  $p_j^w$ ;
  // Add constraints for connecting patches
2 if  $p_i^c$  and  $p_j^w$  are functional patches then
3   | Constrain center of  $p_i^c$  to lie on  $p_j^w$ ;
4 else
5   | Constrain the vertices of  $p_i^c$  and  $p_j^w$  to match;
6 end
  // Compose parameters  $\mathbf{q}$  and feasible set  $\mathcal{Q}$ 
7  $\mathcal{Q} \leftarrow \mathcal{Q}^w \cap \mathcal{Q}^c$ ;
8  $\mathbf{q} \leftarrow \mathbf{q}^w \cup \mathbf{q}^c$ ;
9 Update  $\mathbf{q}$  by solving Eq. (2);
10 Update kinematic tree;
```

---

Constraints on point and edge locations can be defined as linear constraints on the shape parameters  $\mathbf{q}$ . When connecting a limb to a body, a new parameter  $\mathbf{q}_c$  indicating the position of the leg along the side of the body is added. Linear constraints are then defined to enforce that the point patch on the limb is attached at this location. In addition, when multiple limbs are attached to the same functional patch, inequality constraints on  $\mathbf{q}_c$  are added to enforce that limbs do not collide during locomotion. We use a conservative approximation that ensures limbs will not collide in any gait.

Once constraints are defined, the system finds the valid parameters that keep the model as close as possible to the current configuration by solving for

$$\mathbf{q}^* := \operatorname{argmin} \|D(\mathbf{q}) - D(\mathbf{q}_{\text{current}})\|_2 \quad \text{s.t. } \mathbf{q} \in \mathcal{Q} \quad (2)$$

The solution of this optimization “snaps” the components into place.

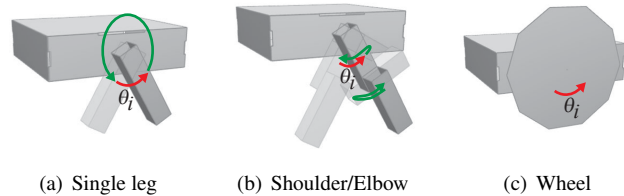
Since assembled models preserve the parametric representation, users can continue to manipulate shapes in the way described above after assembly. When one part is manipulated, the different parts of the composed model will update to satisfy the constraints imposed by the composition algorithm.

Finally, our system also allows users to define additional constraints on composed models. The user interface exposes a list of symmetry constraints that can be enforced or relaxed at any design stage. These include equalizing the length of all limbs, uniformly spacing all limbs along the sides of the body, equalizing thickness of all legs, and equalizing length of links on multi-linked legs.

These constraints are also defined as linear constraints on the shape parameters  $\mathbf{q}$ . We use the parameters  $\mathbf{q}_c$  that indicate the position of each limb along the side of the body to define a uniform spacing. Equalizing length and thickness of limbs and links is done by constraining their bounding box dimensions.

## Gait Design

Motion of a ground robot is dictated by its gait, which consists of the periodic motions performed by the robot’s joints as it locomotes. Similar to the geometry design



**Figure 8.** Joint controllers for each joint type. Controllers for every joint are separated into a step phase (shown in green) and a reset phase (shown in red). Users modify gaits by changing the  $\theta_i$  values for each joint and defining the step sequence.

process, gait design is done by manipulating parameters of gait components and composing them into a gait sequence.

## Gait Component

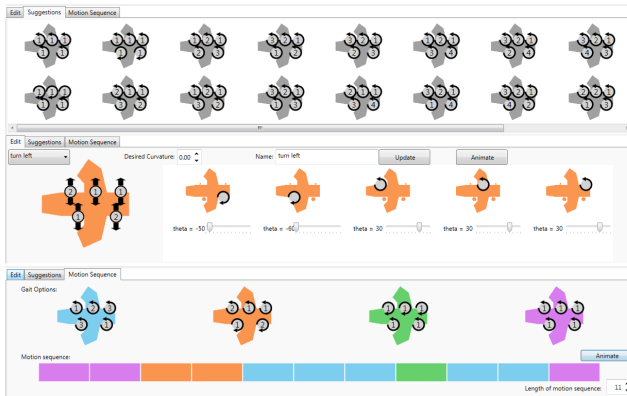
We have designed a gait that consists of two phases: a step phase and a reset phase. During the step phase, limbs take turns lifting off and re-situating. During the reset phase, all limbs move to shift the robot’s joint angles back to their pre-step configuration. In our parametric model, users can vary the size of the step for each limb and select the ordering of limb steps. The reset phase is modeled on these same parameters moving all of the limbs simultaneously after all the steps are performed, completing the gait cycle. When the robot is stable and does not rely on body contact with the ground to move, these phases correspond to the swing and stance gait phases commonly used in the literature [Ayyappa \(1997\)](#), the main difference being that in our robots, the stance phases for all limbs occur simultaneously.

We have designed a parametric controller for each joint type using this gait model. The joint motions that occur under each controller are shown in Figure 8. Green arrows indicate the joint trajectory during the step phase. Red arrows indicate the trajectory during the reset phase. Note that since wheels cannot lift off the ground once in contact, they do not experience a step phase. The combination of these controllers for a given robot topology defines a parametric gait component with gait parameters  $\mathbf{g}$  and a corresponding feasible set  $\mathcal{G}$ .

The gait parameters  $\mathbf{g}$  are as set of parameters for each limb  $(\theta_i, N_i)$ , where the value  $\theta_i$  controls the angle swept out by the  $i$ th limb during the reset phase, and the integer  $N_i$  defines the step order of the  $i$ th limb with respect to the step sequence. During the step phase, limbs can move in groups, therefore more than one limb can have the same step parameter  $N_i$  (see Figure 9). The feasible set  $\mathcal{G}$  includes constraints on  $N_i$  so that every step is associated with at least one leg. It also includes bounds on the absolute value of  $\theta_i$ . Negative values for  $\theta_i$  indicate a sweep in the opposite direction. Timing information for the gait is computed based on the parameter values and the servomotor’s maximum angular speed ( $3.3\pi$  rad/s).

## Gait Manipulation

The system provides users with a list of gait suggestions using a heuristic based on the stability and speed for a given topology. Stability is approximated as the distance between



**Figure 9.** Gait design interface. Users can choose among gait suggestions (top), edit the gait parameters (middle), and compose gaits into a gait sequence by dragging them into a timeline (bottom).

the projection of the robot’s center of mass on the ground plane and the boundary of its support polygon, and speed is approximated by counting the number of steps  $N$ . The system suggests gaits that maximize stability, then speed. It suggests only gaits that keep at least 3 limbs on the ground at all times.

Our system also suggests three standard gaits in which all limbs step simultaneously ( $N = 1$ ). The three variations consist of all positive values for  $\theta_i$ ’s (forward gait), negative values for  $\theta_i$ ’s along the left side of the body and positive values for  $\theta_i$ ’s on the right (rotation to the left), and positive values for  $\theta_i$ ’s along the left side of the body and negative values for  $\theta_i$ ’s on the right (rotation to the right).

Users design a gait by initially choosing a suggestion provided by the software. The users can adjust the gait parameters using the interface in Figure 9 (middle). The image on the left side shows the users the step sequence, while the smaller images to the right show the directions of movement of particular limbs. Users can adjust the step ordering or the sweep angle  $\theta_i$  for each leg by using the arrows and sliders.

### Gait Composition

Users can design multiple gaits by repeating the process for gait manipulation on the system’s suggestions and assigning each of the new gaits a unique name. They can compose a gait sequence by dragging the gaits onto a timeline (Figure 9 (bottom)). The system displays to the user each of the gaits they have designed, using colors to indicate the which gait takes place at each entry in the gait sequence. The system automatically adds a transition gait between any two gaits whose ending and starting joint configurations do not match. The transition gait consists of all joints moving between the ending joint angles of the previous gait to the starting joint angles of the next gait at the servomotor’s maximum speed ( $3.3\pi$  rad/s).

### Interactive Feedback and Optimization

Our system incorporates simulations and metric evaluations that enable users to evaluate their designs and make

additional changes before finalizing and fabricating the robot.

**Simulation** The robot models are simulated by discretizing time and computing the corresponding statically stable geometry at each time step. In particular, at each time  $t_i$ , the configuration of the robot is first computed by updating the joint angles at each of the robot’s joints. Next, the robot’s orientation is found by placing the lowest points of the robot on the ground and iteratively pivoting the robot about its contact points until it is statically stable. **The weight of actuators is accounted for using a point mass at the center of the joints.** The direction of rotation is determined by computing the effect of a downward gravitational force applied at the robot’s center of mass. The robot is statically stable when the projection of its center of mass onto the ground plane lies within its support polygon (i.e., the convex hull of the contact points with the ground).

We approximate friction by assuming that the robot has slipped as little as possible. At each time  $t_i$ , our system finds the contact points of the robot with the ground and identifies those points that were also in contact at time  $t_{i-1}$ . To compute the new pose of the robot, a least-squares rigid transformation between the locations of the contact points from the two time steps is computed and applied to the robot.

Since the robot may exhibit different behaviors over multiple gait cycles (e.g., toppling), the system performs the simulation until it has detected that the robot has reached a steady-state behavior. It detects this case by comparing the pose of the robot at the beginning of a gait cycle to its pose at the beginning of every previous gait cycle. If its pose differs by only a translation and a rotation about  $\hat{z}$ , then the robot has reached a steady state. The system stores information about the start time  $t_{i_0}$  and end time  $t_{i_f}$  for one iteration of the robot’s steady state behavior for metric evaluation. Any pose of the robot after this time can be calculated as a rigid transformation of a pose during this time period.

**Metrics** The robot models are simulated for each of the different designed gaits and the composed gait sequence. The simulations are used to compute metrics that provide the user with information about the robot’s expected performance. All metrics on designed gaits are computed for the robot’s steady state behavior.

The system evaluates the following metrics, which tell users about the robot’s overall performance.

1. Speed: The average speed  $V$  of the robot in steady state is computed as the distance traveled during one iteration of steady state behavior, divided by the amount of time required to traverse that distance:

$$V = \frac{1}{t_{i_f} - t_{i_0}} \sum_{i=i_0+1}^{i_f} \|\mathbf{p}(t_i) - \mathbf{p}(t_{i-1})\|_2$$

where  $\mathbf{p}(t_i)$  is the position of the robot’s center of mass at time  $t_i$ . Speed is reported in mm/s.

2. Wobbliness: The average wobbliness  $W$  of the robot is computed as the amount of orientation variation the robot experiences over one iteration of steady state behavior:

$$W = \frac{1}{i_f - i_0} \sum_{i=i_0+1}^{i_f} |\Delta\theta(t_{i-1}, t_i)|$$

where  $\Delta\theta(t_{i-1}, t_i)$  is the combined yaw and pitch angular change between time  $t_{i-1}$  and time  $t_i$ . Wobbliness is reported in rad.

3. Slip: The average slip of the robot is the root mean square of errors between contact points during the final reorientation step of the simulation.

$$S = \frac{1}{i_f - i_0} \sum_{i=i_0+1}^{i_f} \sqrt{\frac{1}{n_i} \sum_j \|\mathbf{c}_i^j - \mathbf{c}_{i-1}^j\|_2^2}$$

where  $n_i$  is the total number of contact points that remain in contact between times  $t_{i-1}$  and time  $t_i$ , and  $\mathbf{c}_i^j$  is the location of one of those contact points  $j$  at time  $t_i$ . Slip is reported in mm.

The following metrics are also computed for each of the designed gaits to provide the user with additional information for composition.

1. Angle of Rotation: The robot's change in heading is computed over one gait cycle.

$$\Delta\phi = \frac{1}{i_f - i_0} (\phi(t_{i_f}) - \phi(t_{i_0}))$$

where  $\phi(t_i)$  is the heading of the robot at time  $t_i$ . Angle of rotation is reported in degrees.

2. Curvature: The average curvature of a gait is the reciprocal of the robot trajectory's radius of curvature and is computed as

$$\rho = \frac{2 \sin(\Delta\phi/2)}{\|\mathbf{p}(t_{i_f}) - \mathbf{p}(t_{i_0})\|_2}$$

Curvature is reported in 1/m.

3. Variance: The variance of a trajectory is calculated assuming that the robot ideally follows a circular path with curvature  $C$  and uses the perpendicular error from that path.

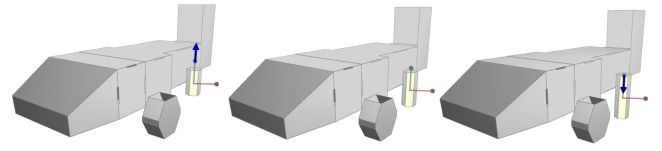
$$E = \frac{1}{i_f - i_0} \sum_{i=i_0}^{i_f} \|\mathbf{p}(t_i) - \hat{\mathbf{p}}(t_i)\|_2^2$$

where  $\hat{\mathbf{p}}(t_i)$  is the closest point to  $\mathbf{p}(t_i)$  on the circular trajectory. Variance is reported in  $\text{mm}^2$ .

In addition to these performance metrics, the system computes fabrication metrics related to the robot geometry and kinematics. These include:

1. Fabrication Cost: The fabrication cost is computed as the mass in grams of nonsupport material required to print the robot body.
2. Electronics Cost: The cost of electronics is the combined cost of the servomotors, the microcontroller, and the battery.
3. Total Mass: The total mass of the robot in grams is calculated as the combined mass of the printed body and the electronics.

**Feedback and Guidance** Our system computes the metrics at interactive rates and exposes them to the user as feedback during the design process. A tab on the right side



**Figure 10.** The system provides guidance arrows for users who want to make local geometry optimizations. The up and down arrows indicate that the user should lengthen and shorten the leg respectively. No arrow indicates that the part dimension is already at a local optimum.

of the UI displays each gait design's metric values, colored according to gait. It also shows an overhead view of the trajectory of the robot's center of mass for each gait. An arrow at the end of the trajectory indicates the robot's final heading. As the user manipulates the model and composes parts, the metrics and trajectories update automatically to reflect the changes. Users can also visualize an animation of any of the designed gaits in order to have a more comprehensive understanding of the robot's motion.

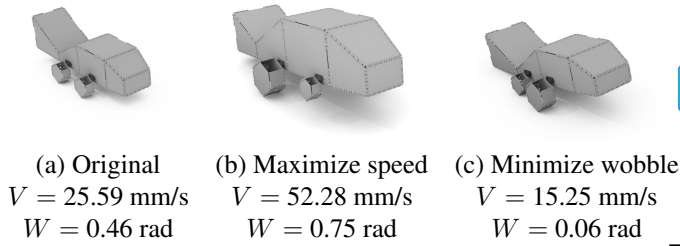
Our system also provides guidance to the user on how to manipulate model dimensions. To activate this feature, users select a metric value to improve (i.e., increasing speed, decreasing wobble, decreasing slip, etc.) and turn guidance on. When they next choose a face on the model to scale or translate, the system displays arrows on the control axes indicating which direction to change the dimensions to improve that value (Figure 10). The system uses finite differences to determine how much the metric improves for each manipulation axis and draws an arrow if the improvement is above a given threshold. The guidance can be performed for metrics on individual designed gaits or for the composed gait sequence, and it helps the user to find optimal part dimensions for that metric.

Finally, our system contains an automatic optimization method that will search for metric improvements in the full geometry space. The system uses NLOpt's COBLYA implementation Johnson (2014); Powell (1994) to search for parameter changes that achieve a local optimum of the user-chosen metric. For example, when the four-legged fish design in Figure 11 is optimized for speed, the system increases the overall size of the robot and two of the legs (Figure 11(b)). Because of legs on the same side of the robot are constrained to be far enough apart that they will never collide, and since robot speed is most affected by the maximum leg length, having one long leg and one shorter leg yields a higher speed than having two legs of equal length. The same model can also be optimized for minimum wobbliness, in which case the model becomes flatter and the back legs become shorter than the front ones to account for the heavy tail (Figure 11(c)).

## Fabrication

Our system outputs a full fabrication plan for the robot that the user has designed. These plans include a mesh of the robot's body that can be sent to a 3D printer, a list of electronics and port connections, and software to load onto the microcontroller (see Supplementary Materials





**Figure 11.** Optimization results for a four-legged fish robot for different metrics. Maximizing speed increases the overall size of the robot. Minimizing wobble makes the robot shorter.

---

**Algorithm 2:** Fabricate( $M$ )
 

---

```

1 forall static patch pair  $(p_i, p_j)$  do
2   | Shrink connected faces along connecting edges;
3 end
4 Extrude all faces to 1 mm;
5 forall functional patch pair  $(p_i, p_j)$  do
6   | Add holes for servo mounts;
7 end
8 forall static patch pair  $(p_i, p_j)$  do
9   | Generate printable hinge at patch locations;
10 end

```

---

for example fabrication plan). The user is responsible for assembling the robot model.

### Fabricating the Robot Body

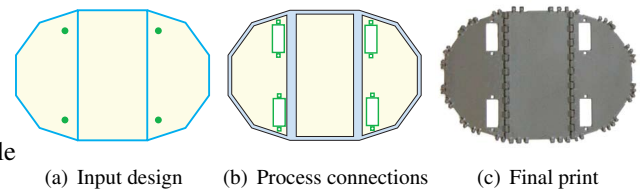
For the robot bodies, we use a combination of 3D and origami-inspired assembly methods that provide a balance between the versatility and rigidity of 3D printing and the speed of planar fabrication. Since the model that the user designs is already associated with a 2D fold pattern, our system can convert the pattern into a 3D printable mesh (Figure 12, Algorithm 2).

In order to construct the mesh, faces are first shrunk along the normals of connection edges to make room for folds and hinges are added to the mesh to enable folding. All the faces need to be shrunk before adding any hinges so that the added hinges do not interfere with each other. The faces are then extruded to 1 mm, which we chose as the thickness that produces rigid faces while still allowing almost  $\pi$  rad fold angles. At each of the functional connections, holes are added to the faces to make room for servomotors to be mounted. The hinge design is parameterized according to its location in the overall model, its length and its angle. It is also designed to snap together so that faces that are not adjacent in the 2D pattern can still be connected in the folded state. The hinges snap to the correct fold angle during assembly.

Finally, our system enables certain parts to be printed as 3D objects instead of folded ones. For example, volumetric feet on some of the legs are printed in a softer, rubbery material to increase friction with the ground. Wheels are printed thicker in order to increase their rigidity.

### Implementing the Motion Sequence

In order to implement the motion sequence, the robot requires actuators and control circuitry. An electronics plan



**Figure 12.** Our system automatically generates a 3D mesh for a foldable design. (a) The original design, with blue edges indicating static connections and green dots indicating functional connections. (b) Connections are processed by shrinking faces to make room for hinges and adding holes for servo mounts. (c) Hinges are added and the result is a complete mesh that is 3D printed.

is generated based on the kinematic tree of the robot design. Our control circuitry uses an Arduino Pro Mini and is based on a standardized electronics module described previously in Mehta et al. (2015). First, one servomotor is assigned to each degree of freedom. For this system, we use the Turnigy TGY-1370A servomotors, modified to also output a position signal. The robots are powered by a 3.7 V lithium ion battery. Connections between the servos, batteries, and the Arduino are assigned using a greedy approach that matches pin types. Connections such as those for power and ground can connect multiple components, while connections sending control signals to servomotors must be one-to-one. Depending on the number of servos and the available ports, more microcontrollers may be added.

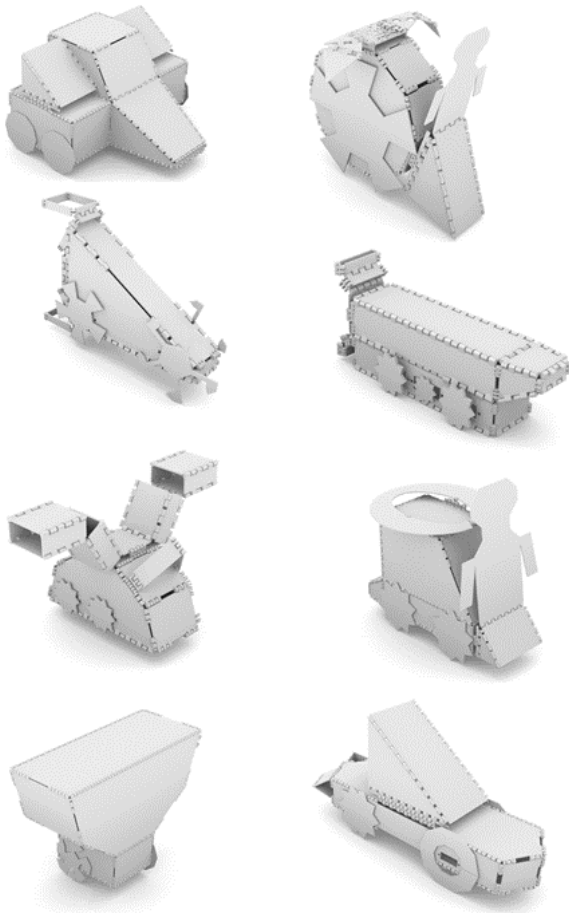
The system then generates control software by converting the timing and angle values from the individual joint controllers in the composed gait sequence into code for sending PWM signals to the robot's servomotors. Because our gaits are based on joint controller modules, the software for the robot can be generated using a software template. The gait parameters and the gait sequence that forms the desired trajectory are written to a gait definition file. The rest of the software is general code that implements the trajectory given the parameters.

### Assembly

The user assembles the robot by folding the body, attaching servomotors to the printed mounts, connecting the electronics parts, and loading the control software. Once the user turns the robot on, it immediately follows gait sequence designed in the system.

### Results

To test the usability of our system and the effectiveness of the combined geometry and gait design approach, we introduced our system to 8 users with no previous experience in robot design. The users were all engineering graduate students (3 female, 5 male) between the ages of 22 and 31. Four of the students had previous experience with CAD and modeling tools (SolidWorks, Blender, Maya, etc.). The users were given 20 min. of training in the system's features and were then asked to perform 2 tasks designed to evaluate the expressiveness of the geometry and gait design frameworks.



**Figure 13.** Cars designed by 8 different novice users after a 20 min. training session with the tool. Users were given 10 min. to design their car.

In addition to these tests, we also fabricated 6 robot models designed in the system.

### Geometry Design

In the first task, users were given 10 min. to interact with the geometry composition tool and create a car. Figure 13 shows the models that the users created. All of the models were functional vehicles that rolled forward without toppling. The results demonstrate a wide range of geometries that are achievable using our system. Interestingly, when body geometries that users wanted were not available as single components in the database, they were able to create new shapes by composing and rescaling individual bodies together. The users generally expressed satisfaction with the expressiveness of the system. An enthusiastic user designed an additional 18 robots with different levels of complexity (Figure 14). Each model took 3 to 25 min. to design.

### Gait Design

In the second task, users were given a robot design and asked to design a trajectory to navigate the robot through an obstacle course in the least amount of travel time. The users were given 15 min. to complete the task. Half of the users were allowed both to design gaits and to change the geometry parameters of the design, while the other half were restricted

to gait design. Figure 15 shows the trajectories designed by each user and the total time of each gait sequence.

The robot geometry provided was similar to the one described in Figure 1, which topples during forward gaits where all legs have equal values for  $\theta_i$ 's. Users who were not allowed to manipulate the geometry were able to explore the gait design tool creatively to reach the goal. Solutions included a combination of right and left rotation gaits (user 1), fixing the back legs and moving forward stably with only the front legs (users 2 and 6), and performing an 180 degree rotation before moving with a backwards gait towards the goal. To reach these solutions, users made extensive use of the interactive feedback during manipulation of gait parameters and gait sequence composition. The users reported that the most useful features were the animations and the trajectory visualization.

On the other hand, users who were not restricted to gait design edited the robot dimensions so that it would be stable during a forward gait with all legs moving simultaneously. The resulting gait sequences were significantly faster (40% time reduction on average) than the ones designed without geometry modification. These users referenced the metrics tab on the user interface when manipulating the shape to find more stable configurations. One user used the guide arrows to minimize the wobbliness metric.

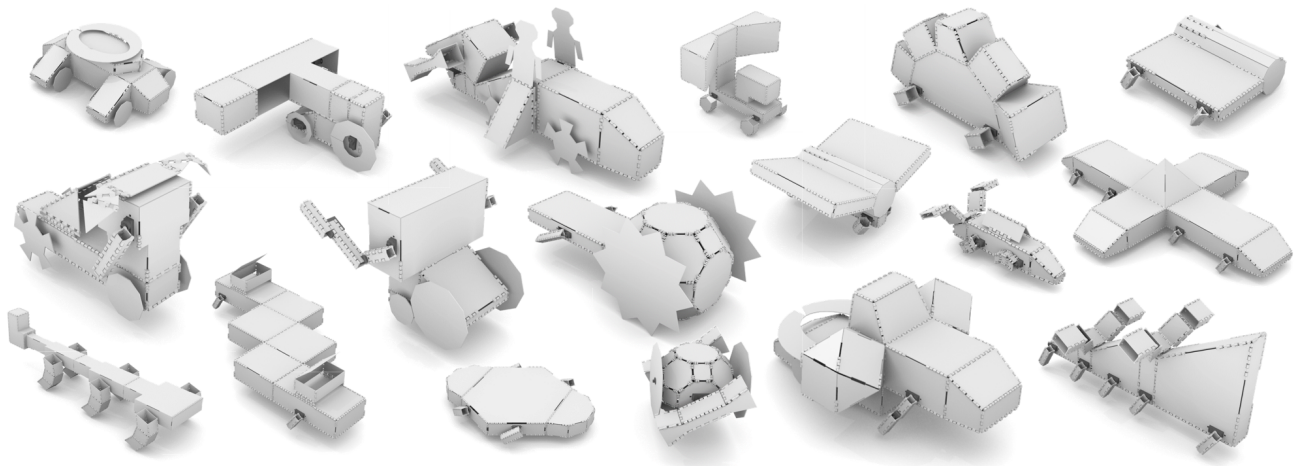
### Fabrication Examples

We tested the full design pipeline by composing and fabricating 6 robot models, shown in Figure 16. Figure 17 shows the electronics inside one of the models. They each took 10-15 min. to design, 3-7 hr. to print, and 30-90 min. to assemble (ref. Table 2). Each robot was fabricated and assembled successfully and was able to execute the gait designed in the system.

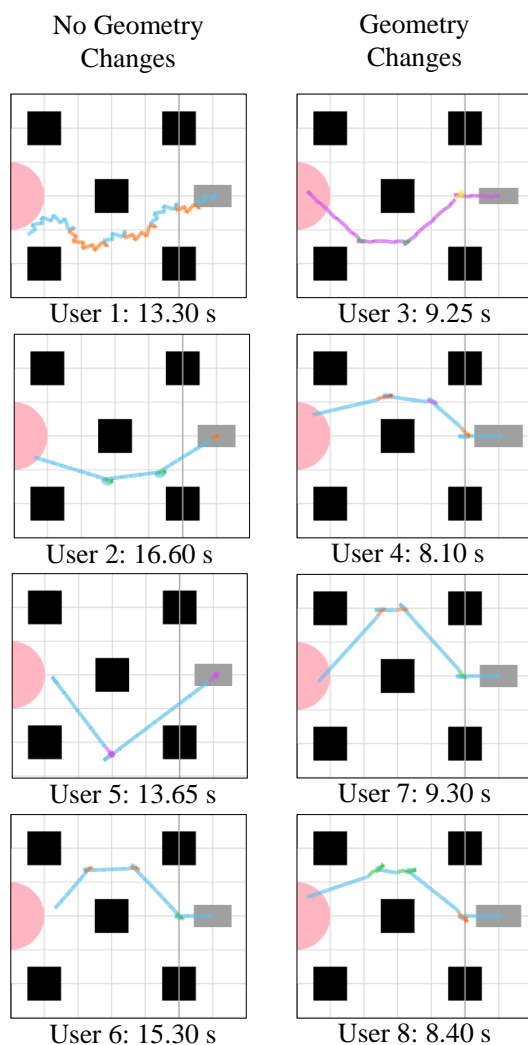
The robots demonstrate a range of locomotion types and gaits. The mouse, dragon, and ant are each robots that use multiple single legs to walk. The mouse uses a gait that steps with one leg at a time, while the dragon and ant move their legs in groups. The ant uses a standard tripod gait commonly seen in robotic hexapods. The wheeled car robot is an example of locomotion with a rigid body and wheels that rotate continuously to move it forward. Two of the robots demonstrate combinations of limb types. The house has two front legs and two back wheels. It moves each of its two legs forward individually but rotates its wheels simultaneously to shift forward. The monkey has double-link front legs and single back legs, and it uses a similar step sequence to the mouse but with a different ordering of legs.

The resulting robot trajectories were similar to those predicted by the system. Figure 18 shows frames from the simulation of the monkey compared with the physical robot at the same time. Each of the legs moves at the expected times and in the correct order. In addition, the robot dips forward when the third leg moves ( $t = 3.0$  s) in both the simulation and in the physical model. Both models have turned slightly to the right by the end of one gait cycle.

To test the degree of similarity of the physical robots with the simulations in the design tool, we tracked the movement of each of the robots over 10 gait cycles for 5 trial runs in a VICON motion capture system. We calculated the speed, mean curvature, and turning angle for each of



**Figure 14.** Gallery of designs created by a novice user after a 20 min. training session. Each of the models took between 3 and 25 min. to design and contains multiple components from the database.



**Figure 15.** Trajectories designed by users during the second task. Users who were allowed to make changes to the robot geometry were able to make the robot navigate the course about 40% faster on average.

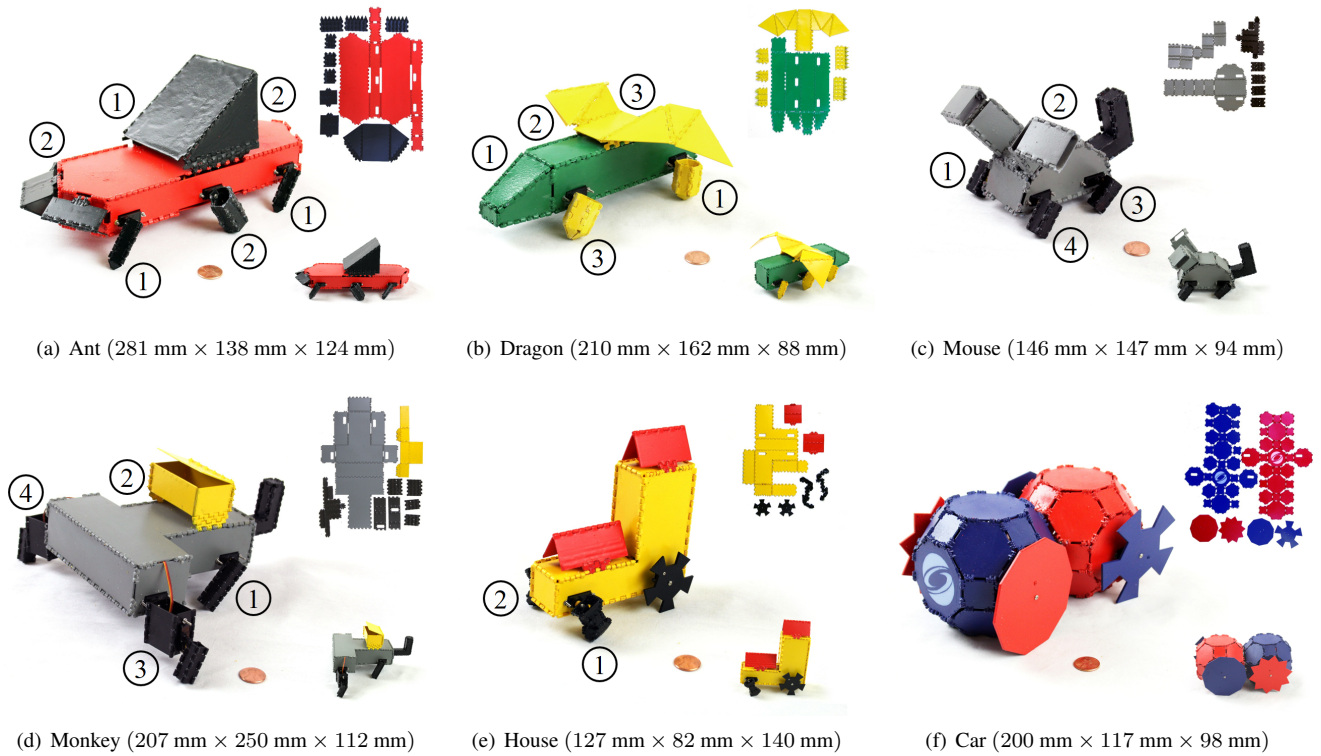
these trajectories. The mean metric values for each robot are given in Table 1. The car is the fastest robot, with a speed of

357.02 mm/s, while the monkey is the slowest of the robots, moving at a speed of 6.99 mm/s. All the experimental speed values are within 26% of the expected value, and turning angles are within 0.13 radians (7.35 deg.) of the expected turning angle. Larger errors in speed are correlated with larger errors in turning angle. During experiments, those robots experienced greater amounts of slip because their limbs were not moving completely synchronously and so dragged on the floor. While they were moving, there were also slight shifts of the positions of electronic internals, such as batteries and microcontrollers, that were not accounted for in the simulations.

Finally, we compared our 3D print and fold method to directly 3D printing the geometries of their folded state. Table 2 compares the printing time and material usage for each process. Our 3D print and fold technique yields a 73.2% reduction in 3D printing time and a 69.9% reduction in material usage on average. The difference in material usage is due to the support material that is necessary to print the robots in their 3D form. In addition to the fabrication efficiency, our method allows fabrication of closed shapes that are empty on the inside. All of our printed models would have been impossible to assemble as 3D models since there is no way to remove the support material from inside the body and to add the electronics.

### Extensions

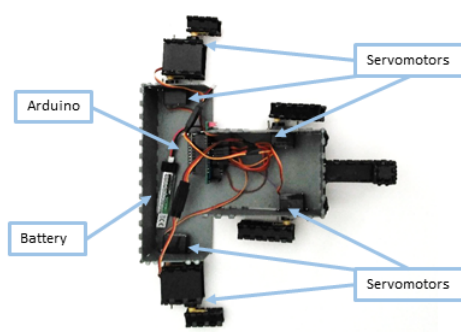
Because the system abstracts away the implementation details, users can design robots without specifying the exact fabrication approach. As a result, the models that users produce can also be fabricated using other origami-inspired methods, allowing our system to scale to robots beyond those models that can be printed in the 3D printer. For example, Figure 19 shows a foldable design for a two-legged, two-wheeled robot that was fabricated using both our 3D print and fold approach and by waterjetting and folding 3.18 mm thick aluminum sheet. The geometries of the two models were the same, up to a scaling factor, and the gaits were identical. In its folded state, the aluminum robot was 627 mm × 602 mm × 368 mm. Because of its size, we changed parts of the electronics design to enable larger motors and voltages to be used. Rubber sheet was attached



**Figure 16.** Six robots designed and fabricated using the system. Numbers over the limbs indicate the step sequence.

	Speed (mm/s)		Curvature (1/m)		Turning Angle(rad)	
	exp	sim	exp	sim	exp	sim
Ant	20.60	27.63	2.63	0.02	0.129	0.001
Dragon	14.39	14.31	1.9	1.79	-0.073	-0.047
Mouse	10.6	12.40	1.19	3.97	-0.045	-0.115
Monkey	6.99	9.17	6.05	4.43	-0.182	-0.106
House	14.56	14.01	1.60	0.21	-0.042	-0.004
Car	352.02	419.40	0.97	0.00	-0.051	0.000

**Table 1.** Simulated and experimental speed, curvature, and turning angle values for each of the six fabricated robots



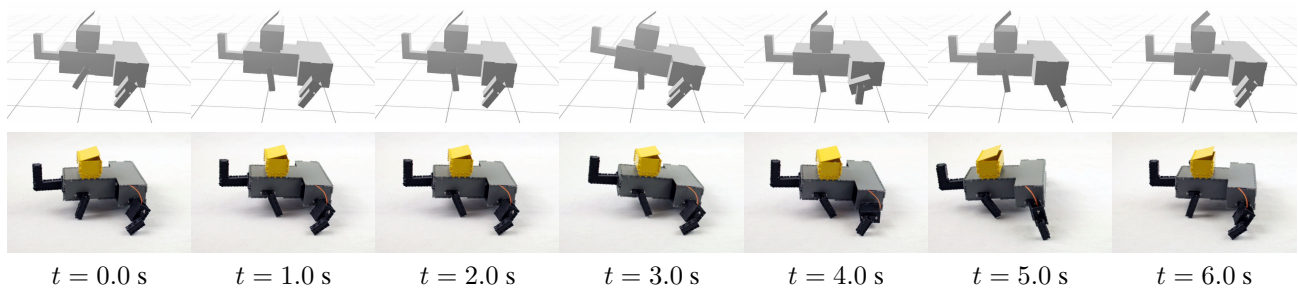
**Figure 17.** The electronics (servomotors, microcontroller, and battery) inside the Monkey example (Figure 16(d)).

to the bottoms of the legs and to the rims of the wheels to prevent the sharp corners of the metal from snagging on or scratching the floor. Table 3 shows a comparison of the robot to its 3D print and fold counterpart. The robot is able to carry a payload in excess of 14 kg. It walks at about half the speed of the 3D print and fold robot due to the gearing in the motors used.

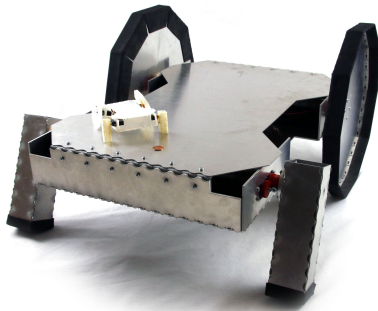
	Material Usage		Print Time		Assembly
	3D	2D	3D	2D	
Ant	2176 g	614 g	24.4 hr.	5.9 hr.	55 min.
Dragon	1795 g	517 g	24.2 hr.	6.9 hr.	45 min.
Mouse	1154 g	358 g	21.2 hr.	3.4 hr.	35 min.
Monkey	2779 g	499 g	33.7 hr.	5.2 hr.	45 min.
House	841 g	342 g	10.8 hr.	4.4 hr.	40 min.
Car	1955 g	661 g	18.1 hr.	6.5 hr.	90 min.

**Table 2.** Fabrication time and material usage for 3D printing vs. our proposed 3D print and fold method. The 3D print and fold method provides substantial time and material savings.

The system currently allows users only to make geometry and motion design decisions, but the automated generation of the fabrication plan also enables end users to more easily program robots for high-level behaviors. To test this feature, the aluminum robot was designed using the system for forward, backward, and left and right turning gaits. It was then instrumented with range sensors. We manually added some additional code to choose a gait depending on the sensor input but did not change any of the generated software. The resulting robot was able to navigate an



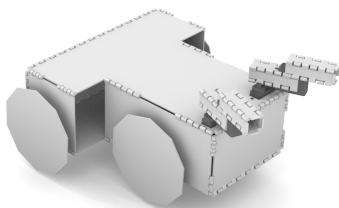
**Figure 18.** Comparison of monkey gait in simulation and from physical robot. The robot motion matches those from the simulations.



**Figure 19.** Large fold robot designed in system and fabricated from 0.125 in. thick aluminum sheet.

	Aluminum	Print and fold
Length	627 mm	103 mm
Width	602 mm	103 mm
Height	368 mm	63 mm
Mass	9.300 kg	75 g
Speed	18.91 mm/s	31.2 mm/s
Payload capacity	14.456 kg	188 g

**Table 3.** Specifications for aluminum and print and fold robot



**Figure 20.** Robot with gripper constructed by attaching 2-link legs to a functional patch on the top of the robot body

obstacle course of cardboard boxes without colliding with the environment.

It is worth noting that although our database was designed for the purpose of creating ground robots, many of the parts can also be used for other tasks. For example, legs and fingers on robots are essentially equivalent in all but intended functionality. Our system is flexible enough to accommodate manipulation tasks by simply allowing users to connect multi-link legs to functional patches on the tops or sides of robot bodies (ref. Figure 20).

### Limitation

Our system is subject to a few limitations. First and foremost, as with most data-driven methods, our system's design space is restricted to the designs that can be composed from the database using our grammar. Experts designers are needed to expand the set of building blocks from which users can compose their own designs. Methods for expanding the database that are accessible to novice users are needed to overcome this limitation.

With regards to the animation and simulation, our system currently only considers the geometry and kinematics of the robot. As a result, the electronics modules in all of our robots are the same. However, actuation is often a large challenge in robot design. Simulations that incorporate environmental or task constraints such as load, dynamic forces, and robustness in order to generate more applicable robot models are needed.

Finally, our user feedback is restricted to local guidance. Performance metrics are calculated for the current robot model, and guidance arrows provide information on how to make local design changes to improve metrics. Thus, the system is currently unable to accommodate situations where more global changes are needed (e.g., changing the dimensions of multiple parts simultaneously) or where additional parts should be added. It is also unable to determine when a user-designed model is unable to achieve the desired performance metrics given the parameter constraints. Methods that enable the system to provide this feedback will significantly strengthen the system.

### Discussion and Future Work

We have presented a system that allows novice designers to create functional robots using an assembly-based modeling approach. The system enables users to explore the space of geometries and gaits in their models in order to efficiently achieve desired performance metrics. It incorporates simulation and optimization methods to provide the user with feedback and guidance on how to modify their designs. It also takes care of the implementation details required for fabrication, allowing users to focus on the conceptual high-level design. We have demonstrated this end-to-end pipeline by designing and fabricating multiple robot models.

In the future, it would be interesting to expand the system to incorporate dynamic simulations and verification, as well as to address a wider variety of tasks. We showed with our obstacle course that the system is able to provide sufficient

feedback for standard planning tasks. However, real world robots must be able to interact with a physical environment. It would therefore be useful for the system to also be able to estimate metrics such as robustness, allowable load, and battery life.

Our results demonstrate that a design approach that simultaneously tackles geometry and motion aspects of a robot design can lead to better design performance. Using a tool that provides interactive feedback on performance metrics, novice users are able to efficiently explore the combined geometry and motion space to produce functional robots within a few hours. In the future, we believe that similar data-driven fabrication systems will enable mass customization and production of robots for the general public.

## Acknowledgements

Support for this project has been provided in part by NSF Grant Nos. 1240383 and 1138967, by the DoD through the NDSEG Fellowship Program. We are also grateful to the following people for resources, discussions and suggestions: Ankur Mehta, Joseph DelPreto, Jacob Haip, and Isaque Dutra.

## References

- Ayyappa E (1997) Normal human locomotion, part 1: Basic concepts and terminology. *Journal of Prosthetics and Orthotics* 9(1): 10–17.
- Bächer M, Bickel B, James DL and Pfister H (2012) Fabricating articulated characters from skinned meshes. *ACM Trans. Graph.* 31(4): 47:1–47:9. DOI:10.1145/2185520.2185543. URL <http://doi.acm.org/10.1145/2185520.2185543>.
- Bächer M, Coros S and Thomaszewski B (2015) LinkEdit: Interactive linkage editing using symbolic kinematics. *ACM Trans. Graph.* 34(4): 99:1–99:8. DOI:10.1145/2766985. URL <http://doi.acm.org/10.1145/2766985>.
- Bächer M, Whiting E, Bickel B and Sorkine-Hornung O (2014) Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph.* 33(4): 96:1–96:10. DOI:10.1145/2601097.2601157. URL <http://doi.acm.org/10.1145/2601097.2601157>.
- Bezzo N, Gebhard P, Lee I, Piccoli M, Kumar V and Yim M (2015) Rapid co-design of electro-mechanical specifications for robotic systems. In: *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, pp. V009T07A009–V009T07A009.
- Calli J, Calian DA, Amati C, Kleinberger R, Steed A, Kautz J and Weyrich T (2012) 3d-printing of non-assembly, articulated models. *ACM Trans. Graph.* 31(6): 130:1–130:8. DOI: 10.1145/2366145.2366149. URL <http://doi.acm.org/10.1145/2366145.2366149>.
- Ceylan D, Li W, Mitra NJ, Agrawala M and Pauly M (2013) Designing and fabricating mechanical automata from mocap sequences. *ACM Trans. Graph.* 32(6): 186:1–186:11. DOI: 10.1145/2508363.2508400. URL <http://doi.acm.org/10.1145/2508363.2508400>.
- Chen D, Sitthi-amorn P, Lan JT and Matusik W (2013) Computing and fabricating multiplanar models. In: *Computer Graphics Forum*, volume 32. Wiley Online Library, pp. 305–315.
- Cheney N, MacCurdy R, Clune J and Lipson H (2013) Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, pp. 167–174.
- Coros S, Thomaszewski B, Noris G, Sueda S, Forberg M, Sumner RW, Matusik W and Bickel B (2013) Computational design of mechanical characters. *ACM Trans. Graph.* 32(4): 83:1–83:12. DOI:10.1145/2461912.2461953. URL <http://doi.acm.org/10.1145/2461912.2461953>.
- Currier DW (1980) Automation of sheet metal design and manufacturing. In: *17th Conference on Design Automation*. IEEE, pp. 134–138.
- Davey J, Kwok N and Yim M (2012) Emulating self-reconfigurable robots-design of the smores system. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4464–4469.
- Deng D and Chen Y (2013) Assembled additive manufacturing—a hybrid fabrication process inspired by origami design. *Solid Freeform Fabrication* : 174.
- Deng D and Chen Y (2015) Origami-based self-folding structure design and fabrication using projection based stereolithography. *Journal of Mechanical Design* 137(2): 021701.
- Digumarti K, Gehring C, Coros S, Hwangbo J and Siegwart R (2014) Concurrent optimization of mechanical design and locomotion control of a legged robot. *Mobile Service Robotics: CLAWAR 2014* 12: 315.
- Felton S, Tolley M, Demaine E, Rus D and Wood R (2014) A method for building self-folding machines. *Science* 345(6197): 644–646.
- Firouzeh A and Paik J (2015) Robogami: a fully integrated low-profile robotic origami. *Journal of Mechanisms and Robotics* 7(2): 021009.
- Fitzner I, Sun Y, Sachdeva V and Revzen S (2017) Rapidly prototyping robots: Using plates and reinforced flexures. *IEEE Robotics & Automation Magazine* 24(1): 41–47.
- Fuge M, Carmean G, Cornelius J and Elder R (2015) The mechprocessor: Helping novices design printable mechanisms across different printers. *Journal of Mechanical Design* 137(11): 111415.
- Funkhouser TA, Kazhdan MM, Shilane P, Min P, Kiefer W, Tal A, Rusinkiewicz S and Dobkin DP (2004) Modeling by example. *ACM Transactions on Graphics* 23(3): 652–663.
- Garg A, Sageman-Furnas AO, Deng B, Yue Y, Grinspun E, Pauly M and Wardetzky M (2014) Wire mesh design. *ACM Trans. Graph.* 33(4): 66:1–66:12. DOI:10.1145/2601097.2601106. URL <http://doi.acm.org/10.1145/2601097.2601106>.
- Gong J, Wang J and Xu Y (2014) Paperlego: component-based papercraft designing tool for children. In: *SIGGRAPH Asia 2014*. ACM, p. 3.
- Gupta SK and Nau DS (1995) Systematic approach to analysing the manufacturability of machined parts. *Computer-Aided Design* 27(5): 323–342.
- Hoover AM, Steltz E and Fearing RS (2008) RoACH: An autonomous 2.4g crawling hexapod robot. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

- IEEE, pp. 26–33.
- Johnson SG (2014) The NLOpt nonlinear-optimization package. <http://ab-initio.mit.edu/nlopt>.
- Khardekar R, Burton G and McMains S (2006) Finding feasible mold parting directions using graphics hardware. *Computer-Aided Design* 38(4): 327–341.
- Koo B, Li W, Yao J, Agrawala M and Mitra NJ (2014) Creating works-like prototypes of mechanical objects. *ACM Trans. Graph.* 33(6): 217:1–217:9. DOI:10.1145/2661229.2661289. URL <http://doi.acm.org/10.1145/2661229.2661289>.
- Lau M, Ohgawara A, Mitani J and Igarashi T (2011) Converting 3D furniture models to fabricatable parts and connectors. *ACM Transactions on Graphics* 30(4): 85.
- Lee DY, Kim JS, Kim SR, Koh JS and Cho KJ (2013) The deformable wheel robot using magic-ball origami structure. In: *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, pp. V06BT07A040–V06BT07A040.
- Li Q, Zhang W and Chen L (2001) Design for control-a concurrent engineering approach for mechatronic systems design. *IEEE/ASME Transactions on Mechatronics* 6(2): 161–169.
- Li XY, Shen CH, Huang SS, Ju T and Hu SM (2010) Popup: Automatic paper architectures from 3d models. *ACM Trans. Graph.* 29(4): 111:1–111:9. DOI:10.1145/1778765.1778848. URL <http://doi.acm.org/10.1145/1778765.1778848>.
- Lipson H and Pollack JB (2000) Automatic design and manufacture of robotic lifeforms. *Nature* 406(6799): 974–978.
- Ma X, Vogtmann D and Bergbreiter S (2016) Dynamics and scaling of magnetically folding multi-material structures. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1899–1906.
- Mavroidis C, DeLaurentis KJ, Won J and Alam M (2000) Fabrication of non-assembly mechanisms and robotic systems using rapid prototyping. *Journal of Mechanical Design* : 516–524.
- McCann J, Albaugh L, Narayanan V, Grow A, Matusik W, Mankoff J and Hodgins J (2016) A compiler for 3d machine knitting. *ACM Trans. Graph.* 35(4): 49:1–49:11. DOI: 10.1145/2897824.2925940. URL <http://doi.acm.org/10.1145/2897824.2925940>.
- Megaro V, Thomaszewski B, Nitti M, Hilliges O, Gross M and Coros S (2015) Interactive design of 3D-printable robotic creatures. *ACM Transactions on Graphics (TOG)* 34(6): 216.
- Mehta A, Bezzo N, An B, Gebhard P, Kumar V, Lee I and Rus D (2014) A design environment for the rapid specification and fabrication of printable robots. In: *International Symposium on Experimental Robotics*.
- Mehta A, DelPreto J and Rus D (2015) Integrated codesign of printable robots. *Journal of Mechanisms and Robotics* 7: 021015.
- Mehta A and Rus D (2014) An end-to-end system for designing mechanical structures for print-and-fold robots. In: *IEEE International Conference on Robotics and Automation*. IEEE.
- Meng Y, Zhang Y, Sampath A, Jin Y and Sendhoff B (2011) Cross-ball: a new morphogenetic self-reconfigurable modular robot. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, pp. 267–272.
- Mitani J and Suzuki H (2004) Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph.* 23(3): 259–263. DOI:10.1145/1015706.1015711. URL <http://doi.acm.org/10.1145/1015706.1015711>.
- Mori Y and Igarashi T (2007) Plushie: An interactive design system for plush toys. *ACM Transactions on Graphics* 26(3): 45:1–45:8.
- Musialski P, Auzinger T, Birsak M, Wimmer M and Kobbelt L (2015) Reduced-order shape optimization using offset surfaces. *ACM Trans. Graph.* 34(4): 102:1–102:9. DOI:10.1145/2766955. URL <http://doi.acm.org/10.1145/2766955>.
- Park JH and Asada H (1994) Concurrent design optimization of mechanical structure and control for high speed robots. *Journal of dynamic systems, measurement, and control* 116(3): 344–356.
- Patel J and Campbell MI (2010) An approach to automate and optimize concept generation of sheet metal parts by topological and parametric decoupling. *Journal of Mechanical Design* 132(5): 051001.
- Pil AC and Asada HH (1996) Integrated structure/control design of mechatronic systems using a recursive experimental optimization method. *IEEE/ASME Transactions on Mechatronics* 1(3): 191–203.
- Powell MJ (1994) A direct search optimization method that models the objective and constraint functions by linear interpolation. In: *Advances in Optimization and Numerical Analysis*. pp. 51–67.
- Prévost R, Whiting E, Lefebvre S and Sorkine-Hornung O (2013) Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph.* 32(4): 81:1–81:10. DOI:10.1145/2461912.2461957. URL <http://doi.acm.org/10.1145/2461912.2461957>.
- Romanishin JW, Gilpin K, Claici S and Rus D (2015) 3d m-blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1925–1932.
- Saul G, Lau M, Mitani J and Igarashi T (2011) Sketchchair: an all-in-one chair design system for end users. In: *Proceedings of the fifth international conference on tangible, embedded, and embodied interaction*, TEI '11. pp. 73–80.
- Schulz A, Shamir A, Levin DIW, Sithi-amorn P and Matusik W (2014) Design and fabrication by example. *ACM Trans. Graph.* 33(4): 62:1–62:11. DOI:10.1145/2601097.2601127. URL <http://doi.acm.org/10.1145/2601097.2601127>.
- Skouras M, Thomaszewski B, Kaufmann P, Garg A, Bickel B, Grinspun E and Gross M (2014) Designing inflatable structures. *ACM Trans. Graph.* 33(4): 63:1–63:10. DOI: 10.1145/2601097.2601166. URL <http://doi.acm.org/10.1145/2601097.2601166>.
- Song S, Kim J and Yamane K (2015) Development of a bipedal robot that walks like an animation character. In: *2015 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3596–3602.
- Sun T and Zheng C (2015) Computational design of twisty joints and puzzles. *ACM Trans. Graph.* 34(4): 101:1–101:11. DOI:10.1145/2766961. URL <http://doi.acm.org/10.1145/2766961>.

1145/2766961.

- Sung C and Rus D (2015a) Automated fabrication of foldable robots using thick materials. In: *Proceedings of the International Symposium on Robotics Research (ISRR)*.
- Sung C and Rus D (2015b) Foldable joints for foldable robots. *Journal of Mechanisms and Robotics* 7(2): 021012.
- Tachi T (2011) Rigid-foldable thick origami. In: *Origami 5: Fifth International Meeting of Origami Science Mathematics and Education*. pp. 253–264.
- Thomaszewski B, Coros S, Gauge D, Megaro V, Grinspun E and Gross M (2014) Computational design of linkage-based characters. *ACM Transactions on Graphics* 33(4): 64.
- Umetani N, Igarashi T and Mitra NJ (2012) Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.* 31(4): 86:1–86:11. DOI:10.1145/2185520.2185582. URL <http://doi.acm.org/10.1145/2185520.2185582>.
- Umetani N, Kaufman DM, Igarashi T and Grinspun E (2011) Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph.* 30(4): 90:1–90:12. DOI:10.1145/2010324.1964985. URL <http://doi.acm.org/10.1145/2010324.1964985>.
- Umetani N, Koyama Y, Schmidt R and Igarashi T (2014) Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.* 33(4): 65:1–65:10. DOI: 10.1145/2601097.2601129. URL <http://doi.acm.org/10.1145/2601097.2601129>.
- Wang CH and Sturges RH (1996) Bendcad: a design system for concurrent multiple representations of parts. *Journal of Intelligent Manufacturing* 7(2): 133–144.
- Whiting E, Shin H, Wang R, Ochsendorf J and Durand F (2012) Structural optimization of 3d masonry buildings. *ACM Trans. Graph.* 31(6): 159:1–159:11. DOI:10.1145/2366145.2366178. URL <http://doi.acm.org/10.1145/2366145.2366178>.
- Whitney J, Sreetharan P, Ma K and Wood R (2011) Pop-up book mems. *Journal of Micromechanics and Microengineering* 21(11): 115021.
- Yan X and Gu P (1996) A review of rapid prototyping technologies and systems. *Computer-Aided Design* 28(4): 307–318.
- Yim S, Miyashita S, Rus D and Kim S (2017) Teleoperated micromanipulation system manufactured by cut-and-fold techniques. *IEEE Transactions on Robotics* 33(2): 456–467.
- Yoon C, Xiao R, Park J, Cha J, Nguyen TD and Gracias DH (2014) Functional stimuli responsive hydrogel devices by self-folding. *Smart Materials and Structures* 23(9): 094008.
- Yoshida H, Igarashi T, Obuchi Y, Takami Y, Sato J, Araki M, Miki M, Nagata K, Sakai K and Igarashi S (2015) Architecture-scale human-assisted additive manufacturing. *ACM Trans. Graph.* 34(4): 88:1–88:8. DOI:10.1145/2766951. URL <http://doi.acm.org/10.1145/2766951>.
- Zhakypov Z, Falahi M, Shah M and Paik J (2015) The design and control of the multi-modal locomotion origami robot, tribot. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, pp. 4349–4355.
- Zhu L, Xu W, Snyder J, Liu Y, Wang G and Guo B (2012) Motion-guided mechanical toy modeling. *ACM Trans. Graph.* 31(6): 127:1–127:10. DOI:10.1145/2366145.2366146. URL <http://doi.acm.org/10.1145/2366145.2366146>.