Interactive Robogami

by

Wei Zhao

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

Author .....
Department of Electrical Engineering and Computer Science
May 18, 2015

Certified by .....
Wojciech Matusik
Associate Professor
Thesis Supervisor

Accepted by .....
Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Interactive Robogami

by

Wei Zhao

Submitted to the Department of Electrical Engineering and Computer Science
on May 18, 2015, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this work, we propose a system that allows casual users to design ground robots that can be easily fabricated with a 3D printer. The system leverages a database of examples created by expert roboticists. A composition tool imbedded in this system allows the users to create new designs by composing parts from the robots in this database. The system automatically ensures that the assembled robot is fabricable and that it can locomote forward while still giving creative freedom to users.

Keywords: digital fabrication, data-drive methods, design, robotics

Thesis Supervisor: Wojciech Matusik
Title: Associate Professor

Acknowledgments

First, I would like thank my advisor Professor Wojciech Matusik for letting me the opportunity to work on this exciting project. I am very grateful for his generous support and guidence throughout this work.

I also would like to thank Professor Daniela Rus and Professor Eitan Grinspun for their helpful input.

I would like to thank Adriana Schulz for sharing her insights with me and many immensely helpful discussions, as well as Robin Cheng, Cynthia Sung, Andrew Speilberg for their great teamwork and support.

Finally, I would like to thank my family for their invaluable support.

Contents

List of Figures

Chapter 1

Introduction

Designing a functional robot requires expert knowledge. Not only is the process of designing and programming a new robot challenging, even modifying an existing robot to perform a new task typically requires substantial time and effort. Consider, for example, a robot whose single function is forward locomotion along the ground. In terms of functionality, this robot needs a physical body with moving parts. In addition, it requires information about the motion of each part that will allow the robot to move forward stably (i.e., without toppling). In terms of fabricability, the robot body must be able to be fabricated and assembled.

Even for expert designers, creating a new robot requires many cycles of redesign and testing. The duration of each cycle is often dominated by the amount of time it

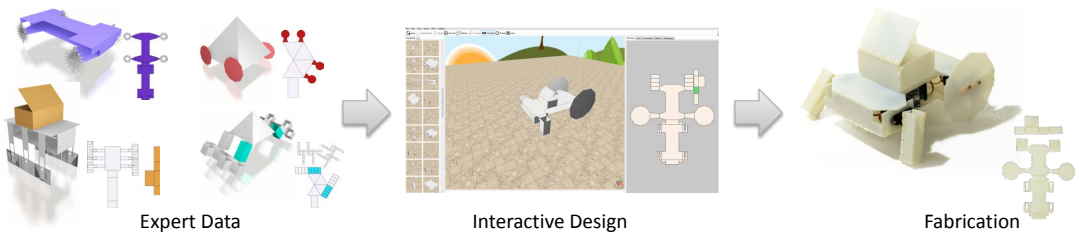

Figure 1-1: Interactive robogami pipeline. From the left to right: expert data is extracted from a collection of robotics designs created by domain specialists. This data is used in a composition tool that allows casual users to easily create new designs following the assembly-based modeling paradigm. The modeling tool is coupled with a physics simulator that helps the user optimize the capabilities of the robot. Finally the designs can be fabricated using a 3D print and fold method.

takes to prototype the latest design. Recent advances in rapid fabrication have already caused significant speedup in the prototyping process. For example, origami-inspired approaches use folding to assemble lightweight robots from a flat sheet [10, 7, 27]. This process is fast and inexpensive. However, restrictions on valid fold patterns makes designing strong structures and flexible joints very challenging. 3D printers can be used to make a wide variety of geometries, but they are slower, and it is difficult to make lightweight hollow structures.

We propose a new fabrication process for mechanical robots, called *3D print and fold*, which combines 3D printing with origami fabrication methods. In our technique, robots are 3D printed as flat faces connected at joints and are then folded into their final shape. The advantage of using a 3D printer is that it allows us to produce stiff, rigid structures with all the flexibility of an addicting manufacturing process. For example, we can easily fabricate parts such as wheels that would be hard to fold from thin 2D sheets. It also allows design of more complex joints, which are essential for actuated mechanisms. At the same time, our technique has several advantages from the origami fabrication process. Since the models are folded out of 2D patterns, the resulting structures are lightweight. In addition, fabrication is cheaper since we need less support material, and faster since the number of layers printed is dramatically reduced. Typically, printing time is reduced by 75.9% and the material usage is reduced by 58.6%.

For novice designers, creating an initial design or updating a previous design can be a challenge. The designer must specify a geometry and motion for the robot that are valid in that they will actually allow the robot to complete its task. In addition to designing the robot itself, using folding as our assembly process introduces additional challenges for the designer. In particular, since a fold pattern cannot self-intersect, unfolding a 3D structure into a 2D fold pattern can be difficult, especially when joints that allow more complex motion must also be integrated.

To help casual users design ground robots using our 3D print and fold technique, we present our Interactive Robogami system. Figure 1-1 illustrates the design workflow. Our system leverages a database of existing robot examples created by expert

roboticists. A composition tool allows users to create new designs by composing parts from the robots in this database. Our tool provides interactive assistance by suggesting the connection and movement between parts based on previous designs and automatically handles 2D and 3D constraints to ensure that the assembled robot is fabricable using our 3D print and fold technique. Designs can be simulated in order to verify that they move as expected, and design parameters may be changed until the user is satisfied. Once the user is done, the system generates a model that can be 3D printed and assembled.

We provide end-to-end demonstrations that start with geometric models composed from parts in our database and undergo several rounds of user transformations and stability checks. We have used our system to create physical functional prototypes for a biped, three multi-legged crawlers, and a wheeled robot.

1.1 Contributions

This thesis makes the following contributions:

1. an end-to-end system that allows casual users to design and fabricate custom 3D print and fold robots,

2. a new fabrication process consisting of 3D printing and then folding a 2D fold pattern, which produces strong and lightweight robots, and

3. a representation for parametric 3D geometries and their corresponding 2D fold pattern and algorithms for their manipulation, composition, and interactive simulation.

1.2 Thesis Overview

We first go over the previous work in chapter 2. Then we describe the expert data in our database in chapter 3, highlighting the properties of our data. In chapter 4, we first present the design workflow in our system and then we discuss the composition

algorithms in details. In chapter 5, we go through the fabrication method we used to create the robots. Chapter 6 shows the experiments we have run to test Robogami and the results we have produced. Finally, we discuss the potential extentions and future work in chapter 7.

Chapter 2

Related Work

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

2.1 Origami-Based Fabrication

Origami inspired approaches have been widely used to allow low-cost and fast fabrication of 3D objects from 2D sheets [18, 9, 13, 5]. Recent work in rapid fabrication of robots have explored such methods to allow functional robots to be created within a day [10, 23, 22, 7, 16]. For example, flat sheets can be folded into robot bodies [1, 17, 19, 27] and electronic components [20] with a wide range of functionalities while still being lightweight. However, the robots produced in this way often suffer lower rigidity compared to their machined counterparts. In addition, design of durable functional joints from folded structures is difficult. In our work we propose combining origami-inspired fabrication with additive manufacturing. This allows for more flexibility in designing geometries (e.g., joints), while preserving the lightweight and low-cost properties of folding.

2.2 3D Printing of Functional Mechanisms

3D printing is a technique that has been used to produce complex mechanisms without requiring large amounts of post-printing assembly [28, 3, 4, 2]. However, compared to folded structures, 3D printed object are costly, heavy, and slower to fabricate. Many times printing these mechanisms also requires printing and then removing support material, which adds time to the fabrication process. In this context recent works have suggested alternative designs for more efficient 3D printing. Wang et al. [33] suggest a method for making 3D printing more cost effective by introducing skin-frame structures and Lu et al. [14] propose a method for reducing the material cost and weight by hollowing out structures. Our proposed method of printing flat sheets and folding allows the production of hollow, lightweight structures while using minimal support material. Flat sheets are much faster to print since they require fewer layers. We have also designed new printable, snappable joints that give users greater design flexibility than in traditional print and fold fabrication. Our system therefore combines the lightness and speed of folded designs with the versatility and rigidity of 3D printing into a single fabrication method for quickly printing and assembling robot bodies.

2.3 Design of Functional Mechanisms

Design generation work in the robotics community often considers mechanical structure and task-specific movement in isolation; for example, mechanisms are designed independently of actuation [35, 17, 27]. Although fabrication plans for fully functional robots were outputted in [15], the designs were driven by mechanical considerations, and the electrical and software components generated in a post-processing stage. In contrast, our system simultaneously considers the mechanical and necessary motion of the robot.

The graphics community has also addressed the problem of designing and fabricating machines. Recent work has proposed efficient techniques that generate driving

mechanisms for toy designs that match a desired input motion [35, 6, 28]. These works perform a global optimization over a given motion specification. In our work, we propose an interactive tool and combine composition with physics-driven suggestions. With this we give more low level control to the user and allow a more diverse set of models to be generated. Several previous works have also incorporated simulation tools in interacting modeling system to allow users to explore the space of valid shapes while optimizing for certain goals. In order to achieve real-time performance and interactivity, such simulation techniques and exploration tools tend to be application driven. Examples of such interactive systems are modeling tools for furniture [29], clothes [30], planes [31] and masonry structures [34, 32]. In our work we provide interactive simulation for robots with ground locomotion. We exploit the parametric representation as well as the 2D/3D representation of our database to develop efficient simulation algorithms. We use these to give interactive feedback to the user allowing them to explore the space of valid designs while optimizing for speed and stability.

2.4 Fabrication-oriented Design

Design tools for fabrication has garnered a lot of interest in the computer graphics community. Proposed systems include tools for plush toys [21], furniture [24, 29, 12], clothes [30], inflatable structures [26], wire meshes [8], model airplanes [31], and prototypes of mechanical objects [11]. The work that is the closest to ours is the one by Schulz et al. [25], which also proposes a system to generate new fabricable designs by composing parts in a data collection. This system, however, only handles simple functionality and does not incorporate the kinematic representation necessary to model complex articulated machines. It also has not incorporated interactive simulation which is essential to allow design of functional objects. Finally, this work only takes into account assemblies of parts that come from an items catalog. We allow for more versatile 3D printed geometries and handle composition of parametric shapes that have a combined 2D and 3D representation. This guarantees that the designs created by our tool can be fabricated with our 3D print and fold technique.

Chapter 3

Expert Data

Robogami operates on a database of expert-designed robots, each of which can be folded from a 2D sheet into a 3D robot. They additionally encode information about their functionality and composability, so that when these robots are composed and decomposed, the resulting robot can still behave as we expect. All this information is parametric and hierarchical, allowing easy manipulation and reuse of robot parts, so that a few expert designs in the database can yield a large variety of composed designs.

3.1 Parametrization

The parametric representation of a 3D mesh along with its 2D unfolding is called a "template". The expert defines the meaningful parameters for this template, such as 'width' and 'height', and the 3D mesh and 2D unfolding are then encoded as expressions in terms of these parameters. Parameters must also satisfy certain constraints. In other words, a template can be written as a tuple

$$T = (\mathbf{q}, \mathcal{A}, F) \tag{3.1}$$

where $\mathbf{q}$ is a vector of the parameters and their values, $\mathcal{A}$ is the set of allowed values of $\mathbf{q}$, and F is a function such that $F(\mathbf{q})$ produces the 3D and 2D meshes.

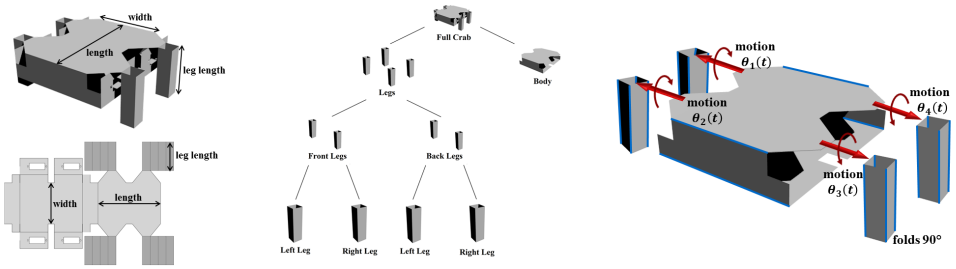

Figure 3-1: From left to right: an example expert design of a four legged robot with its 2D unfolding, the hierarchical structure of the same robot, and a visualization of the connections and its joint motion information..

Because most designs have meshes that can be represented linearly in terms of the parameters, we implement $\mathcal{A}$ as a set of linear equations and inequalities, and F as a linear function. This allows us to only have to tackle linear systems, and linear expressions are also easy to store in a file.

By altering the parameter values, we can change the shape and dimensions of the template.

3.2 Hierarchy

Each robot design is broken down into a hierarchy of templates, semantically representing how the design can be decomposed. Across different templates in the hierarchy, there can be additional constraints as well as connectivity information.

Two templates can be connected by joining a pair of edges. This is accompanied by constraints that make these edges stay connected even when the design is manipulated by the user. When fabricated, each connection turns into a fixed or flexible joint, depending on the connection type and the motion information encoded by the connection.

Chapter 4

Interactive Design

4.1 Design Workflow

Robogami provides a user interface where the interactive designing takes place. A screenshot of the UI is shown in Figure 4-1. The window is divided into three panels, resembling the order of the data flow: the left panel shows the available designs from the database; the center canvas displays the 3D model and allows user manipulation; the right panel shows the 2D unfolding as well as other properties of the resulting design, including the final printable mesh, joint information, cost, and stability.

A design session begins by dragging a design into the 3D workspace. This adds the root template of the design hierarchy as the *main* template. Dragging additional templates onto the main template will compose them onto the main template through new connections. At any point of time, the user can also tweak the design's dimensions by scaling parts of the design, and the template constraints will ensure that the both the 3D model and the 2D unfolding stay consistent. Robogami provides guidance for these operations: when adding a new template, Robogami suggests the best pairs of edges to connect on; when scaling the parts, Robogami suggests the direction of scaling that results in better stability.

As the user composes, decomposes, and manipulates the robot, joint information from each part is maintained. This information can be visualized by clicking *Animate*, which gives a preview of the kinematic behavior of the robot as the joints move.

During the design process, the system continuously evaluates important properties of the design, such as stability, material costs, and printing speed, and displays them in one of the tabs in the panel on the right. If any of these metrics are suboptimal, the *Optimize* menu provides several objectives (such as geometry stability, speed, or cost) to optimize against. When the user issues an optimization command, the system automatically finds scaling parameters to improve the corresponding metric.

4.2 Optimization

Most operations on a design is achieved by optimizing some objective against the template constraints. The template constraints are all the constraints that come with the original design, as well as the constraints that are added through composition.

For example, if a robot leg is connected through an edge to its body, a constraint is added that will always make the leg snap to the body through that edge. Another constraint is added that does the same in the 2D unfolding. Each optimization operation is subject to all the constraints including this one, so no matter what we do, the leg and body are always snapped correctly.

In other words, each optimization operation solves the following least squares problem:

$$\begin{aligned} q = &\operatorname{argmin} \|C(q)\| + \alpha \|D(q) - D(q_{current})\| \\ &s.t \quad q \in \mathcal{A} \end{aligned} \tag{4.1}$$

where $C(q)$ is the objective function, and $D(q)$ is the sum of squares of each face's bounding box dimensions. This part is to incentivize the system to preserve the unaffected parts of the design as much as possible. For example, if a user scales the body of a robot, the dimensions of the legs should remain unchanged.

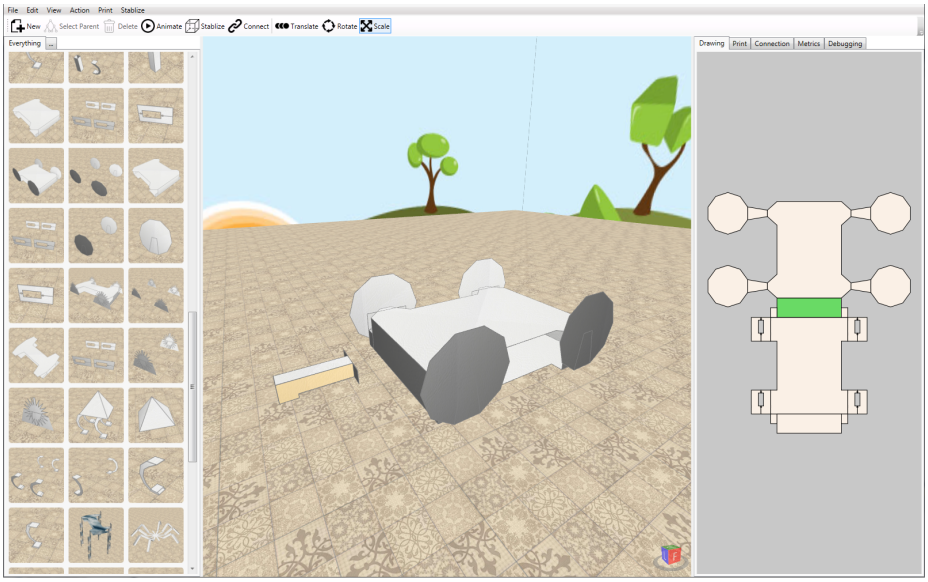

Figure 4-1: The user interface. Icons that link to components of the database are displayed on the left, the modeling canvas is in the center, and the 2D design is displayed on the right.

4.3 Manipulation

The 3D geometry of a template is represented as a triangular mesh whose vertex positions are linear combinations of the symbolic parameters, all in the global coordinate system. This necessitates the need for three additional parameters to encode the global position of the entire mesh.

To manipulate the 3D geometry, the user may translate, rotate, or scale it. Translation affects only the global position parameters, but rotation is nonlinear, so it involves rewriting the vertex positions with new linear coefficients. Scaling, on the other hand, cannot be meaningfully done on the entire template, because in a design with many parameters, there can be many different ways to scale. For example, a robot with four legs can scale horizontally either by making the body wider or making the legs thicker.

Therefore, we only allow scaling on a selected face. The face is assumed to be simple enough so that each face grants enough granularity that the user can achieve any desired parametric configuration by scaling faces.

When the user selects a face, the system heuristically chooses the axes direction

on the face, and allows scaling in either direction (see Figure 4-2). Once the axes direction is picked, a symbolic 2D bounding box of the face can be computed, as two linear expressions. To do this, we first project the 3D symbolic vertex positions onto the numeric 2D plane the face lies on. Then, for each axis of the bounding box we project the symbolic 2D vertex positions onto the numeric 1D axis. Finally we take the minimum and maximum of these symbolic 1D vertex positions and subtract them to get the symbolic bounding box dimension. Note that *max* and *min* are not linear, so we substitute the current parametric values just for comparison. This is based on our assumption that the shape of each face should stay roughly the same across different parametric configurations.

Then the user may scale along a dimension, which then triggers an optimization where the objective function is $C(q) = d(q) - k \cdot d(q_{current})$, where $d(q)$ is the linear expression for the scaled dimension of the bounding box, and k is the scaling factor.

Because all constraints are enforced in every optimization, scaling a single face propagates changes to other faces and templates of the design, including the 2D unfolding. Additionally, the system detects polygon collisions in the 2D unfolding (which would be unprintable), and if a scaling operation would result in a 2D collision, the parameters are reverted to avoid the collision.

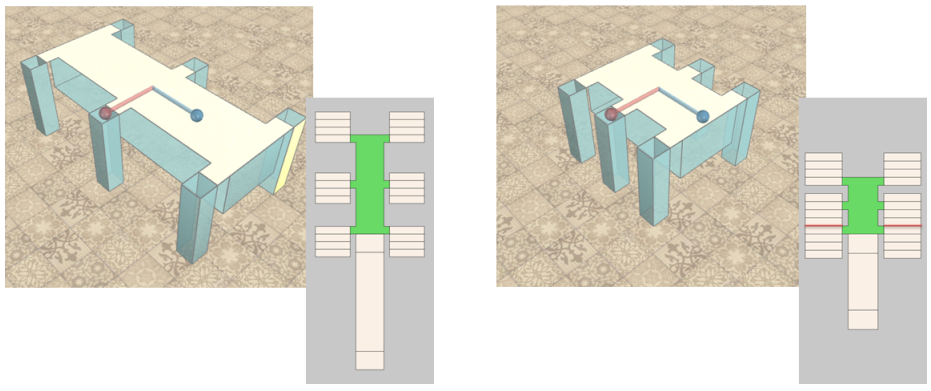

Figure 4-2: Manipulation controls that are drawn for each selected face to allow dimension manipulation.

4.4 Composition

Robogami allows the user to compose new designs in two basic ways. First, the user can remove parts from existing designs. Second, the user can also add parts to existing designs. This part removal and addition can happen at any level of the hierarchy, giving users flexbility to create new designs that are very distinct from the original designs the user starts with.

4.4.1 Removing a Part

The user can select a face and traverse up the template hierarchy, in order to select any desired template in the hierarchy. Once selected, it can be removed from the rest of the design. Removal of template D involves the following steps:

1. Any constraint that involves any parameter defined under D is removed.
2. Any connection between an edge in D and an edge not in D is replaced by a *half-open connection* with the former edge replaced by a placeholder but the joint information preserved. Later, when connected with a new edge, it becomes a normal connection again with the original joint information.
3. D is cut off from the template tree.

This process tries to preserve as much as information as possible while leaving the resulting incomplete design valid.

4.4.2 Adding a Part

The user may also add a part by dragging in another design (such as a leg) and connecting it to the current design. Robogami assists the user in choosing where to connect, by trying different combinations of edge pairs and considering their feasibility. This process is called *snapping*.

Figure 4-2 shows an example of snapping. Suppose the user wants to add template D^A to template D^W. For each open connection edge $e_i^A = (v_i^A, w_i^A)$ in D^A, a

corresponding edge $e_i^W = (v_i^W, w_i^W)$ is found in D^W that is optimal (one that gives the least optimization cost). Then, to snap the two templates together, the system performs an optimization with the cost function

$$C(q) = \sum_i \|v_i^A - v_i^W\|^2 + \|w_i^A - w_i^W\|^2 \tag{4.2}$$

where $v_i^A, w_i^A, v_i^W, w_i^W$ are vertices evaluated for parameters q.

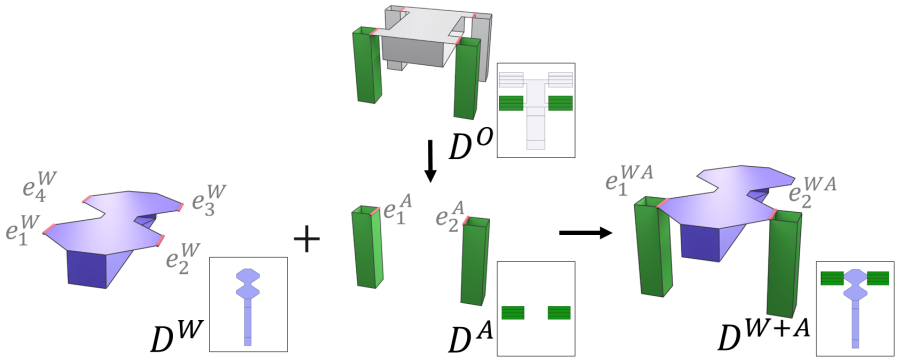

Figure 4-3: Example of snapping. When the user wants to add D^A (removed from D_O) to D^W, the open edges e_1^A and e_2^A matches the open edges e_1^W and e_2^W. D^A is then added to D^W by connecting these two edge pairs.

For each e_i^A, we consider e_i^W from the k closest edges in D^W to e_i^A (open edges in D^W are given preference). In our implementation, $k = 3$. Then for each combination of choices of e_i^W we try this optimization, and in the end we use the choices that give the minimal cost. For each e_i^A we also consider the option of not connecting it to anything, with a cost penalty to incentivize the system to snap more edges if feasible.

If there is only one edge to snap, the user is given additional flexibility. Before doing the optimization we first rotate D^A such that e_1^A matches the direction of e_1^W, so that even if D^A is in a bad orientation, it can still be attached. Furthermore, after snapping, the user can rotate D^A around the snapped edge to adjust the angle of the connection, as well as scaling D^A's faces while maintaining the snapped configuration, which may involve splitting e_1^A or e_1^W into two or three edges if e_1^A's length is changed due to scaling.

If the user is satisfied with the snapping configuration, they may commit the

snapping, and the following will happen:

1. D^A is added to D^W as a child template.

2. For each edge pair (e_i^A, e_i^W), six linear constraints are added (two three-dimensional vertex pairs) that ensure that the edge pair will always coincide.

3. For each edge pair (e_i^A, e_i^W), create a connection. If either or both of them already belong to a half-open connection, the joint information from the original half-open connection is carried into the new connection.

4. The 2D unfolding of A is added to the 2D unfolding of D^W by similarly joining the corresponding edge pairs. The unfolding of D^A may have to be rotated so that each pair of edges have the correct orientation. If the resulting unfolding contains an overlap, edge pairs in 2D are separated and the corresponding 2D components moved aside until no overlaps still exist. If an edge pair in 2D is separated, the joint type for that connection is updated if necessary so that the printed parts can still be re-joined together. See Figure 4-3 for an example.

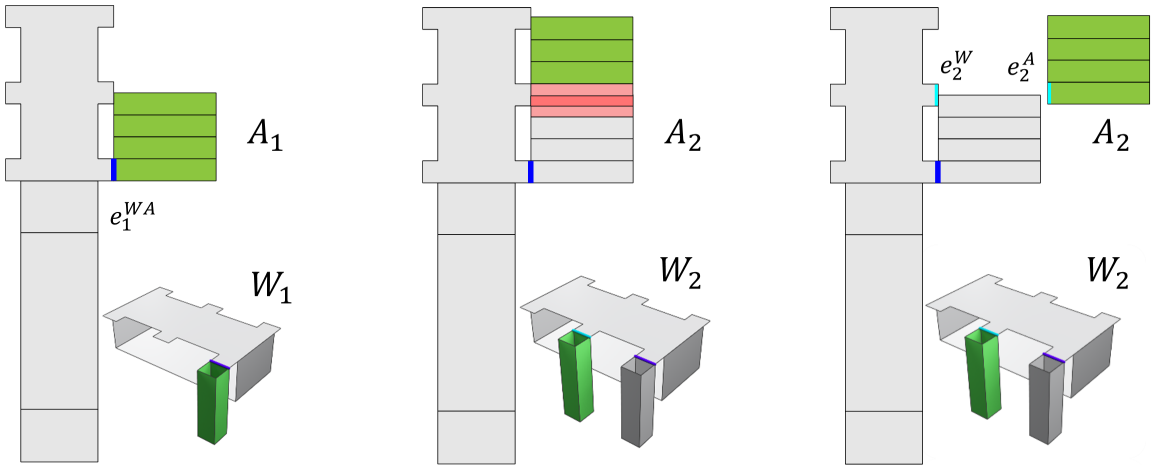

Figure 4-4: Connecting parts in 2D with collision prevention. On the left, adding one leg D^{A_1} does not cause any collisions, so the connecting edge e_1^{WA} may stay together in both 2D and 3D. However, when when adding a second leg D^{A_2}, a collision (highlighted in red) would be introduced, so we move D^{A_2} outside the bounding box of D^{W_2} to avoid collision.

4.5 Evaluation

Throughout the designing process, the design is constantly evaluated on several metrics: stability, movement speed, cost of materials.

The cost of materials is simple to compute because it's the total surface area of all the faces.

To analyze stability and movement speed, we first need to animate the robot based on the motion information encoded at the joints. At each timestep, the transformations are computed recursively for each template, and the geometry recomputed for the whole design. The center of gravity is computed from the mass and center of each face, and from this, we compute the points on the robot that come into contact with ground. By assuming that points that remain in contact with the ground do not move, we can simulate the robot's horizontal translation just like the physical world, and compute the movement speed. If the convex hull of the points that contact ground does not contain the projection of the center of gravity onto the ground, the model is unstable for that timestamp, and we tell the user that it is unstable by highlighting the model in red.

These metrics are displayed in the appropriate tabs on the right hand side panel in the UI. The user may optimize a certain metric using the Optimize menu. Suppose the user wants to optimize the speed of the robot, then we run an optimization with the cost function being $C(q) = S(q)$, where $S(q)$ is the speed of the robot when parameterized at q. Because this cost function is not linear and not differentiable, we use finite differences to approximate the gradient, and then gradient descent for the optimization.

The user can also do stability optimization manually. When a face is selected, arrows will indicate which direction of scaling results in better stability. For example, in Figure 4-4, the blue arrow pointing down indicates that shrinking the leg vertically provides more stability.

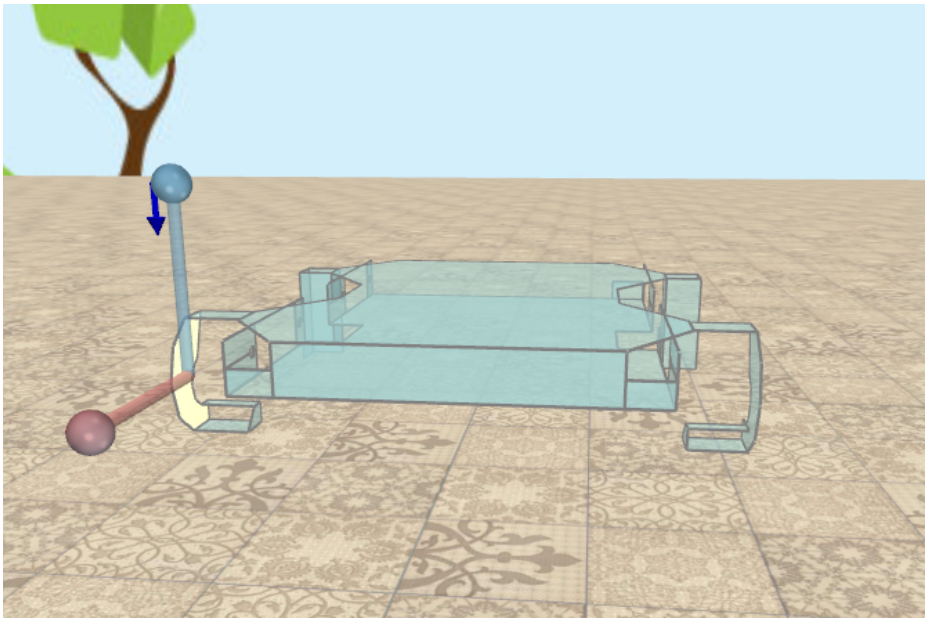

Figure 4-5: An unstable design that is being manipulated. The arrow indicates that making the legs shorter improves stability.

Chapter 5

Fabrication

Once the user is satisfied with the robot design, Robogami generates a 3D-printable mesh from the final composed 2D unfolding, which are represented as a set of 2D polygons.

First, each polygon is extruded by 1mm. We choose this thickness because it's durable enough while still allowing flexible folding angles up to almost 180°.

Second, depending on the connection type, we replace the folding edges with 3D meshes. Figure 5-1 shows the four types of connections translated into 3D meshes. The hinge/revolute joint is used for parts that need to flexibly rotate along with the edge. The teeth joint is used for snapping together edges that lie on two ends of the 2D folding. The prismatic joint is used to move an edge along another edge. The ball and socket joint is used for arbitrary rotation such as legs. These meshes are designed so that the two parts snap and function properly whether printed together or separately, so even if part of the 2D unfolding is disconnected to avoid collisions during composition, the printed result can still snap together in one piece. Of course, printing as many edges together as possible will reduce the assembly effort needed.

If an edge corresponds to a folding angle of 0, it is not replaced with 3D meshes; instead the two polygons that connect to this edge are merged. The user may also choose to exclude any edges from being printed as connections, which is useful if we want to replace a joint with electronics.

Because a connection mesh take space itself, we trim the face mesh along the edge

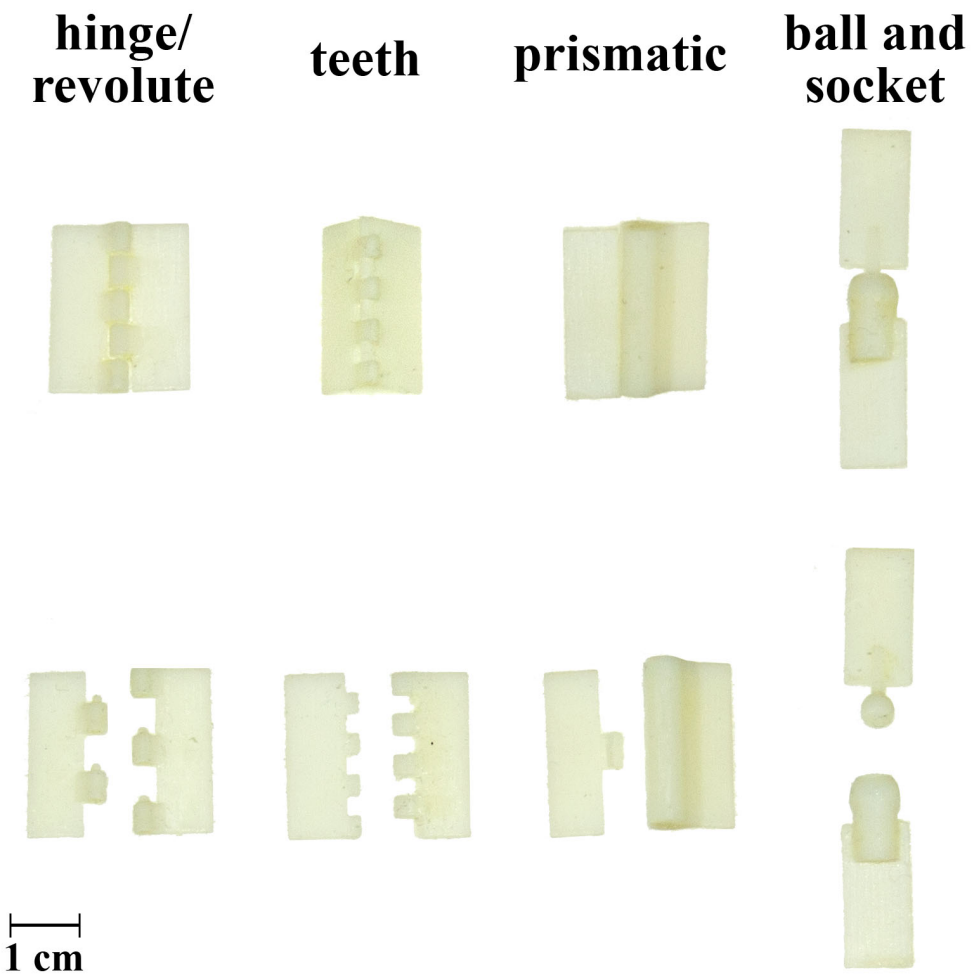

Figure 5-1: Joint designs. Each can either be printed together (top row), or separately and then later snapped (bottom row).

to make just enough room for the connection mesh. See Figure 5-2 for an example. The amount of trimming depends on the size of the connection mesh. To avoid collision between the connection meshes of two adjacent edges of the same face, the length of the edge is calculated after the face is trimmed on all edges.

Finally, we represent the whole output as a tree of CSG operations, which are then flattened into a single 3D mesh printable with a 3D printer.

After the mesh is printed, the robot must be folded, and the electronics and motors mounted.

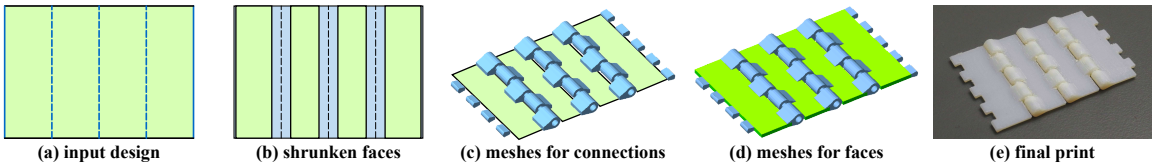

Figure 5-2: Insertion of connection meshes, for a simple rectangular prism part. (a) The 2D unfolding containing four faces. (b) The faces are trimmed. (c) Connection meshes are inserted and (d) together with the extruded faces. (e) The final mesh is printed.

Chapter 6

Experiments and Results

In this chapter we discuss the evaluations of our pipeline and experiments we have run to test Robogami.

6.1 Expert Data

Our database builds upon the work in [17], in which robots foldable from 2D sheets are designed using Python scripts. We augmented its Python API to output to a common format readable by Robogami, and in addition, include additional information such as template hierarchy and motion information (legged motion and wheel-based motion). This allows us to load in the expert designs and simulate the locomotion. With this augmentation, the design time generally takes 1 - 2 hours, depending on the complexity of the design. The experts need to specify the 2D face geometries with folding angles, hierarchical structure, constraints on parameters, connections, and motions. The Python API then computes the 3D parametric mesh and other necessary information. Typically, a model in our database consists of a body, legs or wheels, and sometimes functional or decorative components like a robot arm or card holder.

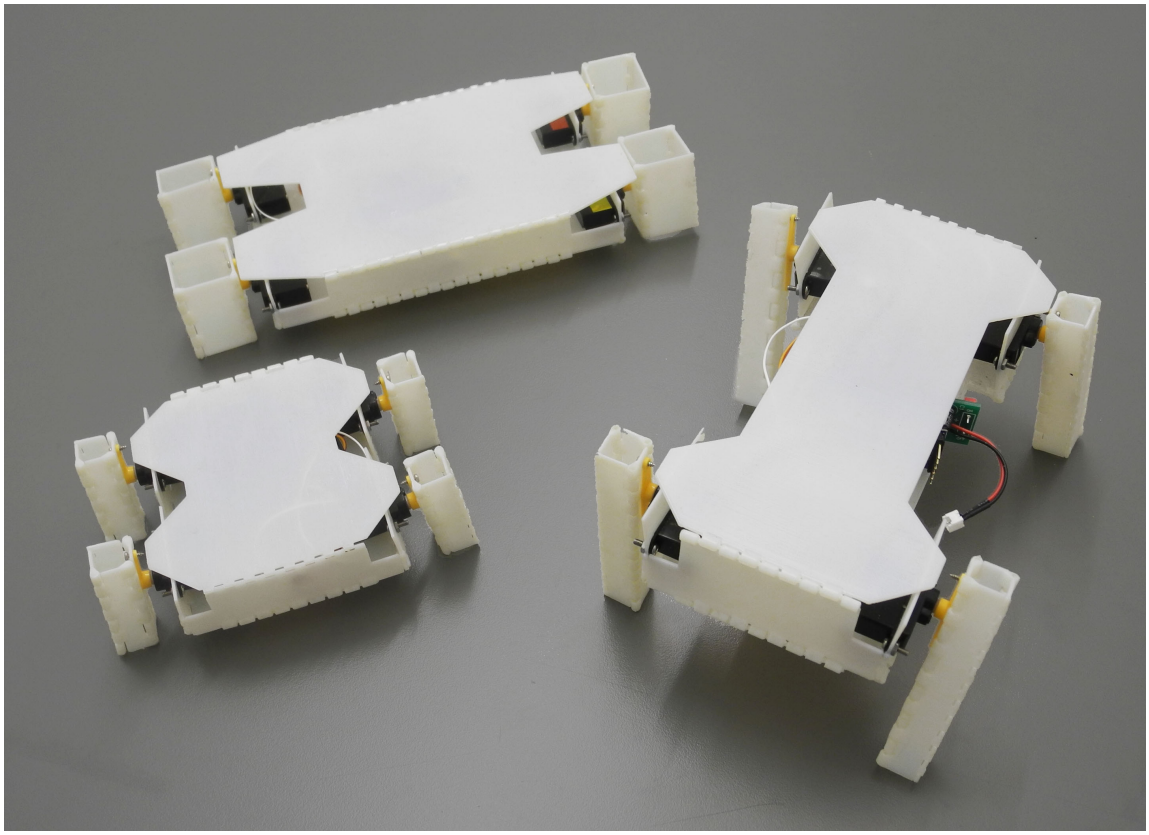

Figure 6-1: Three functional four-legged robots generated and fabricated by varying the parameters of an existing design in the database.

6.2 Parameter Manipulation

We demonstrate that changing parameters of a design in Robogami still preserves the functionality ad fabricability of the robot. Figure 6-1 shows the resulting robots after we have significantly varied the paramters of its original design. It took about 10 minutes to design and simulate, and 25 minutes to assemble each robot. After assembly, all three robots are able to walk forward as desired.

6.3 Composition Tool

We have composed and built a variety of new robots out of the existing parts in our database to test the composition capabilities of Robogami. Figure 6-2 shows a collection of robots that we have designed using our composition tool. To show the

complexity of these newly designed robots, we also color each part that originates from a different design with a different color. The hierarchical repsentations of our models give users flexibilities to design at different levels by taking smaller or larger substructures of existing models. Because we can mix and match each part, the designs space grows exponentially with the size of the database.

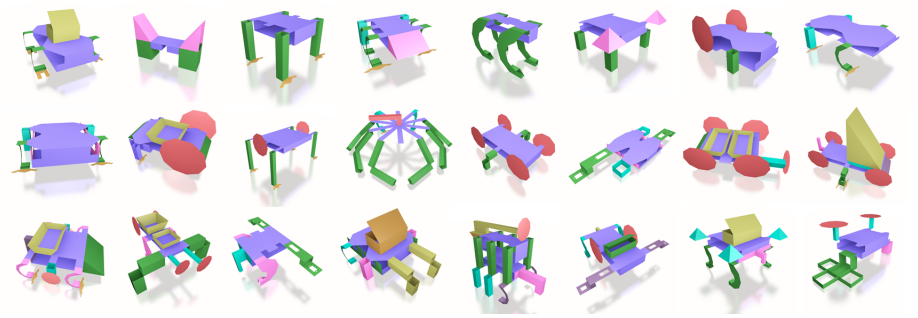

Figure 6-2: Various designs composed with Robogami. Different colors indicate parts that come from different designs.

6.4 Simulation and Evaluation

Evaluation of metrics generally takes about 10 milliseconds per analysis run. This includes stability, speed, and cost of materials. Because of the short running time, we update these metrics whenever the user modifies the design by composition or manipulation. When the model is not stable, we visually suggest directions to stablize it, as shown in Figure 4-5.

The optimization commands each take about 5 seconds. Figure 6-3 shows the results when optimized for two different objectives: initial geometry and speed. The results respects our intuitions about static stability. As shown in the figure, in order to make the model stable but preserve as much of its initial geometry as possible, the legs should be shorter and its body should be wider. This makes sense as shorter legs move the center of mass closer to the ground, and a wider body ensures that the center of mass will fall inside the convex hull of the contact points on the ground when one of the leg is lifted above ground. To make the robot stable while making

it faster, the legs should be longer, but to ensure its stability, the body needs to be even wider.

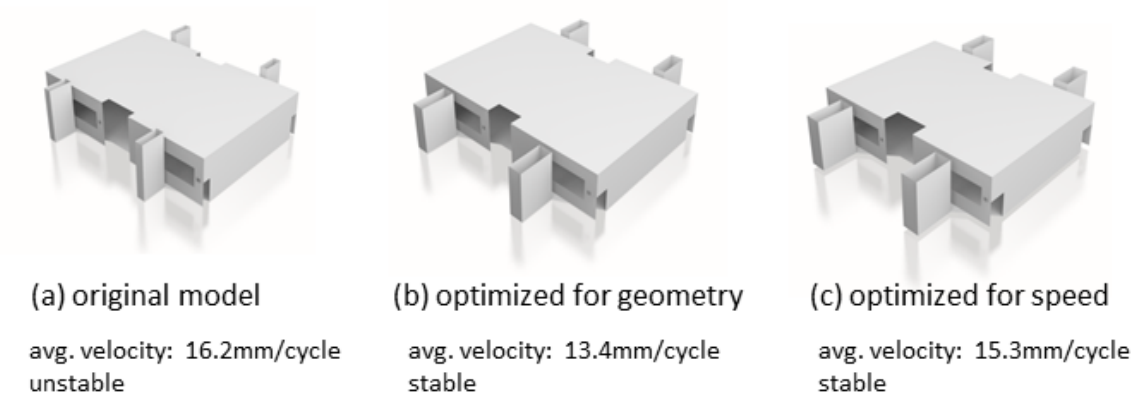

Figure 6-3: Example of optimization results for geometry and for speed.

6.5 Fabrication

We tested the system end-to-end for five different models:

1. Biped. This is a robot that uses prismatic joints to move its body and legs in a sequence that moves the robot forward. The body and legs are actutated with linear servomotors. This robot can walk forward at a speed of 16.7 mm/s.

2. Car. A robot with wheels. The wheels are printed separatedly and attached through servos. The car moves forward at 298 mm/s.

3. Crab. A simple four-legged walking robot that rotates one of its legs at a time. It can walk forward at 28.5 mm/s.

4. Truck. A mix of car and crab, with two front legs and two back wheels. It rotates its wheels at the same time as moving the front legs alternatively. It can walk forward at 31.2 mm/s.

5. Ant. Robot with six legs, three of which remain stationary to maintain stability, and the others rotate together. It can move forward at 21.8 mm/s.

Table 4.1 shows the times taken to design, print, and assemble each of these robots.

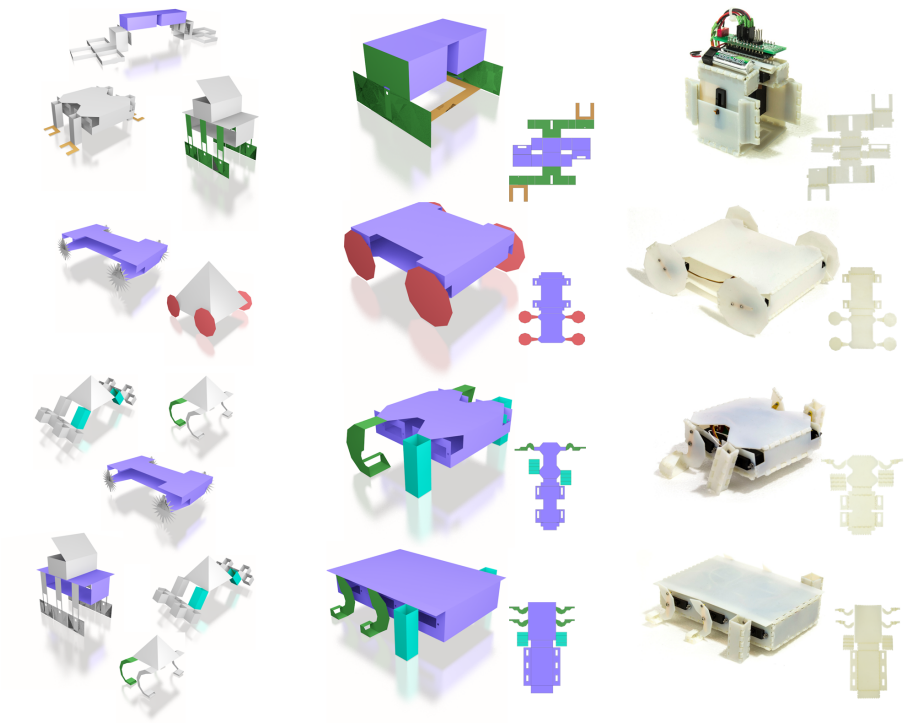

Figure 6-4: From left to right: input designs (with used parts highlighted), models created using the system, and fabricated results.

Model	Design	Printing	Assembly
Biped	1.5 min.	99 min.	32 min.
Car	1.5 min.	155 min.	14 min.
Crab	1 min.	63 min.	20 min.
Truck	1.5 min.	135 min.	18 min.
Ant	2 min.	123 min.	29 min.

Table 6.1: Length of each phase for robot creation.

6.6 Limitations

Robogami has several limitations.

1. The possible designs are limited by the designs in the database.

2. The system currently only supports models that can be represented as linear combinations of the parameters. This means that we cannot use models that have non-linear features, such as a parameter that controls rotation.

3. Connection information is currently copied from the original model to the newly

composed model. This may not always be appropriate.

4. The system currently does not keep track of electronics information, so it's up to the assembler to be knowledgeable enough about how to mount the electronics.

Chapter 7

Conclusions and Future Work

Robogami provides a UI for casual users to design a new functional robot by composing parts extracted from a expert-created database. The system assists the user in manipulating the robot, and provides analysis of robot's stability, speed, and cost in real time. After a robot is satisfactory, the system generates a 3D-printable mesh that folds into the desired physical robot. We used the system to compose several robots and verified that they are fabricable and functional.

My work has mostly been focused on the UI aspect of the project. To ease implementation of user interactive features, we use C# WPF to implement the UI, and use a C++/CLI bridge to communicate between the UI and the backend logic. This has been very useful especially when implementing manipulation gizmos (such as rotation and scaling) in the 3D view, and when adding new UI features such as editing joint information in the right-hand side panel.

Work is in progress to extend the system with the ability to include non-foldable but printable 3D components such as a 3D wheel. Currently we can load a template from a parameterized OpenSCAD script. In the near future we plan to integrate this with the rest of the system, making it attachable with linearly parameterized models, and print these 3D components together with the 2D unfoldings.

Bibliography

[1] Byoungkwon An, Shuhei Miyashita, Michael T Tolley, Daniel M Aukes, Laura Meeker, Erik D Demaine, Martin L Demaine, Robert J Wood, and Daniela Rus. An end-to-end approach to making self-folded 3d surface shapes by uniform heating. In *IEEE International Conference on Robotics and Automation*, pages 1466–1473, 2014.

[2] Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. Fabricating articulated characters from skinned meshes. *ACM Trans. Graph.*, 31(4):47:1–47:9, July 2012.

[3] Jacques Calì, Dan A. Calian, Cristina Amati, Rebecca Kleinberger, Anthony Steed, Jan Kautz, and Tim Weyrich. 3d-printing of non-assembly, articulated models. *ACM Trans. Graph.*, 31(6):130:1–130:8, November 2012.

[4] Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. Designing and fabricating mechanical automata from mocap sequences. *ACM Trans. Graph.*, 32(6):186:1–186:11, November 2013.

[5] Desai Chen, Pitchaya Sitthi-amorn, Justin T Lan, and Wojciech Matusik. Computing and fabricating multiplanar models. In *Computer Graphics Forum*, volume 32, pages 305–315. Wiley Online Library, 2013.

[6] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. *ACM Trans. Graph.*, 32(4):83:1–83:12, July 2013.

[7] S Felton, M Tolley, E Demaine, D Rus, and R Wood. A method for building self-folding machines. *Science*, 345(6197):644–646, 2014.

[8] Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. Wire mesh design. *ACM Trans. Graph.*, 33(4):66:1–66:12, July 2014.

[9] Jiangtao Gong, Jue Wang, and Yingqing Xu. Paperlego: Component-based papercraft designing tool for children. In *SIGGRAPH Asia 2014 Designing Tools For Crafting Interactive Artifacts*, SA '14, pages 3:1–3:4, New York, NY, USA, 2014. ACM.

[10] Aaron M Hoover, Erik Steltz, and Ronald S Fearing. RoACH: An autonomous 2.4g crawling hexapod robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 26–33. IEEE, 2008.

[11] Bongjin Koo, Wilmot Li, JiaXian Yao, Maneesh Agrawala, and Niloy J. Mitra. Creating works-like prototypes of mechanical objects. *ACM Trans. Graph.*, 33(6):217:1–217:9, November 2014.

[12] Manfred Lau, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. Converting 3d furniture models to fabricatable parts and connectors. *ACM Transactions on Graphics*, 30(4):85, 2011.

[13] Xian-Ying Li, Chao-Hui Shen, Shi-Sheng Huang, Tao Ju, and Shi-Min Hu. Popup: Automatic paper architectures from 3d models. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 111:1–111:9, New York, NY, USA, 2010. ACM.

[14] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph.*, 33(4):97:1–97:10, July 2014.

[15] Ankur Mehta, Nicola Bezzo, Byoungkwon An, Peter Gebhard, Vijay Kumar, Insup Lee, and Daniela Rus. A design environment for the rapid specification and fabrication of printable robots. In *International Symposium on Experimental Robotics*, 2014.

[16] Ankur Mehta, Joseph DelPreto, and Daniela Rus. Integrated codesign of printable robots. *Journal of Mechanisms and Robotics*, 7:021015, 2015.

[17] Ankur Mehta and Daniela Rus. An end-to-end system for designing mechanical structures for print-and-fold robots. In *IEEE International Conference on Robotics and Automation*. IEEE, 2014.

[18] Jun Mitani and Hiromasa Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph.*, 23(3):259–263, August 2004.

[19] Shuhei Miyashita, Isabella DiDio, Ishwarya Ananthabhotla, Byoungkwon An, Cynthia Sung, Slava Arabagi, and Daniela Rus. Folding angle control by designing a curved crease for self-assembling origami propellers. *Journal of Mechanisms and Robotics*, 7(2):021013, 2015.

[20] Shuhei Miyashita, Laura Meeker, Maurice Gouldi, Yoshihiro Kawahara, and Daniela Rus. Self-folding printable elastic electric devices: Resistor, capacitor, and inductor. In *IEEE International Conference on Robotics and Automation*, pages 1446–1453. IEEE, 2014.

[21] Yuki Mori and Takeo Igarashi. Plushie: An interactive design system for plush toys. *ACM Transactions on Graphics*, 26(3):45:1–45:8, 2007.

[22] Yash Mulgaonkar, Caitlin Powers, and Vijay Kumar. Kinematic analysis of quadrotors with manufacturing errors. In *Advances in Mechanisms, Robotics and Design Education and Research*, pages 205–214. Springer, 2013.

[23] CD Onal, RJ Wood, and D Rus. An origami-inspired approach to worm robots. *IEEE/ASME Transactions on Mechatronics*, 18(2):430–438, 2013.

[24] Greg Saul, Manfred Lau, Jun Mitani, and Takeo Igarashi. 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, pages 73–80, 2011.

[25] Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-amorn, and Wojciech Matusik. Design and fabrication by example. *ACM Trans. Graph.*, 33(4):62:1–62:11, July 2014.

[26] Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. Designing inflatable structures. *ACM Trans. Graph.*, 33(4):63:1–63:10, July 2014.

[27] Cynthia Sung and Daniela Rus. Foldable joints for foldable robots. *Journal of Mechanisms and Robotics*, 7(2):021012, 2015.

[28] Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. Computational design of linkage-based characters. *ACM Transactions on Graphics*, 33(4):64, 2014.

[29] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4):86:1–86:11, July 2012.

[30] Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. Sensitive couture for interactive garment modeling and editing. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 90:1–90:12, New York, NY, USA, 2011. ACM.

[31] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.*, 33(4):65:1–65:10, July 2014.

[32] Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. Design of self-supporting surfaces. *ACM Trans. Graph.*, 31(4):87:1–87:11, July 2012.

[33] Weiming Wang, Tuanfeng Y. Wang, Zhouwang Yang, Ligang Liu, Xin Tong, Weihua Tong, Jiansong Deng, Falai Chen, and Xiuping Liu. Cost-effective printing of 3d objects with skin-frame structures. *ACM Trans. Graph.*, 32(6):177:1–177:10, November 2013.

[34] Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. Structural optimization of 3d masonry buildings. *ACM Trans. Graph.*, 31(6):159:1–159:11, November 2012.

[35] Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. Motion-guided mechanical toy modeling. *ACM Trans. Graph.*, 31(6):127:1–127:10, November 2012.