# Interactive Robogami: Supplemental Material

In this supplemental material we the discuss details of the three main components of our system: database, interactive design, and fabrication. We also show results and discuss limitations and future work.

## 1 Expert Data

### 1.1 Foldable Robots

Our database consists of robot designs that are functional and also fabricable using our 3D print and fold method. The designs each contain three main types of information: *geometric* information about the 3D robot and its 2D unfolding, *hierarchical* information about how the design can be semantically decomposed into smaller subcomponents, and *connection* information describing how subcomponents are geometrically and kinematically assembled. All information about the robot's geometry and kinematics are parametric, as discussed in section 1.2. Our database builds upon the work in [Mehta and Rus 2014], which outlined a Python API for designing the geometry of cut-and-fold robots. Specifically, we have extended the concept of a 'design' to include information about the robot's intended motion, which is important for both fabrication and simulation.

**Geometry** The geometric description of the robot contains information about both its folded 3D representation and its 2D unfolding. The 3D representation consists of oriented polygonal faces connected at their edges. The 2D information is similarly comprised of polygons, but here edge data is also associated with connection information, which is used during fabrication. The vertices and polygons of the 3D and 2D geometries have a one-to-one correspondence with each other. Although the 3D folded state could be computed with only the 2D information and the fold angles on the edges, pre-computing the 3D geometry allows for the geometry to be linearly parameterized before being loaded into the UI, enabling efficient manipulation and rendering.

**Hierarchy** In addition to the geometry, each design includes a representative hierarchy and connectivity graph (ref. Figure 1). The hierarchy is used to semantically represent subcomponents. This representation allows for composition of new designs using small grouping (e.g., single legs in Figure 1), medium groupings (e.g., leg pairs), or large groupings (e.g., full set of four).

**Connections** The connectivity graph describes connections between the kinematic links in our robots. Figure 1 shows an example of a collection of links for a simple walking robot and illustrates the kinematics of the connections. Connections may be articulated, in which case we refer to them as *joints*. Connections contain information necessary for composition, simulation, and fabrication.

Connections contain information about the two edges being connected and a geometric center at which the two connecting edges are centered. Note that we require that two connected edges be the same length so that the endpoints of each of these edges share the same spatial coordinates. Articulated connections contain in addition information regarding principal axes of motion, the types of motion (rotational or translational) that occur along those axes, and a motion sequence, encoded as a sequence of (waypoint, timestamp) pairs, for each axis in the connection. We take into account four types of combinations: a single axis of rotation, which describes the *revolute* motion, three orthogonal axes of rotation, which describe the classical *ball-and-socket* motion, a single axis of translation, which describes *prismatic* motion, and no axes for a fixed connection. Joints contain upper and lower limits for the range of motion along each axis.

Finally, robot designs contain information about the fabrication of each connection. We allow for many fabrication options for each connection depending on the connection type and the joint limits (ref. section 3). Users have the option of specifying the print type of the connection, or of specifying no physical connection despite a semantic connection. This is useful if users wish to place a motor at the joint. Flexibility in cases like this and extensibility in particular are examples of why we keep the fabrication description separate from the kinematic description.

### 1.2 Parameterization

Each design $D$ in our database is a parametric shape that can be written as:

$$D = \{\mathbf{q}, \mathcal{A}, F\} \tag{1}$$

where $\mathbf{q}$ are the degrees of freedom for the template; $F$ is a deformation function that, given $\mathbf{q}$, computes a new design; and $\mathcal{A}$ is the feasible space of $\mathbf{q}$, which is chosen to ensure that the geometry produced by a template remains fabricable and collision-free.

In our implementation, $F$ is a linear parametric function of $\mathbf{q}$ and $\mathcal{A}$ is a set of linear constraints on the $\mathbf{q}$. Because our geometry is linearly parameterized, the centers and axes of our connections are also necessarily parameterized. The linearity allows us to optimize part placement and perform stability analysis at interactive rates.

## 2 Interactive Design

### 2.1 Design Workflow

Our system is based on a design-by-example workflow. The user interface is illustrated in Figure 2. Icons that link to components of the database are displayed on the left; and the canvas in the middle is used to design a new model. Users compose parts by dragging them onto the scene, and they can also remove selected parts at any level of the hierarchy. The right panel is a group of windows that show the user supplementary information regarding this design, such as the 2D view of the model, the printable mesh, joint information, and other design properties.

To create a new design, the user drags in new parts and the system guides the user in positioning them on the scene and connecting them to the working model. The user can also vary the shape of any component in the database by manipulating its template parameters. Our system handles composition to ensure that the working model, like the components of the database, is a hierarchical, parametrized template. Therefore, users can continue to manipulate parameters after components are assembled. The system also guides the user though composition by guaranteeing fabricability and suggesting connections. This includes handling 3D and 2D constraints as well as the motion sequence.

Finally the system evaluates the design properties such as stability, speed, and fabrication cost. The interface allows users to visualize an animation of the ground locomotion of the composed design while highlighting unstable parts of the motion sequence in red. If a motion is unstable, arrows are drawn on the design during manipulation to indicate the direction of change that will improve stability. (See Figure 2).

### 2.2 Composition Tool

The composition tool aids the user throughout the design process in two ways. First it suggests placement for parts that the user

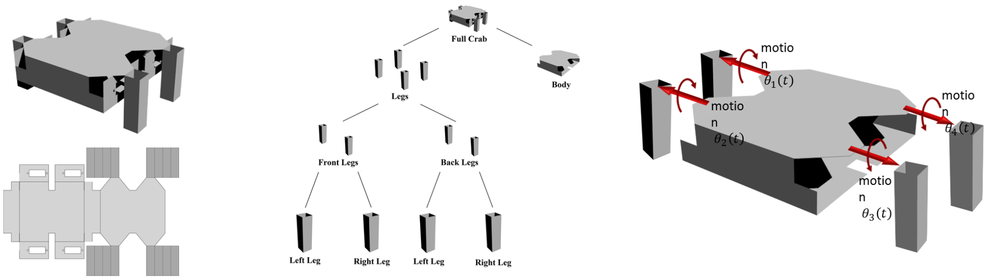


**Figure 1:** *From left to right: a design example of a four legged robot with its corresponding unfolding, the hierarchical tree, and a visualization of the connections*

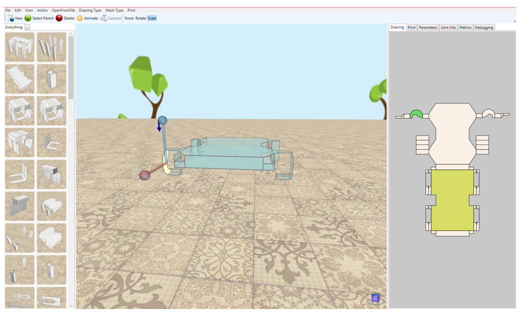

**Figure 2:** *The user interface. Icons that link to components of the database are displayed on the left, the modeling canvas is in the center, and the 2D design is displayed on the right. We show an unstable design that is being manipulated. The arrow indicates that making the legs shorter improves stability.*

manipulates in the UI. Second, it connects the parts together once the user is satisfied with the configuration.

**Part Placement** Suggestions for part placement are made by incorporating information from the database and from the user. Suppose the user adds a part $A$ that originated from design $D$. The connectivity information that $A$ used to have in $D$ will remain with $A$ as it is snapped to the design $B$ that the user is building. We then use the connectivity information in both $A$ and $D$ to choose pairs of an edge in $A$ and an edge in $B$ through which $A$ can be connected to $B$. For each of these pairs, we attempt to snap the edges together by rotating $A$ if necessary and then running an optimization with the goal of making the two edges coincide. Finally we apply the transformation that gives the least cost from the least-squares optimization.

Formally, part placement involves solving a quadratic optimization that select the shape parameters. This quadratic formulation follows from the fact that parameters are linear and is similar to the methods discussed in [Schulz et al. 2014]. Since the cost depends on user defined positioning and scaling, as they continue to manipulate the added part the system updates the snapping configurations.

**Connection** Once the user is satisfied with the part placement, the system will create a composed design.

To connect two parts the system needs to infer the type of joint that should be used to connect the new adjacent edges (hinge joint, ball and socket joint, etc.). The joint information includes not only the fabrication method but also the motion that will be used to simulate the resulting robot. The system makes a suggestions for each new edge pair based on the information of the original database designs $D$ and the user can make modifications using the UI.

Along with the 3D information, the 2D design also needs to be updated. The system joins the two 2D designs and computes the new 2D design that will fold to the snapped 3D robot. We check for collisions on the 2D plane and highlight them in red.

Finally, the new part is added to the hierarchical tree. the system adds the new part as a sibling of the part to which it connects, creating a new root if necessary. The system also adds new constraints to ensure that the two connected edges will always coincide during future user manipulations.

### 2.3 Simulating Capabilities

Though our composition process guarantees that we can fabricate a model that will match the appearance specified by the user, there is still the question of how the model will behave in the physical world. We therefore simulate the ground locomotion and display it to the user with an animation. Parts of the motion that are unstable are highlighted in red. We also use the results of the simulation to provide a feedback loop to the user, guiding them during manipulation toward stable configurations (ref. Figure 2).

**Animation** We animate the models by discretizing time and computing the geometry at each timestamp. For each timestep, we consult the robot's connections graph, which must be a tree, and we compute the state of the the equivalent kinematic chain. We then compute a rigid transformation of the model by applying two assumption. The first is that the quasistatic approximation holds, i.e., that dynamics do not play a part in the robot's locomotion. Thus, if the model is placed at a given position, gravity will make it rotate and translate so that the lowest points touch the ground. The second assumption is that the robot locomotes without slip. We enforce that points that are in contact with the ground on two subsequent timestamps do not move. Both of these assumptions hold for the robot designs in the database.

Under the first assumption, we compute the rigid transformation that makes the plane formed by the 3 lowest points coincide with the ground plane. For the second, we take the intersection of the set of points that are in contact with the ground on both the current and the previous timestamp. We then compute a rigid registration between these sets of points (which can be done in 2D since the z axis is fixed). If the resulting rigid transformation does not result in an exact mapping between the two point sets, the algorithm returns an error that indicates that the robot slips during locomotion.

**Stability** To evaluate the stability of the model, we compute the distance from the projection of the center of mass onto the ground plane with the convex hull of the points in contact with the ground. Though this has to be done for every timestamp it is quite fast since the contact points are already computed during the animation and the center of mass can be evaluated efficiently by exploring the parametric representation. Since we are working with a foldable design, where each vertex is a linear combination of the template parameters $q$, the center of each face can also be written a linear function of $q$. In addition, the weight of each face is proportional to the face area, which can be written a quadratic function of $q$. Therefore, we precompute these functions, and then, given a parameter configuration and an animation timestamp, we compute the center of mass by evaluating each function, applying the relevant rigid transformations for each face center and then performing a weighted average based on each face area.

Because the stability measurement can be done in real time, we can use finite differences to determine the how much this measurement improves at each manipulation direction. This allows us to provide interactive feedback to the users though arrows that indicate the how the robot can be manipulated to improve stability.

**Additional Metrics** Finally we can evaluate speed and fabrication cost using the previous computations. We compute the distance traveled by adding up the distance between the projected centers of mass at each pair of subsequent timestamps. We divide this by time to compute the speed. We evaluate the fabrication cost by the computing amount of printed material, which can be approximated by adding up the areas of all faces. These additional metrics are displayed to the user on the the UI tabs.

# 3 Fabrication

Our fabrication process involves two main steps. First, we convert the 2D body design into a mesh that we print using a 3D printer. Then, we attach actuators and control circuitry to make the robot functional, and we fold the print into a 3D robot body.

## 3.1 Printing

By the end of the design phase, the user has created a 2D fold pattern with edge connections annotated by joint type. Our system converts the pattern into a 3D printable mesh using the following procedure.

First, for each edge connection, the system generates a 3D model of the corresponding connection type. We have designed four types of connections, shown in Figure 3. Hinge/folds enable single-axis rotational motion and can also be used as folds for the assembly process. Prismatic joints allow single-axis translational motion, and ball and socket joints allow three-axis rotational motion. Teeth are used to connect the edges of two non-adjacent edges at a fixed angle. The models for each connection are parameterized according to the location of the connection in the overall model, the length and angle of the connection, and the joint limits if the connection is a joint. Each model is designed in two pieces, with one piece aligned to each edge forming connection. In this way, two faces that should be connected in the 3D design but that are printed nonadjacent can be snapped together during assembly.

Second, the system generates models of the faces of the fold pattern. For each connection, the faces being connected are shrunk along the normals of the connection edges by half the width of the model generated for that edge. This ensures that the printed faces do not interfere with the mechanics of the connection. All faces are then extruded to 1 mm, which we chose as the thickness that produces rigid faces while allowing almost $\pi$ rad fold angles.

Finally, all of the models are merged into a single print, and the design is printed using a 3D printer.

## 3.2 Assembly

We attach a servomotor to each actuated joint and control them using an Arduino Pro Mini programmed to follow the motion sequence designated by the user. We then fold the print into its final 3D form.

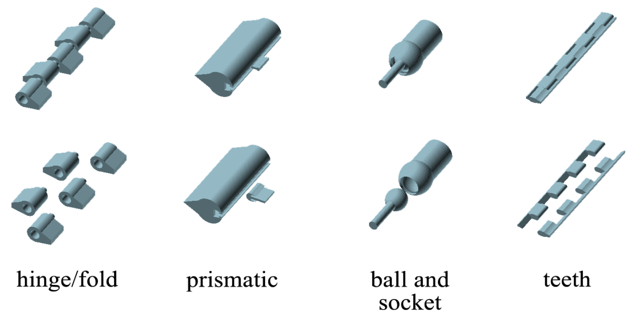


**Figure 3:** *Printable, snappable joints designed used in STL generation. The assembled joints are shown in the top row, and the individual pieces on the bottom.*

# 4 Results

We demonstrate the capabilities of our system by designing and building a variety of different robots. Figure 4 shows a set of virtual models that were designed using our tool. We highlight the different parts that were composed together in different colors. Note that we can explore the hierarchical representation to compose parts using smaller or larger substructures.

We tested the full pipeline to create a physical functional prototype for a biped, a multi-legged crawler, and a wheel-based robot, illustrated in Figure 5. These models were created by combining parts from multiple designs, also shown in the figure. The models could be assembled in 20 to 30 minutes.

# 5 Limitations and Future Work

As with all data-driven methods, the main limitation of our system is that we are restricted to the designs in the database. An interesting direction of future work is to extend our database to robots that go beyond ground locomotion. It would also be interesting to include in the database 3D parts that are not necessarily foldable but that could still be fabricated with our system since it uses a 3D printer.

Another current limitation is that suggestions of connections and articulations are done locally. It would be interesting to further explore the database to propose connections based on more global functionality.

Finally it would be interesting to further exploit the simulatable properties (currently speed, stability and fabrication cost). For example we can globally optimize over the design parameters to minimize user specified costs. Also, while the parameters that are taken into account now are geometric, in the future we could also include parameters in the motion sequence description.

# References

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.

SCHULZ, A., SHAMIR, A., LEVIN, D. I. W., SITTHI-AMORN, P., AND MATUSIK, W. 2014. Design and fabrication by example. *ACM Trans. Graph. 33*, 4 (July), 62:1–62:11.

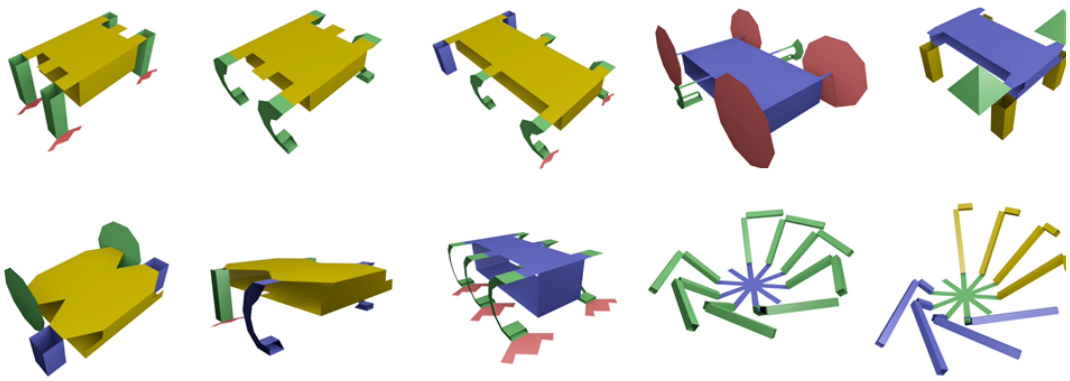

**Figure 4:** *Examples of designs that were built using our system. Different colors indicate different parts that were added to the design.*

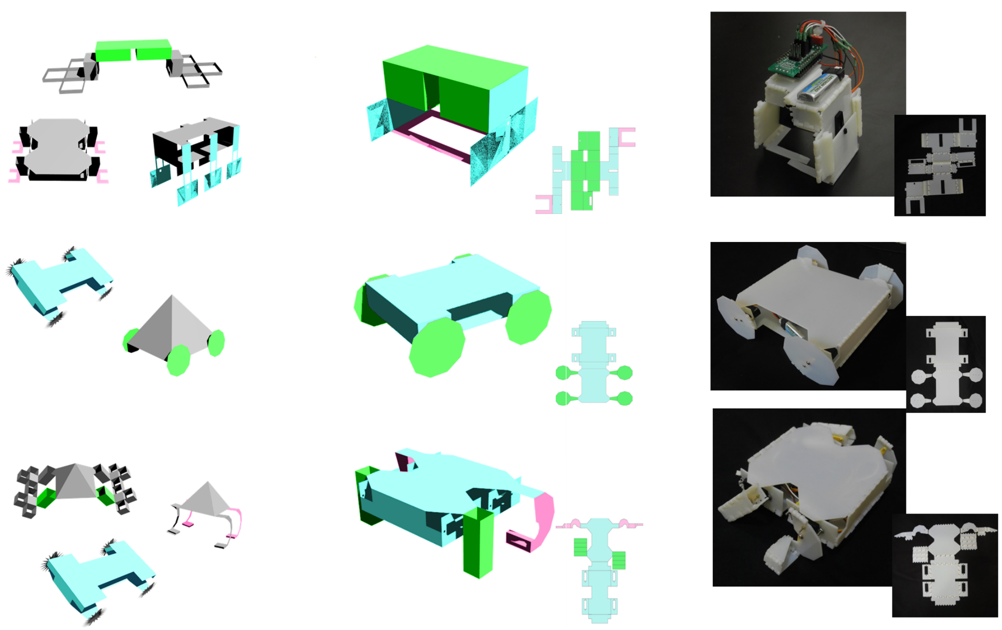

**Figure 5:** *From left to right: input designs (with used parts highlighted), models created using the system, and fabricated results.*