# Assembling and Disassembling Planar Structures With Divisible and Atomic Components

Yinan Zhang, Emily Whiting, and Devin Balkcom

***Abstract*— This paper considers an assembly problem. Let there be two interlocking parts, only one of which may be cut into pieces. How many pieces should we cut the divisible part into to separate the parts using a sequence of rigid-body motions? In this initial exploration, we primarily consider 2-D polygonal parts. This paper presents an algorithm that computes a lower bound on the number of pieces that the divisible part must be cut into. This paper also presents a complete algorithm that constructs a set of cuts and a motion plan for disassembly, yielding an upper bound on the required number of pieces. Applications of the future extension of this paper to 3-D may include robot self-assembly, interlocking 3-D model design, search-and-rescue, packaging, and robotic surgery.**

***Note to Practitioners*—This paper presents an opposite problem of immobilization or caging. Given two interlocking parts, only one of which may be cut into pieces. How can we unimmobilize or uncage one from another? We explore this problem in two aspects: the lower bound and the upper bound on the number of pieces that the divisible part must be cut into. This paper is an early theoretical exploration of this problem. We verified our methods in a virtual 2-D environment instead of building physical structures. Extension of this paper to 3-D could have many applications, including robot self-assembly, 3-D fabrication model design, search-and-rescue, packaging, and robotic surgery.**

***Index Terms*— Assembly, interlocking structure, manipulation, manufacturing, material/parts handling, theoretical foundations.**

## I. Introduction

ASSEMBLY of parts is one of the oldest problems in robotics. This paper considers a variant of the assembly problem in which some parts can be cut into pieces and others cannot be. How many pieces must an amber fossil be cut into to extract a fly? How many pieces must a model ship be broken into in order to construct a ship-in-a-bottle? How many pieces must rubble be cut into to rescue an injured person? How should styrofoam packaging be assembled to support a delicate object for transport?

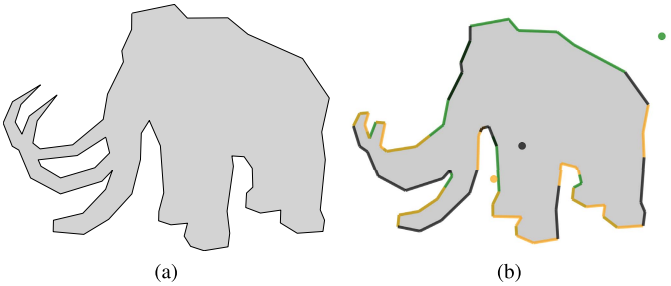



Fig. 1. Mammoth in the ice and one necessary solution to remove ice without damaging the mammoth body. Rotations are all in negative (clockwise) direction here. (a) Mammoth body (gray) inside an ice cube (white). We want to separate the mammoth and the ice (except the hole) by breaking the ice into pieces. (b) Edges with the same color can be locally separated from the atomic part using a single rotation about the corresponding rotation centers.

As an initial exploration, we consider planar devices composed of one polygon of each of the two material types. We allow arbitrary rigid body motion of the parts after cutting; the cuts may be along arbitrary curves. Fig. 1(a) shows an example of a mammoth in an ice cube. The gray material may not be cut, but the white material may be cut to be separated from the indivisible, or *atomic*, part.

We provide algorithms to find lower and upper bounds on the number of pieces that the device must be cut into. We also provide a provably complete algorithm for design and for determining an assembly sequence. Fig. 1(b) shows three rotation centers that could be used to locally separate ice edges of corresponding colors from the mammoth; the ice must be cut into at least three pieces. However, there is no guarantee that three is a sufficient number, because global properties of the geometry also matter; we find that cutting into 61 pieces is sufficient [see Fig. 10(c)].

We believe that extension to 3-D would have practical value for many problems in 3-D printing and prototyping for robotics. Many structures cannot be printed out of a single material; robots may contain atomic components, such as motors, wires, microcontrollers, and batteries that cannot be cut into pieces, supported by a rigid but divisible structure that fits around and supports the atomic components. We can also imagine applications in other areas, including search-and-rescue and robotic surgery.

However, the focus of this paper is not on applications, but on the exploration of a fundamental robotics problem: the diffi-

Manuscript received May 2, 2017; revised January 9, 2018; accepted February 9, 2018. This paper was recommended for publication by Associate Editor R. Alterovitz and K. Bekris and by Editor S. Reveliotis upon evaluation of the reviewers' comments. This work was supported by the Dartmouth Neukom Institute for Computational Science, Dartmouth College. *(Corresponding author: Yinan Zhang.)*

The authors are with the Department of Computer Science, Dartmouth College, Hanover, NH 03755 USA (e-mail: yinan.zhang.gr@dartmouth.edu).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. The Supplementary Material contains slides for the talk at WAFR 2016 conference. It explains two parts of the paper: a graphical method for computing a lower-bound of the number of cuts a divisible part must be cut into two separate from an interlocked atomic, and a method for computing an upper-bound of the number of cuts and planning of extraction paths. This material is 2.96 MB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2018.2809595

culty of assembly, measured by the number of required pieces, if some of the parts can themselves be disassembled. Such analysis of lower and upper bounds on *physical complexity* forms a useful basis for thinking about robotics problems, just as bounds on computational complexity are useful in computer science. Because the difficulty of assembly appears to depend on the shapes of the parts in nontrivial ways, we cannot simply provide an interesting bound directly; however, we can devise algorithms that compute bounds for an input shape. These bounds may give insights into the question of which shapes are hard, which are easy, and how to design appropriate shapes.

There are both local and global properties of the geometry that may cause a part to need to be cut into many pieces. Locally, two contacting rigid bodies may be interlocked, in the same way that a robot grasp may interlock with a part, effectively forming a single rigid body that must be cut to allow separation. Globally, there may be narrow openings through which only small parts can fit the classic piano mover's problem.

When computing a *necessary* number of pieces, we focus on the local geometry, and relax or ignore the global geometric constraints. We consider cutting the interlocking parts into sufficiently many pieces such that there is no single immobilizing grasp, using a geometric method inspired by Reuleaux's method [21], [25] to formulate the problem as a minimum-set-cover problem. Computationally, minimum-set-cover is NP-complete, but can be approximated in polynomial time to within a logarithmic factor. In future work, better lower bounds might be found by also considering global constraints.

We also give an algorithm to compute a *sufficient* number of pieces, by constructing cuts and an assembly order that respect global and local constraints. We prove that as long as atomic components do not contain voids, the parts can be cut into a finite set of pieces and disassembled using only translation; rotations are not required.

This paper extends [42] to include a linear-algebraic formulation of the lower bound algorithm and also discusses the geometry of shapes that require so many cuts to disassemble.

### A. Related Work

This paper is closest in spirit to work on $k$-moldability. In the $k$-moldability problem, a separable $k$-piece mold is taken apart using a single translation per piece to expose a molded atomic part [17], [23]. Ravi and Srinivasan [24] give a list of criteria to aid the engineer in making decisions of parting surfaces. Priyadarshi and Gupta [23] used accessible directions to decompose molds into a small number of pieces. Exact-cast-mold design methods require models to be moldable or resulted in a large number of mold pieces. Herholz *et al.* [15] deform a model into an approximate but moldable shape, and then decompose mold pieces.

The primary contribution of this paper is the relaxation of the requirement that the mold can be separated using single translations; this allows the study of the fundamental theoretical limits of disassembly. Most of the structures studied in this paper are not $k$-moldable, because there is no set of directions from which all of the divisible structure is visible; some portions of the structure are occluded. We show that in fact, any structure without inaccessible voids can be disassembled using only sequences of translations. The lower and upper bounds that we study are true physical bounds—they hold over *any* sequence of rigid body motions, not just single translations or rotations.

This paper is also inspired by Snoeyink's work on the number of hands required to disassemble a collection of rigid parts [32]. Because we allow parts to be cut, simultaneous motions are not required for disassembly. In the *Carpenter's Rule* problem studied by Connelly *et al.* [10], Streinu [36], and others, the rigid pieces are also all atomic and connected by joints, typically requiring many simultaneous motions.

This paper is, therefore, somewhat closer in spirit to work by Wang and Balkcom [39] that studies the number of fingers needed to tie a knot—in that work, the string is treated as a collection of rigid bodies, but the joints may be placed arbitrarily, essentially cutting the carpenter's rule, without disconnecting the pieces. This paper is also close in spirit to work by Bell *et al.* [5] that studies the number of pieces that a mechanical knotting device must be cut into to extract the knotted string.

There has been significant work in the graphics community in computational fabrication. Song *et al.*'s [33] approach to fabricating large 3-D objects was to break the shell of the object into pieces and assemble after fabrication. Hu *et al.* [16] presented a method to decompose 3-D object into a set of pyramidal shapes such that no support material is needed when 3-D printing the model. Fu *et al.* [14] and Song *et al.* [34] studied computational interlocking furniture design.

Our approach to the lower bounds problem, in particular, grows out of seminal work on immobilizing rigid bodies or *grasping*. Traditional geometric approaches to grasping attempt to prevent all possible sliding and rotational motions of a polygonal object by placing fingers around the object. Reuleaux [25] is credited with the concept of *form closure*. Mishra *et al.* [22] proved the sufficiency of four fingers to immobilize any polygonal object. Czyzowicz *et al.* [11], [12] showed that polygons without parallel edges can be immobilized using three fingers. Rimon and Burdick [27], [28] showed how two-finger grasps can be analyzed using the *second-order immobility*. Cheong *et al.* [7] provided an algorithm to compute all immobilizing grasps of a simple polygon. Cheong *et al.* [8] also showed that $n + 3$ contacts suffice to immobilize a chain of $n$ hinged polygons.

A polygonal object can be *caged* by surrounding an object with fingers, such that the object has some freedom locally but cannot escape the cage. Some of the earliest work on caging was by Rimon and Blake [26]. Allen *et al.* [1] and Vahedi and van der Stappen [38] proposed algorithms to find all caging grasps of two disk fingers. Erickson *et al.* [13] studied the case of three-finger caging for an arbitrary convex polygon. Makita and Maeda [20] extend the caging problem from 2-D to 3-D with multifingers. This paper, instead of caging an object, can be viewed as removing contacting pieces to uncage a polygon in the plane.

A classic problem of *self-assembly* is to move a set of small robots to specified target positions; some of the challenges with narrow corridors and coordination are similar to those

faced in this paper. Kotay *et al.* [18], for example, designed a robotic module, groups of which aggregate into 3-D structures. Rus and Vona's [31] early work on the crystalline robot presented an algorithm to do self-reconfiguration. Recently, working on the problem of scale, Rubenstein *et al.* [30] provided an algorithm for moving kilobots one-by-one to form certain planar shapes. Arbuckle and Requicha [2] allowed identical memoryless agents to construct and repair arbitrary shapes in the plane. Zhang *et al.*'s own work [40] on assembly of interlocking structures, nine kinds of blocks are used to build large-scale voxelized models such that all blocks are interlocked and the whole structure is rigid as a whole. Self-assembly and modular robots in the presence of obstacles have also been extensively studied. Becker *et al.* [4] proposed an algorithm to efficiently control a large population of robots in this scenario. Rubenstein *et al.* [29] used multiple robotic units to manipulate the positions of obstacles.

## II. Computing a Lower Bound on the Number of Pieces

Let $A$ and $B$ be two interlocked parts, where $A$ is atomic and $B$ is divisible. What is a lower bound on the number of parts that $B$ must be partitioned into such that $A$ and $B$ can be separated, so that $A$ and parts of $B$ are at any desired arbitrary distance? In this section, we propose algorithms to compute such lower bound for any given shapes. We introduce two approaches to attack this problem: a geometric method inspired by Reuleaux's graphical method for analyzing constrained motion of planar rigid bodies and a linear-algebraic approach.

### A. Graphical Method Analyzing the Rotation Centers of Boundary Points

The approach is inspired by the analysis of contact modes for contacting 2-D rigid bodies [3], [21], as well as by Reuleaux's graphical method for analyzing whether a collection of points fully constrains (or in our case, is constrained by) the motion of a rigid body [25].

We first replace the divisible body $B$ with a collection of points $P$ from $B$ along the boundary of $A$; whichever points we choose, *at least* these points must be separated from $A$ using a collection of rigid-body motions. For simplicity, we place three points per edge: one in the center and one at each endpoint. Choosing more points may allow a larger lower bound to be computed at the cost of some additional computation.

Now consider any subset of these points. Can this subset be contained in a single rigid piece after cutting, in such a way that the rigid piece may be separated from $A$? In order to separate this piece from $A$, there must at least instantaneously be a single rigid body motion that does not cause collision for any of the points.

Every motion of a planar rigid body is instantaneously a rotation or a translation. Does there exists a translation direction or a rotation center that allows separation? How do we compute good (large) subsets of $P$ that can move together without considering the power set over $P$?

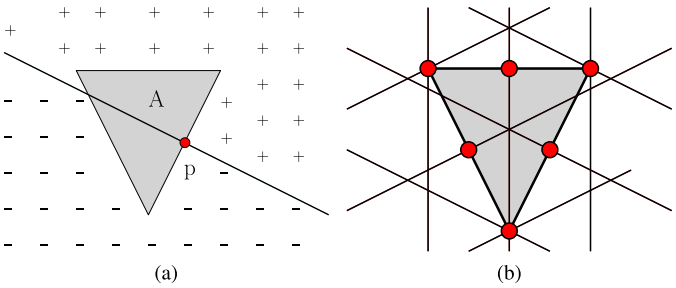



Fig. 2. Reuleaus's graphical method for analyzing if a contact point or a collection of points fully constrain(s) the motion of a rigid body. (a) Contacting point $p$ can rotate in positive direction about centers in $+$ area and in negative direction about centers in $-$ area. (b) Normal lines through the contact points form a set of cells, such that the number of points that may be separated from the gray part is maximized by choosing a rotation center at a cell vertex.

*Theorem 1:* If $n$ points $P$ of a planar rigid body $B$ are in contact with a polygonal rigid body $A$, then there are at most $n(n-1)/2+2n$ maximal subsets of $P$, such that each subset may be moved together as a rigid body without colliding with $A$, and any other noncolliding subset is contained within one of the maximal subsets.

*Proof 1:* We would like to group subsets of points in $P$ and see if they can be separated from $A$ as a group. Let us first consider rotations. To find a compatible group, we might choose a particular rotation center and a positive or negative direction for rotation. Then, find all points in $P$ that separate from (or at least do not collide with) $A$ under that motion; we have found a compatible subset. This is our basic approach; but how many potential rotations must we consider, and how many compatible subsets may be generated?

Consider a rotation center $r$ somewhere in the plane, with an associated direction, either positive (counterclockwise) or negative rotation (clockwise). Reuleaux's method makes use of the fact that for a particular point $p_i$, for most choices of rotation center, one direction of rotation (either positive or negative) is permissible, while the other causes collision of $p_i$ with $A$. Along the normal to the edge of $A$ at $p_i$, either negative or positive rotation is possible. Let $P_r \subset P$ be the set of points compatible with rotation center and direction $r$. If we move $r$ along a continuous trajectory, membership in $P_r$ only changes, as $r$ crosses one of the normals through one of the points in $P$. The constraint is least restrictive along the normals themselves; so to compute the *maximal* subsets of compatible points, such that any other compatible subset is either a singleton or a subset of a computed subset, we need only consider rotation centers at the intersections of the normals, as shown in Fig. 2. The normal lines form an *arrangement* [35], [37], and there are $n(n-1)/2$ possible intersections, each with at most one corresponding maximal subset (see Fig. 2). Once candidate maximal subsets have been generated, discard any that are contained within other computed subsets.

Any translation of a set of points can be viewed as a rotation about a center infinitely far. To test translation directions, for each point $p_i$, we choose two directions each along its normal directions of both connecting edges. (Any point on the edge is also considered a vertex connecting two parallel edges.) This is equivalent to selecting points infinitely far from $p_i$ on both normal direction lines. Each point $p_i$, if included in a subset,

forbids an open half-plane of translation directions. So for each point, it is sufficient to test two translation directions, each corresponding to sliding in one direction or another along the point's normals. Collect all $2n$ directions, and for each direction, test the remaining points against that direction to generate candidate maximal subsets.

In total, we have done $n(n-1)/2 + 2n$ tests representing the same amount of subsets of $P$. ■

Theorem 1 implies an algorithm for computing a lower bound on the number of pieces that the divisible part must be cut into to allow assembly or disassembly. Compute the maximal subsets as suggested; then, solve the minimum-set-cover problem to find the minimum number of such sets needed to separate all points in $P$ from $A$. If the number of maximal subsets is small, minimum-set-cover may be solved exactly. The pseudocode is described in Algorithm 1. The simple examples presented in this paper were solved exactly using integer-linear programming. If there are a large number of subsets, then a greedy approach yields a solution in polynomial time, with guaranteed logarithmic approximation quality.

**Algorithm 1**: Graphical Method to Compute the Lower Bound

1: **procedure** COMPUTELOWERBOUND(P)
2: **for all** point $p \in P$ **do**
3: Generate two lines going through $p$ and perpendicular to each connected edge.
4: Add the two lines to set $L$
5: **end for**
6: **for** every pair of non-parallel lines in $L$ **do**
7: Compute intersection $i$.
8: $F(i,+) \leftarrow$ find all points in $P$ with feasible motions using $i$ as rotation center in the positive direction.
9: $F(i,-) \leftarrow$ find all points in $P$ with feasible motions using $i$ as rotation center in the negative direction.
10: **end for**
11: Run minimum set cover algorithm over $F(i,+/-)$ for all intersections such that all points in $P$ are covered.
12: **end procedure**

Based on the angle of two edges connecting a point, contact points are classified into three kinds: convex vertex, concave vertex, and edge vertex. An edge vertex connects two parallel edges. For each edge connecting a vertex, we draw a line parallel to the normal and going through the vertex (lines 2 and 3). With two lines, the plane is divided into four areas allowing positive or negative rotation centers. Each area is closed on the boundaries. Fig. 3(a) is an example of a convex vertex; the space is divided into four areas: $A_1, A_2, A_3$, and $A_4$, where $A_1$ allows positive rotation centers for vertex $P_1$, $A_3$ allows negative rotation centers, and $A_2$ and $A_4$ allow positive and negative rotation centers. Mathematically, each area is represented using two linear inequality constraints. Given a rotation center and a rotation direction, we test if a vertex can rotate about the center by checking which area the rotation center falls into. Each rotation center and rotation direction is tested for concave vertex and edge vertex separation in the same way.

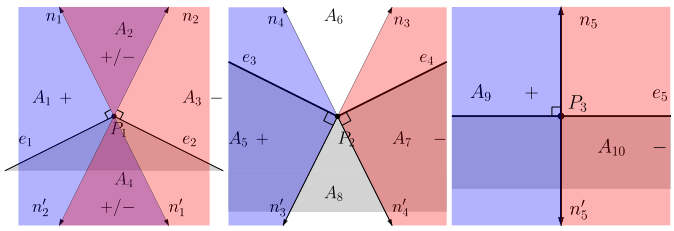



Fig. 3. Three kinds of vertices in a polygon. The blue areas allow positive rotation centers, whereas the red area allows negative rotation centers. No points can rotate about centers in the empty areas. (a) Convex vertex. (b) Concave vertex. (c) Edge vertex.

For each normal line going through contact points, the algorithm finds intersections with all other normal lines. Therefore, a set $I$ of $O(n^2)$ intersection points are maintained, where $n$ is the number of normal lines. For each intersection point, we test both positive and negative rotation centers against every contact point on the atomic part (lines 8 and 9). Let $F(i,d)$ be the set of contact points that can rotate about intersection $i$ in direction $d$ without colliding with the atomic part instantaneously. The algorithm also tests $F(i,d)$ for all $i \in I$ and $d \in \{+,-\}$ and eliminates the sets containing the exact same contact points. (If a set is a subset of another, we only keep the larger set.)

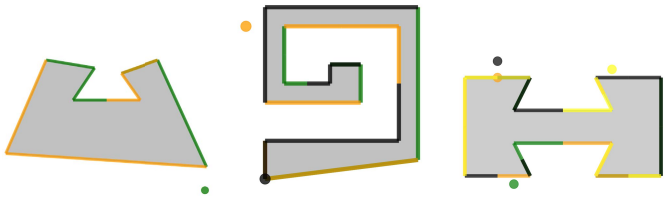
Because the intersections are on the boundary of areas, testing if a point can rotate about an intersection point by checking linear inequalities might suffer numerical issues caused by floating point arithmetic. A solution is to triangulate the intersections and use the triangle centers as rotation centers to test against all contacts on the atomic part boundaries.

We now have a set of sets each containing contact that can rotate about the same center in the same direction. With a minimum-set-cover algorithm, we find the minimum number of sets that covers all contacts (line 11). Contacts in the same set are partitioned into the same piece of a divisible material to be separated from the atomic part. The minimum-set-cover problem is NP-complete, so the algorithm to find an exact solution can take a long time if the number of subsets is large.

To accelerate the computation, we may, instead, use a greedy algorithm to rapidly solve the set cover problem approximately. Given a set of $n$ subsets, Chvatal [9] showed that the approximation ratio $H(n)$ of the greedy algorithm is $H(n) = \sum_{k=1}^{n}(1/k) \leq ln(n+1)$. Dividing the result from the greedy algorithm by the approximation factor, we have a lower bound on the number of pieces that the divisible material must be cut into to separate from the atomic part.

Fig. 1(b) shows a solution of the necessary number of pieces needed to extract the planar mammoth from the ice. If two points on the same edge are in the same set, the segments between the points are considered able to rotate about centers in the same region as the two points. In this case, three sets cover all edges.

Edges containing points in the same set are not necessarily connected. Whether there exist cuts to separate edges exactly into the derived sets as connected rigid bodies is an open question, as is whether those bodies can be extracted after



(a) (b) (c)

Fig. 4. Three examples of analyzing necessary number of pieces for the divisible part. In each example, edges with the same color are in the same set. (a) Cavity. Orange edges translate to the left, and green edges rotate about the green point in negative direction. (b) Spiral. Black edges rotate about the black point in positive direction, and orange edges rotate about the orange point in negative direction. Green edges translate to the right. (c) Dumbbell. Four sets of edges in four colors. Each set of edges rotates about centers with the same color. Rotation directions: positive for black, negative for the rest.

TABLE I
LOWER BOUND ANALYSIS EXAMPLES

| shape | edges | largest set size | set cover size | time cost |
|---|---|---|---|---|
| cavity | 8 | 144 | 2 | 0.2018 s |
| spiral | 14 | 612 | 3 | 1.8764 s |
| dumbbell | 12 | 1104 | 4 | 7.2543 s |
| mammoth | 64 | 12012 | 3 | 540.327 s |

initial separation. This technique, thus, yields only a lower bound, and we expect that the lower bound might be significantly improved in future work.

Results for other shapes can be found in Fig. 4. A few statistics are shown in Table I. Time costs were measured on a 2016-model MacBook Pro with a 2.6-GHz Intel processor and a 8-GB 1600-MHz DD R3 memory and are intended only to give a sense of the practicality of analysis of structures with various numbers of edges. From Table I, we can see that with the increase of maximum number of rotation sets, the time cost increases dramatically, as we would expect for an $O(n^3)$ check of rotation centers and a linear-integer program optimal solution to minimum-set-cover; we expect that much larger problems could be solved with good approximation by using greedy minimum-set-cover techniques.

### *B. Linear-Algebraic Method for Lower Bounding the Number of Pieces*

The geometric method described is hard to extend to 3-D parts. Although the focus of this paper is on planar parts, we now show how to design a linear-algebraic approach to the same problem, as a guide for future extension to 3-D.

As in the previous section, consider a finite set of points $P$ contacting the boundary of $A$. Can a subset of points be set in the same rigid piece of divisible part and separate from $A$? In this section, we attack the problem by analyzing the motion of contact points using a set of linear inequalities.

Let $x_i$ be the Cartesian location of point $p_i$, and let $x = [x_0, \ldots, x_n]$ be the vector representing locations of all points. We describe rigid-body motions of the atomic part using (1), where $q$ is the configuration of the atomic part and $J(q)$ is the collection of Jacobian matrices of all points in $P$ the current configuration

$$J(q)\dot{q} = \dot{x}. \tag{1}$$

Consider a point $v$ on an edge $e$. The point has a feasible motion if the angle between its speed $\dot{x}_v$ and the edge normal $n_e$ is smaller than 90°, or

$$n_e \cdot \dot{x}_v = n_e \cdot J_v(q)\dot{q} \geq 0 \tag{2}$$

where $J_v$ is the Jacobian of vertex $v$. For convex vertices connected with two edges, a motion is feasible if the vertex is not penetrating either edges. A concave vertex motion is feasible if the vertex is not penetrating both edges connecting the concave vertex.

Let $N$ be a matrix that collects normals to $A$ at the contacts in such a way that nonpenetrating motion of contacts [see (2)] can be described as

$$NJ\dot{q} \geq 0. \tag{3}$$

For simplicity, we phrase the problem slightly differently and consider motions of the atomic part, which are symmetric to the motions of the divisible boundary. How many rigid motions (with $|\dot{q}| \geq 0$) of the atomic part do we need, such that every point in $P$ does not penetrate for at least one of the motions? Since parts $A$ and $B$ are interlocked, there does not exist a single motion $\dot{q}$ of the atomic part that satisfies (3). We want to find a set of actions $Q = \{q_1, q_2, ..\}$ such that every selected point on the boundary of the atomic part has at least one feasible motion satisfying (2).

Define $M = NJ$. For any particular velocity $u = \dot{q}$, the product $Mu$ has some rows with nonnegative values causing some contacts' feasible motion constraints to be satisfied, and some negative values that cause impingement. Let the contacts with feasible motions be given by

$$s(u) = \{p \in P : p \text{ has } a \text{ feasible motion under } u\}. \tag{4}$$

If $u_j$ is a positive scaling of $u_i$, the $s(u_i) = s(u_j)$. Therefore, we can always normalize the configuration velocity of the atomic part so that $|u| = 1$. Consider the feasible motion constraints of a contact point as half-spaces in a 3-D configuration space. Two linearly independent configuration velocities $u_i$ and $u_k$ have the same set of contacts with feasible motions, if $u_i$ and $u_k$ fall into the same subspace formed by the same set of constraints.

*Theorem 2:* The collection of all feasible motion constraints for every contact point is a set of at most $2n$ half-spaces in a 3-D space partitioning the space into $O(n^2)$ convex cones.

*Proof 2:* The trivial solution $\dot{q} = 0$ tells us that the boundary of every half-space (a plane) goes through the origin. Thus, the 3-D space is partitioned into convex cones.

We next consider the number of convex cones. Let $f(d, n)$ denote the maximum number of cells in a $d$-D space under $n$ cuts. In 2-D, $n$ cuts (each going through the origin) partition the space into at most $2n$ parts, because every cut goes through at most two parts and partitions each part into two parts. Therefore, $f(2, n) = 2n$. In $d$-D space, every cut goes through at most $f(d-1, n-1)$ cells and adds as many new cells. Therefore

$$f(d, n) = f(d, n-1) + f(d-1, n-1). \tag{5}$$

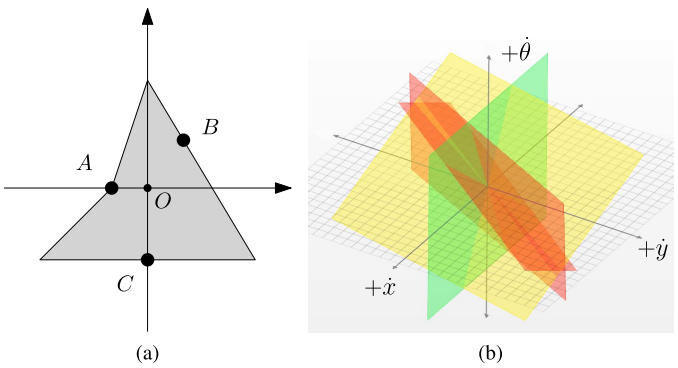



Fig. 5. Analysis of a simple atomic shape and visualization of constraints for points $A$, $B$, and $C$. (a) Simple atomic shape. Three points $A$, $B$, and $C$ are selected for analysis. (b) Two red linear constraints are for point $A$, because it is on a concave vertex. The yellow constraint is for point $B$, and the green constraint is for point $C$.

So $f(3,n) - f(3,n-1) = f(2,n-1) = 2(n-1)$, and $f(3,1) = 2$. Therefore, $f(3,n) = O(n^2)$. ■

Fig. 5 shows an example of analysis using the method proposed in this section for an atomic shape with one concave vertex. The coordinates are: $A = (-1,0)$, $B = (1,4/3)$, and $C = (0,-2)$. Let $(x,y,\theta)$ be the configuration of the atomic shape; constraints for point $A$ are $-3\dot{x} + \dot{y} + \dot{\theta} \geq 0$ and $-\dot{x}+\dot{y}+\dot{\theta} \geq 0$. The constraint for point $B$ is $5\dot{x}+3\dot{y}+2\dot{\theta} \geq 0$, and the constraint for point $C$ is $\dot{y} \geq 0$. Constraints are shown in Fig. 5(b).

Theorem 2 implies an algorithm to compute the lower bound on the number of pieces that the divisible part must be cut into to allow assembly or disassembly. Construct the convex cones in the 3-D space. In each convex cone, select a point and compute the set of contact points that will not collide with the atomic part under this motion. Run minimum set-cover algorithm to find out the minimum number of sets such that all contact points in $P$ have at least one feasible motion. Algorithm 2 gives the pesudocode.

Line 6 of Algorithm 2 selects two linearly independent constraints and computes the null space. This is because any three linearly independent constraints, all going through the origin, intersect only at the origin. Intersections of the null space and a unit sphere are candidates of configuration velocities, and their corresponding contact points with feasible motions are computed (line 7). Since every intersection is on the boundary of a convex cone, using this velocity to compute contacts with feasible motions might also suffer numerical issues. The solution is similar to the graphical method. Since the null space of every two linearly independent constraints is a line, we list all these lines and intersect with a unit sphere. Triangulate the points on the surface of the unit sphere, and use the center of each triangle to compute contact points with feasible motions.

### C. Shape Hard to Disassemble

If the atomic part is a convex shape, the divisible part can always be partitioned into two pieces to be separated from the atomic part. Fig. 6(a) shows an atomic shape with four

**Algorithm 2**: Linear-Algebraic Method to Compute the Lower Bound

1: **procedure** COMPUTELOWERBOUND(P)
2: Split every convex and concave point into two overlap ping points each on an edge connected to the original point.
3: Construct matrix $N \cdot J\dot{q} \geq 0$ as the matrix to describe non-penetrating motion of all contact points. Define $M = NJ$.
4: $R \leftarrow$ all combinations of two linearly independent rows in matrix $M$.
5: **for** combination in $R$ **do**
6: Let $x'$ be solution for $M'x = 0$ and $|x| = 1$, where $M'$ is a matrix with two rows from the combination.
7: $F(x') \leftarrow$ the subset of points with feasible motions under $x'$.
8: **end for**
9: Run a set-cover algorithm over $F(x)$ for all $x$ computed in step 6 to find a minimum number of configuration velocities whose sets of feasible motion points cover all contacts
10: **end procedure**

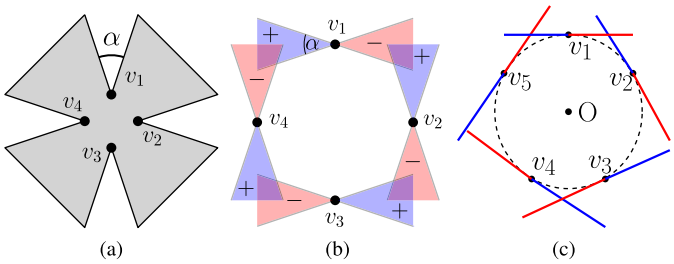



Fig. 6. Two examples of atomic shapes that require divisible part to be disassembled into at least four or five parts to be separated. (a) Atomic part with four concave vertices. Points on the boundary must be partitioned into four pieces to separate from the atomic part. (b) Graphical analysis of concave vertices in Fig. 6(a). Blue areas allow positive rotation centers, whereas red areas allow negative rotation centers. (c) Five concave vertices. No pair can rotate together. Blue areas allow positive rotation centers, whereas red areas allow negative rotation centers.

concave vertices. To separate every point on the boundary of the shape, the divisible part must be broken into no less than four pieces. Fig. 6(b) shows an analysis of the four-concave-vertex shape. Each vertex rotates about a point in the blue region in positive direction or a point in the red region in negative direction. No two points have overlapping positive regions or overlapping negative regions. Therefore, no concave vertices share the same rotation center.

Extending the idea of having no overlapping rotation areas for any pair of concave points, we construct a shape with $n$ concave vertices and require the divisible part to be partitioned into at least $n$ pieces.

*Theorem 3:* Given $n$ concave vertices, there exists an atomic shape whose surrounding divisible material must be disassembled into at least $n$ pieces to separate from the atomic shape.

*Proof 3:* We first draw a circle, equally divide the circle into $n$ arcs, and put a vertex at the center of each arc. The radius

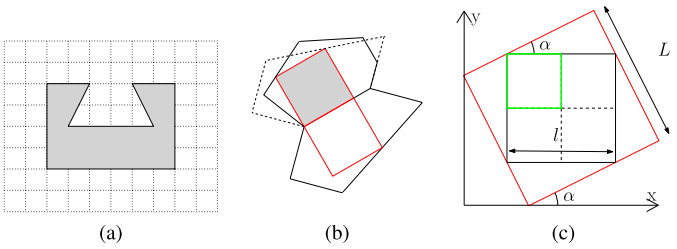



Fig. 7. Gridding the divisible part to find sufficient pieces to separate parts. (a) Two interlocked parts where the gray part is atomic and the rest is divisible. Gridding the divisible part into small enough cells and removing them will separate both parts. (b) Between two adjacent convex shapes, there exists a square that can move freely from one to another without leaving either shape. (c) Partitioning the space using a (green) square of width $l/2$ guarantees that there exists at least one square in the transition square.

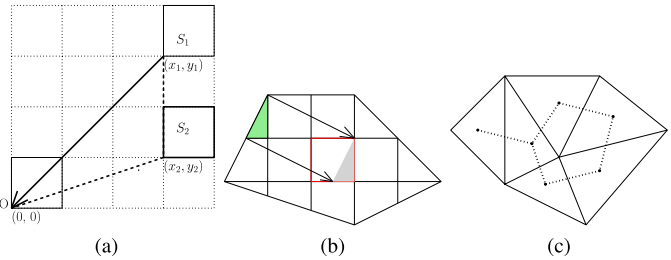



Fig. 8. Moving cells inside a convex shape and between multiple convex shapes. (a) Squares in a grid space can move to any target position with no collision following the order of their distances to the target. (b) Cells can also move to any target square (red) with no collision in the grid space bounded by a convex polygon using the same method. (c) Convex polygons form a graph. Edges are weighted by the size of transit squares connecting polygon pairs. Removing the smallest edges but keeping the graph connectivity improves the grid square size.

of each arc is $2\pi/n$. Let the angle between the two edges of a convex vertex be $0 < \alpha < 2\pi/n$. The angles of cones allowing positive or negative rotation centers are also $\alpha$, and no two positive areas or negative areas intersect. Thus, the surrounding divisible material of any shape with these $n$ concave vertices each has two edges forming an $\alpha$ angle that must be broken into at least $n$ pieces to be removed from the atomic part.

Fig. 6(c) shows an example with five concave vertices. The resulting shape is similar to Fig. 6(a) but with five leaves.

## III. Computing a Sufficient Number of Pieces

In this section, we present a complete algorithm that chooses where to cut the divisible part $B$, and finds a motion plan to achieve separation from the indivisible part $A$. We want to find an upper bound on the minimum number of pieces $B$ that can be cut into to move each piece out of a planar box that contains both $A$ and $B$.

The algorithm first decomposes the part into a small number of convex polygons and moves pieces within these convex shapes; the number of pieces depends on the number and size of polygons. In our implementation, we used a Delaunay triangulation, but better bounds could be achieved by finding larger convex components.

After decomposition of $B$ into convex polygons, consider two adjacent convex polygons that share an edge [see Fig. 7(b)]. Flipping one polygon about the shared edge and intersecting with the other polygon give a new convex polygon. Inside the new polygon, we compute a largest inner square with one edge on the shared edge. This square, called the *transit square*, can move freely between the two convex polygons without leaving the interior.

We would like to find an axis-aligned grid such that at least one complete grid cell fits entirely within the transit squares for each pair of adjacent convex polygons. For each transit square, we find the largest axis-aligned inscribed square, divide the width by two [shown in Fig. 7(c)], and take the minimum over all such values as the width of cells in the grid.

To find the size of the largest axis-aligned square, assume that the width of the outer square is $L$ and the small angle between the $x$-axis and the edges of the square is $\alpha$. There exists another square of edge length $l = L(1 - 2 \cdot \tan\alpha / (1 + \tan\alpha)^2)^{(1/2)}$ inside the outer square.

We intersect the grid cells with the convex polygons to create small pieces that we will call *components*. We will extract the material from each convex polygon; we will say that a polygon that has already had its material extracted is empty, and one that has not is full. We will prove that components can move from one full convex polygon to an adjacent empty polygon without leaving either convex polygon (and thus without collision with atomic parts or uncleared divisible parts), assuming each component disappears once it has completely entered the empty polygon. This is sufficient to prove inductively that the entire structure can be disassembled.

*Lemma 1:* In a planar grid of square cells with a designated target square, continuous translation of each grid cell to the target will not cause collision, if the cells are translated in order of $L_2$ distance from the target.

*Proof 4:* Let $S_1$ be a square whose bottom-left point is at position $(x_1, y_1)$, moving in one direction toward the target square $O$, with bottom-left point at $(x_o, y_o)$. Without loss of generality, assume $x_1 > 0, y_1 > 0$ and $x_o = 0, y_o = 0$, and that the width of each cell is 1. We know that, during the motion, every point of square $S_1$ is bounded by the rectangle $R$ defined by its bottom-left point $(0, 0)$ and top-right point $(x_1 + 1, y_1 + 1)$ [see Fig. 8(a)].

Assume $S_1$ collides with a square $S_2$ whose bottom-left point is at $(x_2, y_2)$, $x_2, y_2 \in \mathbb{Z}^+$. Then, $0 \le x_2 \le x_1$ and $0 \le y_2 \le y_1$; otherwise, no point in the square is in $R$. Because $OS_1$ is along the diagonal of $R$, $OS_1$ is the longest edge in triangle $\triangle OS_1S_2$. So $|OS_2| < |OS_1|$, and square $S_2$ would have been moved before $S_1$ using the proposed order. ■

Although we claim and prove Lemma 1 in the plane, it extends easily to similar results in arbitrary dimensions. We now come to the main result of this section in the following.

*Theorem 4:* Given an intersection of a set of nonoverlapping unit squares (grids) with a pair of adjacent convex polygons, such that at least one complete grid cell is completely contained within each of the transit squares of the polygons, the intersection parts of one convex polygon can be moved into the other convex polygon without collisions, using the intersections of the cells with the polygons as components, assuming that each component disappears once it has completely entered the empty polygon.

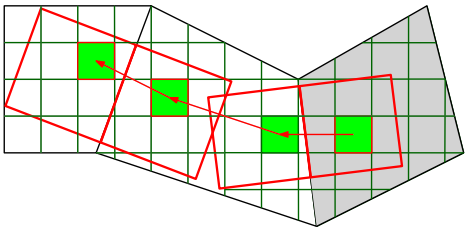


Fig. 9. Moving components through a chain of convex polygons. White polygons are empty, and the gray one is filled. Using the pairs of transit squares, cells can move between any two adjacent polygons, or through a chain of connected polygons.

*Proof 5:* Lemma 1 indicates that cells in a grid can be translated to a target in order of distance without collision. Since all motions of all points in cells are along straight lines during translation, the result extends trivially to a case where the space is constrained to a convex polygon, and the cells are clipped by the convex polygon, as are the previously defined *components*.

We therefore have the following approach. First, empty the transit square in the full polygon into the transit square in the empty polygon, by sorting the grid cells in order of distance from an arbitrarily chosen complete grid cell in the empty polygon and translating those cells to the target in that order; since the pair of transit squares is together a convex polygon, there are no collisions. Then, choose a target cell in the now-empty transit square in the first polygon. Sort the components in the polygon based on their distance from the target cell. In this order, first translate each component first into the transit square, and then into the adjacent empty polygon. ■

### A. Algorithm 1: Simple Separation

The previous theorem suggests a simple, but complete, algorithm for cutting and separating the interlocked parts. First, using triangulation or some other means, decompose $B$ and its containing square (from which we would like to remove $B$) into convex polygons. Polygons within $B$ will be full, and polygons outside of $B$ will be empty. Define a *boundary* polygon as a polygon that is connected by a sequence of adjacent empty polygons (the *exit sequence*) to the outside of the containing square, the *exit*.

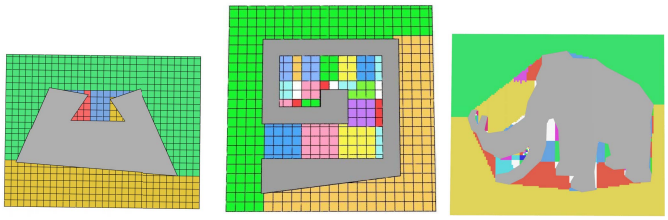
Fig. 9 shows the approach for extraction. Choose a boundary polygon and an exit sequence. Extract components from that polygon one at a time in the order suggested in the proof of the theorem. As each component enters the exit sequence, move it through the sequence of empty polygons to the exit, using translation first to and then through each pair of transit squares along the exit sequence.

A greedy way of finding a good grid size is to first build a graph with convex polygon centers as vertices and convex pairs as edges. Each edge is weighted by the size of transit squares connecting polygon pairs. Keep removing the smallest edges but maintain the connectivity of the graph until no more edge can be removed. Then, compute the grid that fits in the smallest transit square [see Fig. 8(c)].

TABLE II

ANALYSIS OF SUFFICIENT NUMBER OF PIECES FOR A FEW EXAMPLE SHAPES

| shape | components | pieces | decompose time | planning time |
|---|---|---|---|---|
| cavity | 144 | 6 | 0.862 s | 0.822 s |
| spiral | 462 | 26 | 0.786 s | 1.500 s |
| dumbbell | 269 | 8 | 0.529 s | 0.496 s |
| mammoth | 61 | 9954 | 16.816 s | 149.192 s |



(a) (b) (c)

Fig. 10. Decompositions of separable parts into pieces. Colors are reused as needed; each set of the same color components is a single piece. (a) Cavity. The divisible part is divided into six pieces. (b) Spiral. The divisible part is divided into 26 pieces. (c) Mammoth. The divisible part is divided into 61 pieces.

### B. Algorithm 2: Greedy Separation With Grouped Components

There are many shortcomings of Algorithm 1. For example, if there is a tiny shape in the convex decomposition of the divisible part, the cell size will be very small. Inspired by Zhang *et al.*'s [41] work of moving grids along axis, in this section, we propose an algorithm to plan a path for every cell generated by the decomposition algorithm. Based on the paths, we aggregate components into *pieces* and cut only along piece divisions; this will greatly reduce the number of pieces needed.

Let the *path* of a component be $P = [p_0, p_1, p_2, \ldots, p_n]$, where $p_i \in P$ is the intermediate position after the $i$th control and $p_0$ is the initial position of the component, where a *control* is a single translation. Define the *control sequence* for the component as $C = [c_1, c_2, \ldots, c_n]$, where $c_i = p_i - p_{i-1}$ is the $i$th control.

A simple greedy approach to grouping components is as follows. Components may be *whole* (entire grid cells) or *partial*. We first deal only with whole components. Simulate motions of all of the whole components in each of the four cardinal directions. For each direction, count the number of components that reach the exit area, and may be grouped together based on 4-connectivity; greedily, choose the direction that gives the fewest such grouped pieces.

Add the resulting cleared squares as a target area, and attempt motion in each of the four cardinal translation directions, potentially creating new piece groups; attempt to then move these new groups to the exit.

Table II shows some statistics for the sufficient decomposition for some example shapes and a few illustrative run times. The main time cost is spent on testing collisions. Decomposition solutions are shown in Fig. 10.

## IV. CONCLUSION

We proposed the problem of separating a divisible polygon from an interlocked atomic polygon by cutting. We explored

lower and upper bounds on the number of pieces that the divisible part must be cut into and presented algorithms to make the cuts and achieve the separation using a sequence of translations. Both bounds are very conservative; the main contribution of this paper is the proposal of the problem and an initial exploration of solutions.

Improving the quality of bounds is of great interest for future work. In the section of computing the *necessary* number of pieces that the divisible part must be cut into, we only considered some sets of points immediately adjacent to the edges of the atomic part, ignoring global properties. For example, extremely small exit corridors through the atomic part should increase the lower bound. Such properties might be considered by how large a component might be before it "plugs" a hole in configuration space.

While computing a *sufficient* number of pieces, the algorithm first decomposes the divisible part into convex polygons and then computes a grid resolution using all pairs of adjacent polygons. The algorithm can generate a large number of pieces if there exists a single narrow corridor anywhere in the divisible part. Different convex decomposition methods might also yield better decompositions, as might the selection of good disassembly sequences [6], [19].

In future work, we would also like to expand the problem scope. We expect that most practical problems in this area are 3-D, rather than planar. Also, we can imagine situations where there are several atomic components and several divisible components. How should such multicomponent 3-D puzzles be designed, assembled, or disassembled?

## REFERENCES


[1] T. F. Allen, J. W. Burdick, and E. Rimon, "Two-finger caging of polygonal objects using contact space search," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1164–1179, Oct. 2015.

[2] D. J. Arbuckle and A. A. G. Requicha, "Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: Algorithms and simulations," *Auto. Robots*, vol. 28, no. 2, pp. 197–211, 2010.

[3] D. J. Balkcom and J. C. Trinkle, "Computing wrench cones for planar rigid body contact tasks," *Int. J. Robot. Res.*, vol. 21, no. 12, pp. 1053–1066, 2002.

[4] A. Becker, G. Habibi, J. Werfel, M. Rubenstein, and J. McLurkin, "Massive uniform manipulation: Controlling large populations of simple robots with a common input signal," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 520–527.

[5] M. P. Bell, W. Wang, J. Kunzika, and D. Balkcom, "Knot-tying with four-piece fixtures," *Int. J. Robot. Res.*, vol. 33, no. 11, pp. 1481–1489, 2014.

[6] L. Beyeler, J.-C. Bazin, and E. Whiting, "A graph-based approach for discovery of stable deconstruction sequences," in *Advances in Architectural Geometry*. Springer, 2015, pp. 145–157.

[7] J.-S. Cheong, H. J. Haverkort, and A. F. van der Stappen, "Computing all immobilizing grasps of a simple polygon with few contacts," *Algorithmica*, vol. 44, no. 2, pp. 117–136, 2006.

[8] J.-S. Cheong, A. F. van der Stappen, K. Goldberg, M. H. Overmars, and A. Rimon, "Immobilizing hinged polygons," *Int. J. Comput. Geometry Appl.*, vol. 17, no. 1, pp. 45–69, 2007.

[9] V. Chvátal, "A greedy heuristic for the set-covering problem," *Math. Oper. Res.*, vol. 4, no. 3, pp. 233–235, 1979.

[10] R. Connelly, E. D. Demaine, and G. Rote, "Straightening polygonal arcs and convexifying polygonal cycles," in *Proc. IEEE 41st Annu. Symp. Found. Comput. Sci.*, Nov. 2000, pp. 432–442.

[11] J. Czyzowicz, I. Stojmenovic, and J. Urrutia, "Immobilizing a polytope," in *Proc. Workshop Algorithms Data Struct.*, 1991, pp. 214–227.

[12] J. Czyzowicz, I. Stojmenovic, and J. Urrutia, "Immobilizing a shape," *Int. J. Comput. Geometry Appl.*, vol. 9, no. 2, pp. 181–206, 1999.

[13] J. Erickson, S. Thite, F. Rothganger, and J. Ponce, "Capturing a convex object with three discs," *IEEE Trans. Robot.*, vol. 23, no. 6, pp. 1133–1140, Dec. 2007.

[14] C.-W. Fu, P. Song, X. Yan, L. W. Yang, P. K. Jayaraman, and A. Cohen-Or, "Computational interlocking furniture assembly," *ACM Trans. Graph.*, vol. 34, no. 4, 2015, Art. no. 91.

[15] P. Herholz, W. Matusik, and M. Alexa, "Approximating free-form geometry with height fields for manufacturing," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 239–251, 2015.

[16] R. Hu, H. Li, H. Zhang, and D. Cohen-Or, "Approximate pyramidal shape decomposition," *ACM Trans. Graph.*, vol. 33, no. 6, p. 213, 2014.

[17] J. Huang, S. K. Gupta, and K. Stoppel, "Generating sacrificial multi-piece molds using accessibility driven spatial partitioning," *Comput.-Aided Des.*, vol. 35, no. 13, pp. 1147–1160, 2003.

[18] K. Kotay, D. Rus, M. Vona, and C. McGray, "The self-reconfiguring robotic molecule," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1998, pp. 376–386.

[19] A. Loomis, "Computation reuse in stacking and unstacking," Tech. Rep., 2005.

[20] S. Makita and Y. Maeda, "3D multifingered caging: Basic formulation and planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 2697–2702.

[21] M. T. Mason, *Mechanics of Robotic Manipulation*. Cambridge, MA, USA: MIT Press, 2001.

[22] B. Mishra, J. T. Schwartz, and M. Sharir, "On the existence and synthesis of multifinger positive grips," *Algorithmica*, vol. 2, nos. 1–4, pp. 541–558, 1987.

[23] A. K. Priyadarshi and S. K. Gupta, "Geometric algorithms for automated design of multi-piece permanent molds," *Comput.-Aided Des.*, vol. 36, no. 3, pp. 241–260, 2004.

[24] R. Ravi and M. N. Srinivasan, "Decision criteria for computer-aided parting surface design," *Comput. Aided Des.*, vol. 22, no. 1, pp. 11–17, Jan. 1990.

[25] F. Reuleaux, *Theoretische Kinematik: Grundzüge einer Theorie des Maschinenwesens*, vol. 1. Friedrich Vieweg und Sohn, 1875.

[26] E. Rimon and A. Blake, "Caging 2D bodies by 1-parameter two-fingered gripping systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2. Apr. 1996, pp. 1458–1464.

[27] E. Rimon and J. W. Burdick, "Mobility of bodies in contact. I. A 2nd-order mobility index for multiple-finger grasps," *IEEE Trans. Robot. Autom.*, vol. 14, no. 5, pp. 696–708, Oct. 1998.

[28] E. Rimon and J. W. Burdick, "Mobility of bodies in contact. II. How forces are generated by curvature effects," *IEEE Trans. Robot. Autom.*, vol. 14, no. 5, pp. 709–717, Oct. 1998.

[29] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal, "Collective transport of complex objects by simple robots: Theory and experiments," in *Proc. Int. Conf. Auto. Agents Multi-Agent Syst.*, 2013, pp. 47–54.

[30] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[31] D. Rus and M. Vona, "Crystalline robots: Self-reconfiguration with compressible unit modules," *Auto. Robots*, vol. 10, no. 1, pp. 107–124, 2001.

[32] J. Snoeyink and J. Stolfi, "Objects that cannot be taken apart with two hands," in *Proc. ACM 9th Annu. Symp. Comput. Geometry*, 1993, pp. 247–256.

[33] P. Song *et al.*, "CofiFab: Coarse-to-fine fabrication of large 3D objects," *ACM Trans. Graph.*, vol. 35, no. 4, p. 45, 2016.

[34] P. Song, Z. Fu, L. Liu, and C.-W. Fu, "Printing 3D objects with interlocking parts," *Comput. Aided Geometric Des*, vols. 35–36, pp. 137–148, May 2015.

[35] J. Steiner, "Einige Gesetze über die Theilung der Ebene und des Raumes," *J. Reine Angewandte Math.*, vol. 1, pp. 349–364, 1826.

[36] I. Streinu, "A combinatorial approach to planar non-colliding robot arm motion planning," in *Proc. 41st Annu. Symp. Found. Comput. Sci.*, Nov. 2000, pp. 443–453.

[37] C. D. Toth, J. O'Rourke, and J. E. Goodman, *Handbook of Discrete and Computational Geometry*. Boca Raton, FL, USA: CRC Press, 2004.

[38] M. Vahedi and A. F. van der Stappen, "Caging polygons with two and three fingers," *Int. J. Robot. Res.*, vol. 27, nos. 11–12, pp. 1308–1324, 2008.

[39] W. Wang and D. Balkcom, "Grasping and folding knots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 3647–3654.

[40] Y. Zhang and D. Balkcom, "Interlocking structure assembly with voxels," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 2173–2180.

[41] Y. Zhang, X. Chen, H. Qi, and D. Balkcom, "Rearranging agents in a small space using global controls," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3576–3582.

[42] Y. Zhang, E. Whiting, and D. Balkcom, "Assembling and disassembling planar structures with divisible and atomic components," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2016, pp. 1–15.

**Yinan Zhang**, photograph and biography not available at the time of publication.

**Emily Whiting**, photograph and biography not available at the time of publication.

**Devin Balkcom**, photograph and biography not available at the time of publication.