Snipperclips: Cutting Tools into Desired Polygons using Themselves

Erik D. Demaine* Matias Korman† André van Renssen‡, § Marcel Roeloffzen‡, §

Abstract

We study *Snipperclips*, a computer puzzle game whose objective is to create a target shape with two tools. The tools start as constant-complexity shapes, and each tool can snip (i.e., subtract its current shape from) the other tool. We study the computational problem of, given a target shape represented by a polygonal domain of n vertices, is it possible to create it as one of the tools' shape via a sequence of snip operations? If so, how many snip operations are required? We show that a polynomial number of snips suffice for two different variants of the problem.

1 Introduction

Snipperclips: Cut It Out, Together! [8] is a puzzle game developed by SFB Games and published by Nintendo worldwide on March 3, 2017 for their new console, Nintendo Switch. In the game, up to four players cooperate to solve puzzles. Each player controls a character[1] whose shape starts as a rectangle in which two corners have been rounded so that one short side becomes a semicircle. The main mechanic of the game is *snipping*: when two such characters partially overlap, one character can *snip* the other character, i.e., subtract the current shape of the first character from the current shape of the latter character; see Figure 1. In addition, a *reset* operation allows a character to restore its original shape. An unreleased 2015 version of this game, *Friendshapes* by SFB Games, had the same mechanics, but supported only up to two players [4].

Puzzles in Snipperclips have varying goals, but an omnipresent subgoal is to form one or more players into desired shape(s), so that they can carry out required actions. In particular, a core puzzle type ("Shape Match") has one target shape which must be (approximately) formed by the union of the character's shapes. When the target shape has one connected component per character, the puzzle is equivalent to the characters reaching a desired set of target shapes, one per character. In this paper, we study when this goal is attainable, and when it is, analyze the minimum number of operations required.

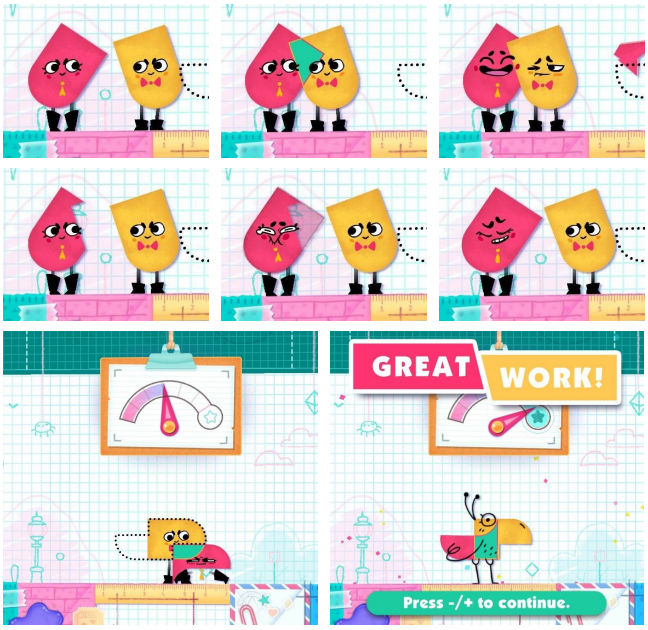

Figure 1: Cropped screenshots of Snipperclips: snipping, resetting, and solving a Shape Match puzzle. Sprites copyright SFB/Nintendo and included here under Fair Use.

2 Problems and Results

For the remainder of the paper we consider the case of exactly two characters or *tools* $\mathcal{T}_1$ and $\mathcal{T}_2$. For geometric simplicity, we assume that the initial shape of both tools is a unit square. Most of the results in this paper work for nice (in particular, fat) constant-complexity initial shapes, such as the rounded rectangle in Snipperclips, but would result in a more involved description.

We view each tool as an open set of points that can be rotated and translated freely.[2] After any rigid transformation, if the two tools have nonempty intersection,

*Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, edemaine@mit.edu

†Tohoku University, Sendai, Japan, mati@dais.is.tohoku.ac.jp. Partially supported by MEXT KAKENHI Nos. 12H00855, and 17K12635.

‡National Institute of Informatics, Tokyo, Japan, {andre, marcel}@nii.ac.jp

§JST, ERATO, Kawarabayashi Large Graph Project. Supported by JST ERATO Grant Number JPMJER1305, Japan

[1]The game in fact allows one human to control up to two characters, with a button to switch between which character is being controlled.

[2]In the actual game, the tools' translations are limited by gravity, jumping, crouching, stretching, standing on each other, etc.,

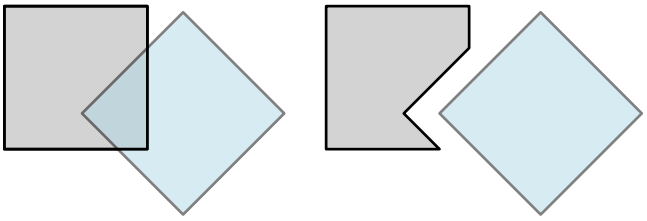

Figure 2: By translating and rotating the two tools we can make them partially overlap (left figure). In the right we see the resulting shape of both tools after the snip operation.

we can *snip* (or *cut*) one of them, i.e., remove from one of the tools the closure of the intersection of the two tools (or equivalently, the closure of the other tool); see Figure 2. (The closure is used to preserve the invariant that both tools remain open sets.) In addition to the snip operation, we can *reset* a tool, which returns it back to its original unit-square shape.

Although we forbid it in our study, the actual game has an additional *undo/redo* operation, allowing each tool to change into its previous shape (before its last snip or undo/redo operation, but not before its last reset operation), effectively implementing an undo stack of size 1. Our positive results are stronger without needing this operation; our negative results (Section 3) are weaker without allowing this operation, and may not hold in the stronger undo/redo model.

After a snip operation, the changed tool could become disconnected. There are two natural variants on the problem of how to deal with disconnection. In the *connected model*, we force each tool to be a single connected component. Thus, if the snip operation disconnects a tool, we can choose which component to keep. In the *disconnected model*, we allow the tool to become disconnected, viewing a tool as a set of points to which we apply rigid transformations and the snip/reset operation. The Snipperclips game by Nintendo follows the disconnected model, but we find the connected model an interesting alternative to consider.

Ideally, given two target shapes P_1 and P_2, we would like to find a sequence of snip/reset operations that transform tool $\mathcal{T}_1$ into P_1 and at the same time transform $\mathcal{T}_2$ into P_2. However, as we show in Observation 1, this is not always possible, even when $P_1 = P_2$. Instead, we consider creating a single target shape P_1 by one of the tools $\mathcal{T}_1$. Because our initial shape is polygonal, and we allow only finitely many snips, the target shape must be a polygonal domain, say of n vertices. The primary goal of this paper is to design an algorithm that, for any target shape P_1, can transform tool $\mathcal{T}_1$ into the desired shape P_1 using as few snip and reset operations as possible. Specifically, our aim is for the number of snip and reset operations to depend only on n (and not depend on other parameters such as the feature size of the target shape).

though in practice this is not a huge limitation. Rotation is indeed arbitrary.

2.1 Results

For negative results (Section 3), we show a pair of shapes that cannot be simultaneously realized by both tools. We also provide a shape that requires $\Omega(n)$ snips when we aim to construct it in a single tool (in both the connected and disconnected models). For positive results, we give constructive algorithms to create any target shape in the connected model using $O(n)$ snips (Section 4) and in the disconnected model using $O(n^2)$ snips (Section 5).

2.2 Related Work

Computational geometry has considered a variety of problems related to cutting out a desired shape using a tool such as circular saw [3], hot wire [5], and glass cutting [6, 7]. The Snipperclips model is unusual in that the tools are themselves the material manipulated by the tools. This type of model arises in real-world manufacturing, for example, when using physical objects to guide the cutting/stamping of other objects—a feature supported by the popular new *Glowforge* laser cutter [1] via a camera system.

Our problem can also be seen as finding the optimal Constructive Solid Geometry (CSG) expression tree, where leaves represent base shapes (in our model, rectangles), internal nodes represent shape subtraction, and the root should evaluate to the target shape, such that the tree can be evaluated using only two registers. Applegate et al. [2] studied a rectilinear version of this problem (with union and subtraction, and a different register limitation).

3 Lower Bounds

We begin with the intuitive observation that not all combinations of target shapes can be constructed.

Observation 1 *In both the connected and disconnected models (without undo/redo), there is a target shape that cannot be realized by both tools at the same time.*

Proof. Consider the target shape shown in Figure 3: a unit square in which we have removed a very thin hole in the middle. First observe that, if we perform no resets, neither tool has space to spare to construct a thin auxiliary needle to carve out the middle section of the other tool. Thus, after we have completed carving one tool, the other one would need to reset. This implies that we cannot have the target shape in both tools at the same time.

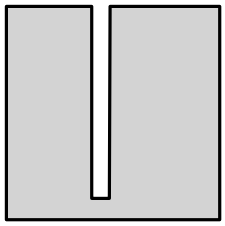

Figure 3: A target shape that cannot be realized by both tools at the same time.

Now assume that we can transform both tools into the target shape by performing a sequence of snips and resets. Consider the state of the tools just after the last reset operation. One of the two shapes is the unit square and thus we still need to remove the thin hole using the other shape. However, because no more resets are executed, the other tool is currently and must remain a superset of the target shape. In particular, it can differ from the square only in the thin hole, so it does not have any thin portions that can carve out the hole of the other tool.

Because the above argument is based solely on the shape of the figure, it holds in both the connected and disconnected model. □

Next we show that constructing a target shape in only one of the two tools may require a linear number of operations in both the connected and disconnected model.

Theorem 2 *There are target shapes that require $\Theta(n)$ snips to construct, both in the connected and disconnected model (without undo/redo).*

Proof. Consider the target shape P_1 to be a set of $n/3$ triangles on a line such that the distance between two consecutive triangles grows exponentially. In the connected model, we add a strip to connect these triangles; see Figure 4. We complete the construction by scaling it so that the width and height of P_1 is unit (and thus fits in either tool).

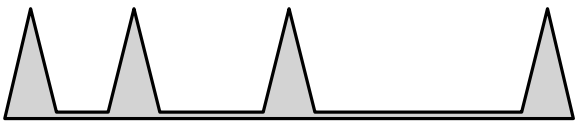

Figure 4: The target shape used in the lower bound.

First observe that it is straightforward to construct P_1 into one of the two tools by making the other tool a rectangle with width smaller to or equal to the shortest segment of P_1. Thus, we now proceed to prove the lower bound.

Consider now a sequence of snip and reset operations that can be used to construct P_1 in one of the tools. If we only have tools that cut out a constant number of edges or we have a linear number of tools, the lower bound follows, so we focus on tools that cut out multiple edges from different triangles. We observe that due to the spacing between the triangles, these tools cannot be reused to cut out multiple edges from any other triangles. Hence, if we can show that the number of snips required to construct these tools is linear in their complexity, we are done.

Let us consider such a tool that cuts out multiple edges from different triangles. Note that this tool has strictly fewer vertices than P_1. Using similar arguments as above, we can assume that each such tool was not constructed using only tools that cut out a constant number of edges nor using a linear number of tools. If we continue this recursion, each time we recurse into strictly smaller tools. Hence, in the final step, we end up with tools that are constructed using tools that cut out a constant number of edges or we have a linear number of tools in total. Regardless, the lower bound follows. □

4 Connected Model

In the connected model, the shapes must remain connected and we enforce this by choosing a connected component whenever a snip breaks the shape into multiple pieces. In this model, we show that $O(n)$ snips suffice to create any polygonal shape of n vertices.

Theorem 3 *We can cut one of the tools into any target polygonal domain P_1 of n vertices using $O(n)$ snip operations (and no reset) in the connected model.*

Proof. The idea is that we can shape $\mathcal{T}_2$ into a very narrow triangle, a *needle*, and use that to cut along the edges of the target shape P_1. Whenever a snip disconnects the shape, we simply keep the one containing the target shape. Initially, we start with a long needle to cut the long edges of $\mathcal{T}_2$ and we gradually shrink the needle to cut the smaller edges.

More formally, let α be the smallest angle between any two adjacent edges of P_1. As it will be seen later, we need α to be small. Thus, if $\alpha > 1°$, we simply lower it to $1°$ instead. Furthermore, let h be the shortest distance between any vertex and a non-adjacent edge. Our needle will be an isosceles triangle, with the two equal-length edges making an angle of at most α and the base edge with length equal to h (if this would cause the equal length segments to have length greater than one we reduce them to unit length, see Figure 5).

Now we group all edges of P_1 into sets based on their length. Let $\mathcal{E}$ denote the full set of edges defining P_1 and let $\mathcal{E}_i$, for $0 \leq i$, be the set of edges whose length is between 2^{-i-1} and 2^{-i}. To cut along the edges of $\mathcal{E}_i$, we use a needle where the equal-length edges have length 2^{-i-2}. Such a needle can construct each edge in $\mathcal{E}_i$ using at most four snips; see Figure 5. For an edge e, its

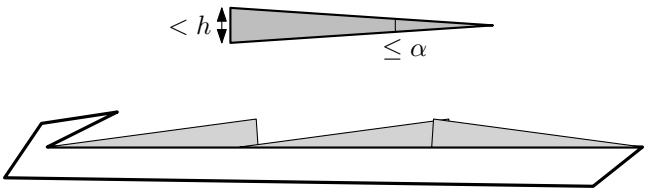

Figure 5: The needle is an equilateral triangle with apex at most α and a base edge of length at most h. The edges of equal length have length at most 1 so that the whole triangle can fit inside a tool.

nearest other features of P_1 are its two adjacent edges, the vertices closest to the edge, and the edges closest to its endpoints. We avoid cutting into the adjacent edges by placing the tip of the needle at the vertex when cutting near a vertex and we cannot cut out non-adjacent vertices and edges, because the base of the needle is at most h.

By making the cuts along the edges in the sets $\mathcal{E}_i$ in increasing order of i the needle has to only shrink, which is easily done by cutting it with any outer edge of the current polygon (all are guaranteed to be at least length h). Making the initial needle requires two snips, cutting each edge requires at most four snips and hence $O(n)$ snips in total, and reducing the needle length requires one snip per nonempty set $\mathcal{E}_i$ of which there are at most $O(n)$. Thus, in total the required number of snips is $O(n)$. □

5 Disconnected Model

Recall that in the disconnected model, we allow the tool to become disconnected, i.e., when a snip disconnects the tool, we keep both components.

In order to carve out a target shape P_1, we virtually fix a location of P_1 inside $\mathcal{T}_1$, pick a corner c of $\mathcal{T}_1$ (say, the lower right one) and consider the set of distances $d_1, \ldots, d_{n'}$ from each of the vertices in the fixed location of the target shape P_1 to c in decreasing order under the L_∞-metric. For simplicity assume that all distances are distinct, and thus $n' = n$ (this can be achieved with symbolic perturbation). We refer to the part of $\mathcal{T}_1$ not in P_1, i.e., $\mathcal{T}_1 \setminus P_1$, as the *free-space*. We will remove the free-space in n steps, where in each step i we remove the free-space from an L-shaped region Q_i that is the intersection of $\mathcal{T}_1$ and an annulus formed by removing the L_∞-ball of radius d_i from the L_∞-ball of radius d_{i-1}. We argue that in each step we will need $O(n)$ snips and resets, thus creating the target shape in $O(n^2)$ operations.

Lemma 4 *The free-space in region Q_i can be removed in $O(n)$ snips and reset operations provided that $\bigcup_{j \leq i} Q_j$ is a square in $\mathcal{T}_1$.*

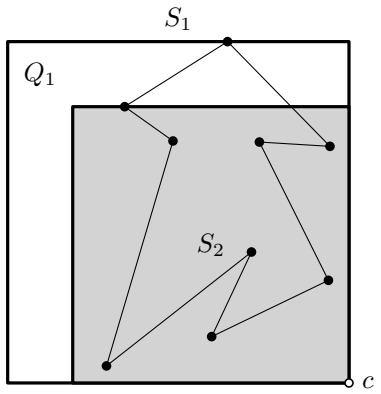

Figure 6: The squares S_1 and S_2 along with L-shaped region Q_1 and corner c.

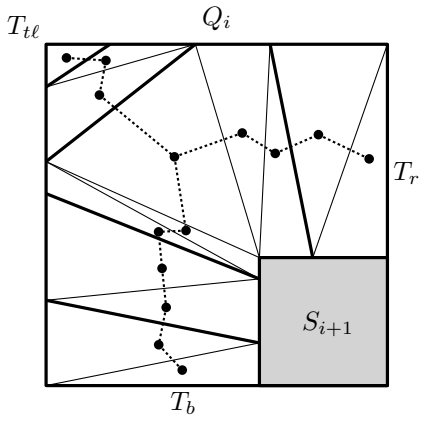

Figure 7: An L-shaped region Q_i, the edges of the target shape that cross it (thick edges) define F_i. We further triangulate each face (thin edges), and consider the corresponding dual graph (dotted edges).

Proof. Let S_i be the bounding square containing Q_i (see Figure 6) and let F_i be the set of faces created when removing the boundary edges of the target shape from Q_i. By definition all vertices of the target shape on Q_i must be on its inner or outer L-shaped boundary and all boundary segments must fully traverse Q_i, i.e., they cannot have an endpoint inside Q_i. It then follows that the set F_i of faces consists of $O(n)$ constant complexity pieces. Now triangulate all faces of F_i and let T_i denote the resulting set of triangles (Figure 7). Note that our aim is to remove some of the triangles of T_i. We will show that we can remove any triangle that fits in $S_i \setminus S_{i+1}$ with a constant number of cuts.

For simplicity in the exposition we first consider the case in which $\boldsymbol{S_{i+1}}$ **is large.** That is, the side length of S_{i+1} is at least half the side length of S_i. Consider a triangle $T \in F_i$ that needs to be removed. To create a cutting tool move $\mathcal{T}_2$ so that its only overlap with $\mathcal{T}_1$ is S_i. Let S_i' denote the area in $\mathcal{T}_2$ corresponding to S_i and let T' be the projection of T on $\mathcal{T}_2$. Our goal

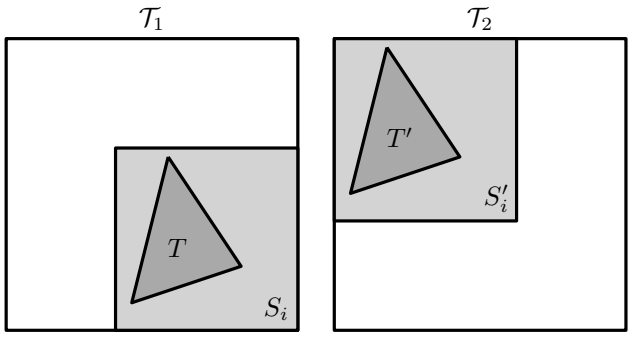

Figure 8: A triangle T in S_i is cut out of $\mathcal{T}_2$ at T'.

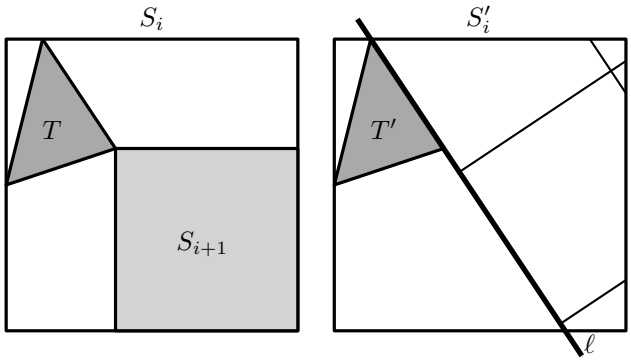

Figure 9: If S_{i+1} is large, we can use it to carve out any desired shape in $\mathcal{T}_2$ with $O(1)$ snips.

will be to remove $S'_i \backslash T'$ from $\mathcal{T}_2$ without affecting T'. Note that we can create a cut where only S'_i overlaps $\mathcal{T}_1$ in S_i, so the shape of $T_2 \backslash S'_i$ does not influence the cut (Figure 8). That means we do not have to cut it away and we do not need to worry about cutting part of it while creating a cutting tool within S'_i.

Consider the halfspace H defined by one of the bounding lines ℓ of T' that does not contain T'. We can remove $H \cap S'_i$ by rotating $\mathcal{T}_1$ so that one of the sides of $\mathcal{T}_1$ along which S_{i+1} is situated aligns with ℓ and repeatedly snip with S_{i+1} in a grid-pattern as shown in Figure 9. Because S_{i+1} is large compared to S'_i we can remove $H \cap S'_i$ in $O(1)$ snips. We then apply the same procedure for the other two halfspaces that should be removed to obtain the cutting tool for T. This means that, under the assumption that S_{i+1} is large, each triangle can be removed in $O(1)$ snips. Since there are $O(n)$ triangles in S_i, the linear bound holds.

It remains to consider the case in which $\boldsymbol{S_{i+1}}$ **is small.** (that is, the side length of S_{i+1} is less than half that of S_i, and potentially much smaller). Although the main idea is the same, we need to remove the triangles in order, and use portions of Q_i that are still solid to create the cutting tools.

Let G_i be the dual graph of T_i. This graph is a tree with at most three leaves. Two leaves correspond to the unique triangles T_b and T_r that share an edge with the lower and right boundary of Q_i respectively and the third exists only if the top-left corner of Q_i is contained in a single triangle $T_{t\ell}$, that is, there is at least one segment contained in Q_i that connects the top and left boundaries; see Figure 7. Finally, we change the coordinate system so that c is the origin, and S_i is a unit square (note that the vertices of this square are $(-1, 1)$, $(-1, 0)$, $(0, 1)$, and $c = (0, 0)$).

We process the triangles in the following order. We first process the *cross-triangles*, triangles with one endpoint on the left boundary and one on the top boundary, (if any exist) starting from $T_{t\ell}$ following G_i until we find a triangle that has degree three in G_i which we do not process yet. The remaining *fan-triangles* form a path in G_i which we process from T_b to T_r.

Cross-triangles. Recall that, by the way in which we nest regions Q_i, there cannot be vertices to the right or below S_i. In particular, cross-triangles have all three vertices in the top and left boundaries of Q_i. Hence, while we have some cross-triangle that has not been processed, the triangle of vertices $(-1, 0)$, $(0, 1)$ and c must be present in $\mathcal{T}_1$. This triangle has half the area of Q_i and can be used to create cutting pieces in the same way as when S_{i+1} is large. Thus, we conclude that any cross-triangle of Q_i can be removed from $\mathcal{T}_1$ with $O(1)$ cuts.

Fan-triangles. We now process the fan-triangles in the path from T_b to T_r in G_i. We treat this sequence in two phases. First consider the triangles that have at least one vertex on the left edge of S_i (that is, we process triangles up to and including the triangle that has degree three in G_i if it exists); out of these triangles, only the triangle of degree three can intersect the triangle of vertices $(0, 1)$, $(0, 3/4)$, and $(-3/4, 3/4)$. This triangle has $1/32$ of the total area of S_i, and as before we can use it as cutting tool to create any desired triangle with $O(1)$ snips.

The remaining triangles have their vertices in the upper edge of S_i and on the upper or left edge of S_{i+1}. In this case we must be more careful as we cannot guarantee the existence of a large square in $\mathcal{T}_1$. However, we do not have to clear the entire space S'_i any longer. Instead it suffices to clear a much smaller area.

Let T denote the next triangle to be removed and let B denote the bounding box of T and c (see Figure 10). As before consider moving $\mathcal{T}_2$ so that the only overlap with $\mathcal{T}_1$ is B, let B' denote this area in $\mathcal{T}_2$ and T' the projection of T onto B'. To create a cutting tool we need only remove the area $B' \backslash T'$.

As before, we look for a region in $\mathcal{T}_1$ that has roughly the area of T to use for carving the desired shape in $\mathcal{T}_2$. Let w be the width of B. Also, let h' the height of S_{i+1}. Note that the height of B is 1, and since S_{i+1} is small, we have $h' < 1/2$. By construction of the bounding box,

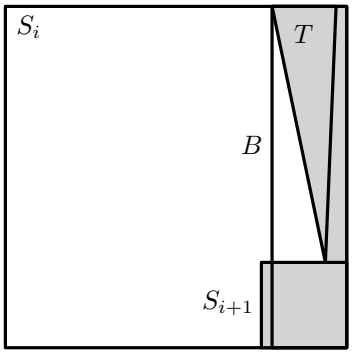

Figure 10: The solid areas (grey) and bounding box B when cutting fan-triangles with no vertices on the left boundary of S_i.

one of the vertices of T will have x coordinate equal to $-w$; let q denote this vertex. The y coordinate y_q of q is either 1 or h' as it must be on the upper edge of S_i or on the upper boundary of S_{i+1}—if T has vertices on left boundary of S_{i+1}, then there is a vertex on the upper boundary of S_i with lower x coordinate. Now consider with vertices $(0,1),(0,h'),q$. This triangle has height at least $1-h' > 1/2$ and width w, and thus its area is at least $1/4$ of the area of B. As in the previous cases, we use this triangle to create a cutting tool from $\mathcal{T}_2$ to remove triangle T from $\mathcal{T}_1$.

Thus, it follows that all free-space triangles can be removed with a cutting tool that is constructed from $\mathcal{T}_2$ in $O(1)$ snips and reset operations, hence we can clear Q_i of free-space in total $O(n)$ operations. □

Because there are at most n distinct distances, we repeat this procedure at most n times, giving us the desired result.

Theorem 5 *We can cut one of the tools into any target polygonal domain P_1 of n vertices using $O(n^2)$ snips and reset operations in the disconnected model.*

6 Open Problems

For cutting one tool into a desired polygonal shape, our results are tight in the connected model ($\Theta(n)$), but the disconnected model (as implemented by the Snipperclips game) still has a gap between $\Omega(n)$ and $O(n^2)$. What is the optimal worst-case number of cuts as a function of n? What about the algorithmic question of cutting out a given shape with the fewest possible cuts for that shape (instead of the worst case)? Is this problem NP-hard, and does it have a constant-factor approximation algorithm?

For cutting two tools (or more tools) simultaneously into desired polygonal shapes, the main open problem is to characterize when this is possible. Is the decision problem NP-hard? How does the problem change if we allow the undo/redo operation described in Section 2?

It would also be interesting to consider the initial shape implemented in the Snipperclips game (instead of the unit squares we used for simplicity), namely, a unit square adjoined with half a unit-diameter disk. This initial shape opens up the possibility of making curved target shapes bounded by line segments and circular arcs of matching curvature. Can all such shapes be made, and if so, by how many cuts?

Acknowledgments

This work was initiated at the 32nd Bellairs Winter Workshop on Computational Geometry held January 2017 in Holetown, Barbados. We thank the other participants of that workshop for providing a fun and stimulating research environment. E. Demaine also thanks Hugo Akitaya, Martin Demaine, Adam Hesterberg, Jason Ku, and Jayson Lynch for helpful discussions about (and games of) Snipperclips. We also thank Hugo Akitaya for taking the screenshots in Figure 1.

References

[1] Glowforge — the 3D laser printer. https://glowforge.com/, 2015.

[2] D. A. Applegate, G. Calinescu, D. S. Johnson, H. Karloff, K. Ligett, and J. Wang. Compressing rectilinear pictures and minimizing access control lists. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1066–1075, New Orleans, Louisiana, 2007.

[3] E. D. Demaine, M. L. Demaine, and C. S. Kaplan. Polygons cuttable by a circular saw. *Computational Geometry: Theory and Applications*, 20(1–2):69–84, October 2001. CCCG 2000.

[4] GDC. European innovative games showcase: Friendshapes. YouTube video, 2015. https://youtu.be/WJGooKIoy1Q.

[5] J. W. Jaromczyk and M. aw Kowaluk. The face-wise continuity in hot wire cutting of polyhedral sets. In *Proceedings of the 16th European Workshop on Computational Geometry*, pages 93–97, Eilat, Israel, March 2000.

[6] M. H. Overmars and E. Welzl. The complexity of cutting paper. In *Proceedings of the 1st Annual ACM Symposium on Computational Geometry*, pages 316–321, Baltimore, Maryland, June 1985.

[7] J. Pach and G. Tardos. Cutting glass. *Discrete & Computational Geometry*, 24:481–495, 2000.

[8] Wikipedia. Snipperclips. https://en.wikipedia.org/wiki/Snipperclips, 2017.