2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

On Designing 2D Discrete Workspaces to Sort or Classify Polyominoes

Phillip Keldenich[1], Sheryl Manzoor[2], Li Huang[2], Dominik Krupke[1], Arne Schmidt[1], Sándor P. Fekete[1], and Aaron T. Becker[2]

***Abstract*— This paper studies the general problem of physically sorting polyominoes according to shape using a 2D, rigid, grid-based workspace. The workspace is designed for sensorless operation, using a fixed set of open-loop force-field inputs that move a polyomino from an inlet port to an outlet port that corresponds to the polyomino's shape, and reset the workspace to classify the next polyomino. This paper proves that static workspaces can classify all orthoconvex polyominoes of width w and height h, and provides a motion sequence and required size of workspace as a function of w and h. By allowing moving polyomino cams that assist in the sorting, we can design dynamic workspaces that can sort all polyominoes that are "completely filled" using a constant number of force-field inputs. Hardware experiments using magnetic and gravity-based actuation demonstrate these static and dynamic sensorless classifiers at the millimeter scale.**

I. INTRODUCTION

While macro-scale assembly typically involves precision manipulators and many actuators, assembly at small scales often relies on self-assembly and the influence of global external conditions, such as the temperature of a vessel, the addition of a catalyst, or turning on a magnetic field.

Inspired by this paradigm, we have investigated techniques that generate multiple copies of desired polyominoes using a series of actuations that move every tile in the workspace in the same direction until halted by an obstacle. A *polyomino* is a 2D structure composed of square tiles joined along edges. Recent experimental work by Manzoor et al. [1] demonstrated this actuation with 300 μm alginate particles, using external magnetic fields to sequentially attach particles to an existing subassembly. Becker et al. [2] showed that the decision problem of whether a simple polyomino can be built or not is solvable in polynomial time. However, errors can occur during the assembly process. Because the assembly sequence is performed in open-loop, these errors propagate, even to the point of plugging the workspace and disabling further construction.

To address this challenge this paper studies the general problem of physically sorting a polyomino according to shape using open-loop actuation, as illustrated in Fig. 1.

II. RELATED WORK

Error detection and shape recognition is a fundamental need at many size scales in biology, from error detection in DNA strands to how antibodies bind to a specific shape of antigen. Many research efforts have been dedicated to developing shape detection and classification methods for cells since these methods have an important role in improving disease diagnosis processes and in devising better treatment strategies. There are many challenges in addressing fatal health conditions, e.g. sickle cell disease, because the malfunctioning cells have complex and heterogeneous shapes. They overlap and have indistinguishable structural features in the diagnosis images. In [3], the authors presented a convolutional neural network based approach to classify red blood cells in sickle cell anemia patients.

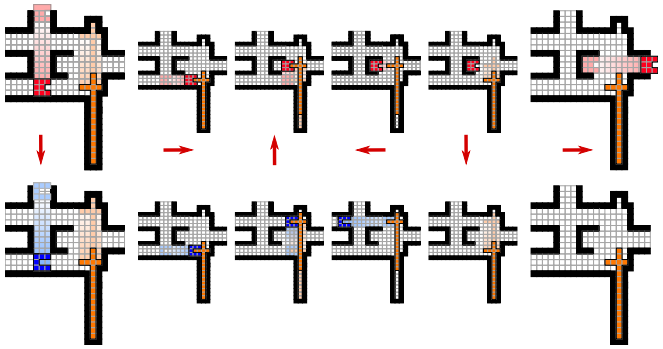

Fig. 1. A 2D workspace with a moving cam that sorts polyominoes based on shape in six moves. A 3×2 polyomino missing the rightmost tile from the middle row exits at the right (red), while a 3×2 polyomino missing the rightmost and middle tile from the middle row exits at the top left (red). See video attachment or https://youtu.be/ZeBur5F7sIo.

Similar skills are used for industrial processes that sort and grade grains and mineral particles. While these traditionally use sieves of various sizes and blowing air, machine vision is being increasingly used. For grading agricultural products, the drawbacks of manual systems include time consumption, variable labor availability, and inconsistency in grade judgement. To overcome these drawbacks, there exist many research works which aim at developing accurate and high speed, automatic sorting and grading systems. Most of these systems rely on machine vision, e.g. [4].

In contrast to systems requiring machine vision, this paper uses an open-loop process that requires no sensor feedback to sort shapes, and thus may be suitable for tiny enclosed environments. Sensorless manipulation has a rich history in robotics. Peshkin introduced a framework for designing stationary fences along a conveyor belt to align objects [5]. Goemans et al. [6] extended this to filter 3D parts by shape and orientation. Early work by Akella et al. demonstrated

[1]Department of Computer Science, TU Braunschweig, Germany. {p.keldenich, d.krupke, arne.schmidt, s.fekete}@tu-bs.de

[2]Department of Electrical and Computer Engineering, University of Houston, USA. {smanzoor2, lhuang28, atbecker}@uh.edu. Work from these authors was partially supported by National Science Foundation IIS-1553063 and IIS-1619278.

using a single-joint robot arm mounted above a moving conveyor belt to position and orient planar parts [7]. Recent work by Zhang et al. [8] uses the same model of global controls and grid-based obstacles as this paper, and shows there exists a workspace a constant factor larger than the number of agents that enables efficient, arbitrary rearrangement for a rectangle of agents.

Another way to filter, sort, or orient parts without sensors is by using vibrating surfaces. Böhringer et al. [9] presented efficient algorithms that compute sequences of force fields that position and orient parts in a predictable way. Beretty et al. [10] and Agarwal et al. [11] provided efficient algorithms to design traps that can be used to orient and sort parts in vibratory bowl feeders. This differs from our work, which focuses on accurately differentiating the detailed aspects of the *shape* of objects, rather than their orientation.

III. MODEL

This paper analyzes two problems: *sorting* and *error detection*. In both problems, we are constructing a workspace that is represented as polyomino with holes. The exterior of this workspace consists of rigid (immovable) obstacles. The interior of a workspace contains one or more mobile polyominoes that can be moved in one of the directions $d \in \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ using global controls. A control d concurrently moves all mobile objects in the specified direction until they become *blocked*. A mobile object is blocked if it is adjacent to a rigid obstacle or another blocked polyomino in the direction of motion. Friction does not influence the behavior of our mobile objects and the objects do not change their orientation.

In the sorting problem, we are given a family $\mathcal{F}$ of input polyominoes; we know in advance that only polyominoes from this family need to be considered. The goal is to compute a workspace W and a global control sequence $\sigma \in \{\uparrow, \downarrow, \leftarrow, \rightarrow\}^*$ that distinguishes the objects of $\mathcal{F}$ from each other in the following sense. The workspace W must contain a designated input region where a polyomino from $\mathcal{F}$ enters the workspace, and one output region for each polyomino $P \in \mathcal{F}$. These regions must be pairwise non-intersecting. Applying σ to the workspace must move any polyomino $P \in \mathcal{F}$ from the input region to its corresponding output region without entering any other output region in the process.

In the error detection problem, we are given a polyomino P. The goal is to compute a workspace W and a global control sequence σ that determines whether the input polyomino is correct, i.e., equal to P, or incorrect. We assume that incorrect polyominoes are not wider or higher than P; filtering polyominoes by height and width is straightforward. Similar to the situation for the sorting problem, the workspace W must contain a designated input region where a polyomino is placed, as well as accepting and rejecting output regions for correct and incorrect polyominoes. If P is placed in the input region, it must be moved to an accepting output region; other polyominoes must be moved to a rejecting output region. A general limitation of our approach is that we cannot detect holes in polyominoes; therefore, we do not consider errors where an object that should be solid has a hole. Depending on the situation, we want to optimize the constructed workspace according to the criteria *sorting speed*, i.e., length of the sorting sequence, and *workspace size*, i.e., the dimensions of the workspace.

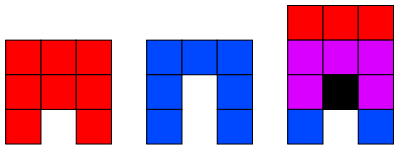

Fig. 2. Two polyominoes with a (vertical) dent of depth 1 (left) and 2 (middle). Static workspaces cannot distinguish these polyominoes. Introducing a difference between the top-left corners of the polyominoes can only be done by moving the polyominoes downwards onto a rigid obstacle (right). However, the only move that can be done after such a downward move is an upward move, leaving both polyominoes in the same position.

IV. STATIC WORKSPACES

In this section, we consider *static workspaces* that consist only of rigid obstacles. The only mobile object in a static workspace is the input polyomino that is currently being sorted. On the positive side, static workspaces are relatively simple and robust. However, there are limits to what kind of polyominoes can be sorted using static workspaces. For instance, it is impossible to measure the depth of a *dent*; see Fig. 2.

Definition 1. *A* dent *of depth d in a polyomino P is a column or row of $d > 0$ consecutive tiles not belonging to P, where the first tile is adjacent to exactly three tiles of P and all remaining tiles are adjacent to exactly two tiles of P.*

An important family of polyominoes that do not have dents are orthoconvex polyominoes. For these polyominoes, we can show that static workspaces suffice for sorting and error detection.

Theorem 1. *Families $\mathcal{F}$ of orthoconvex polyominoes of width w and height h can be sorted with a sorting sequence of length $\mathcal{O}(\min(|\mathcal{F}|, w + h))$ and a static workspace with dimensions $\mathcal{O}(|\mathcal{F}| \cdot wh) \times \mathcal{O}(|\mathcal{F}| \cdot wh)$. For orthoconvex polyominoes, error detection can be done with a static workspace with dimensions $\mathcal{O}(w(w + h)) \times \mathcal{O}(h(w + h))$ and sequences of length $\mathcal{O}(w + h)$.*

Proof. The proof is based on the following ideas. We can subdivide the boundary of any orthoconvex polyomino into four monotonic and four constant pieces as depicted in Fig. 3. We can use a gadget such as depicted in Fig. 4 to classify a polyomino based on the positions of the transition between monotonic and constant pieces. To classify polyominoes for which these positions are identical, we can test each row and column of the monotonic pieces individually; this can be done in constantly many steps per row and column. For error detection, we must also check that the polyomino is actually orthoconvex; this can be done by checking each individual row and column in total time $\mathcal{O}(w+h)$ and space $\mathcal{O}(w(w+h)) \times \mathcal{O}(h(w + h))$. □

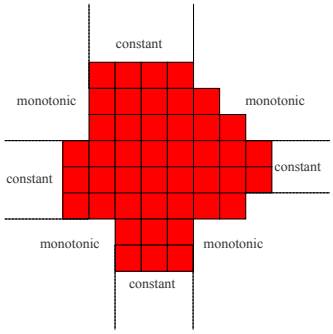

Fig. 3. Decomposition of the boundary of an orthoconvex polyomino.

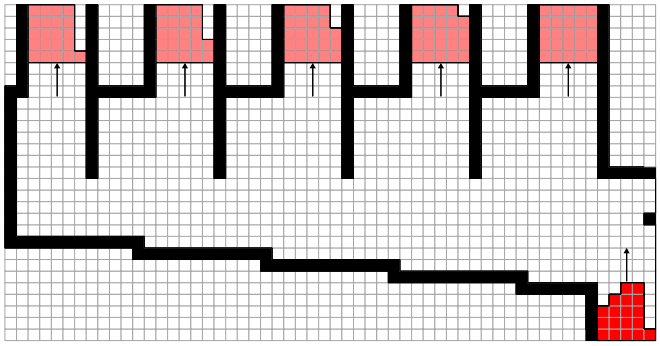

Fig. 4. Applying the control sequence ↑←↑→↑ classifies a polyomino based on the row of the top end of the rightmost constant piece of the boundary.

Moreover we can show that we cannot hope to sort orthoconvex polyominoes with fewer than $\Omega(\min(w+h, |\mathcal{F}|))$ moves in static workspaces. In other words, static workspaces cannot sort orthoconvex polyominoes in sublinear time. Thus, the sequence length required by the technique described in Theorem 1 is asymptotically optimal in the worst case.

Theorem 2. *For every $n \in \mathbb{N}$, there is a family $\mathcal{G}_n$ of n orthoconvex polyominoes of size $\mathcal{O}(n) \times \mathcal{O}(n)$ for which sorting requires $\Omega(n)$ moves in any static workspace.*

Proof. The family $\mathcal{G}_n$ can be constructed as follows; see Fig. 5 for an example. Starting with a staircase polyomino of width $n+2$, each family member $G_i \in \mathcal{G}_n, 2 \leq i \leq n+1$ is constructed by removing the ith tile from the diagonal of the staircase. Because the top and left side of all polyominoes in $\mathcal{G}_n$ are identical, only right and down moves can differentiate between the polyominoes. Let W be a workspace and $\sigma = \sigma_1\sigma_2\ldots$ be a control sequence sorting $\mathcal{G}_n$. We consider applying σ to n copies W_i of W in parallel; to obtain W_i, we place G_i in W's input region. Let (x_j^i, y_j^i) be the position of the top-left corner of G_i in W_i after applying the first j steps of σ. Let S_j be the size of the largest set $\mathcal{H}_j \subseteq \mathcal{G}_n$ of objects G_i for which (x_j^i, y_j^i) are equal. Because the initial position of all objects is identical, we have $S_0 = n$. We prove that $|\sigma| \geq n$ by proving $S_{j+1} \geq S_j - 1$, i.e., in one step, we can only differentiate one element from the others. If $\sigma_{j+1} \in \{\uparrow, \leftarrow\}$, $S_{j+1} \geq S_j$. If $\sigma_{j+1} = \downarrow$, at most one object can be differentiated from the others by becoming blocked one unit later than the others by an obstacle in the column where it has no diagonal tile. Placing such an obstacle in more than one column results in all objects being blocked at the same position. The situation is analogous for $\sigma_{j+1} = \rightarrow$. □

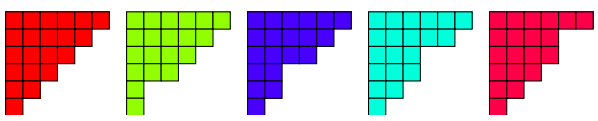

Fig. 5. A family of orthoconvex polyominoes that require linear time to sort in static workspaces can be created by removing single tiles from a staircase polyomino (left).

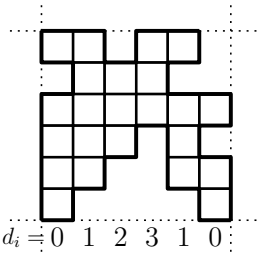

Fig. 6. A completely filled polyomino, its lower, upper, left and right envelope (bold), the corresponding base lines (dotted) and the distance between lower base line and lower envelope.

We also consider the following more general class of polyominoes.

Definition 2. *A polyomino is called* completely filled *iff it consists of all tiles that are below its upper envelope, above its lower envelope, right of its left envelope and left of its right envelope. The* lower base line *of a completely filled polyomino is the horizontal line through its lowest points; see Fig. 6. Upper, left and right base lines are defined analogously.*

Because completely filled polyominoes can have dents, static workspaces are not sufficient to sort or error detect every family of completely filled polyominoes. However, we can prove the following result that allows us to efficiently decide whether sorting and error detection can be done using static workspaces for a given completely filled polyomino.

Theorem 3. *A family $\mathcal{F}$ of completely filled polyominoes can be sorted with static workspaces iff no pair of polyominoes in $\mathcal{F}$ differs only by the depth of dents. Error detection using static workspaces can be done for any completely filled polyomino P iff P does not have a dent for which an error can change the depth.*

Proof. In the following, we argue that for polyominoes P without dents whose depth can be changed by an error, static workspaces suffice for error detection. The statement regarding sorting can be shown in a similar manner. Error detection for P can be done as follows. First we check that the positions where the distance between the baseline and P is zero are correct on all four sides of the given polyomino. This is possible by repeatedly using a construction similar to the one in Fig. 4. This allows us to use a construction such

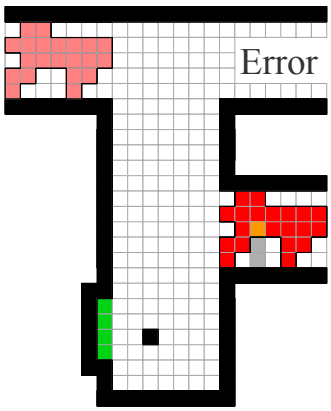

Fig. 7. A gadget that checks the distance between the lower envelope and the polyomino in a certain column. Applying the control sequence ←↓←↑→↑← moves the polyomino to the left exit iff the two gray tiles are empty and the orange tile is present. The height of the window on the left side of the corridor (green tiles) corresponds to the distance between the topmost and the bottommost point where the left envelope touches the left baseline.

as depicted in Fig. 7 to verify the distance between envelope and baseline in some row or column. Note that this step can detect the presence, but not the depth of dents. As a last step, we have to check the boundary for missing tiles. This can be done in a straightforward manner, using constantly many control moves for each tile on the boundary of P. □

V. DYNAMIC WORKSPACES

In this section, we consider *dynamic* workspaces that are composed of rigid obstacles and moving *cams*. Cams are affected by the global controls in the same manner that input polyominoes are. However, they must not enter the input region or any output region. Moreover, we require the sorting or error reporting process to be repeatable; i.e., applying our control sequence must return the workspace to a state that can be used to sort the next incoming polyomino. Dynamic workspaces are considerably more powerful than static ones with respect to sortable objects, workspace size, and sorting speed.

Theorem 4. *Dynamic workspaces can sort any family $\mathcal{F}$ of polyominoes of width up to w and height up to h that are* completely filled *with a sorting sequence of constant length and a workspace of dimensions $\mathcal{O}(|\mathcal{F}| \cdot wh) \times \mathcal{O}(|\mathcal{F}| \cdot wh)$.*

Proof. In the following, we describe how to construct a workspace that sorts a given family $\mathcal{F}$ of completely filled polyominoes with a control sequence of constant length. You can get an intuition of the construction using our interactive visualization applet[1]. In a first step, our procedure groups the polyominoes from $\mathcal{F}$ according to their height and width; we handle each group separately. Therefore we assume in the following that all polyominoes have the same width w and height h. Our sorting procedure checks the left, right, lower and upper envelope separately. For each envelope, constantly many operations are required; therefore, the entire procedure only requires constantly many operations. The main idea of sorting the right envelope is as follows; the construction for the other envelopes is analogous. We use a set of *pins*, one for each row of the polyomino. To sort a polyomino, the pins are pushed against the polyomino from the right. The pins consist of several stages, each stage corresponding to a certain envelope to be tested for. If the envelope matches, a set of interlocking cams called the *plug* unlocks and can be moved to the top, thereby extending a *barrier* that we then use to move the polyomino to the right position. Refer to Fig. 8 for an example of the construction.

In the following, we describe the construction in more detail. Firstly, our construction requires a distance of three between successive rows; therefore, as a first technical step, we use one *expansion cam* per row as depicted in Fig. 8 to introduce additional vertical space. These cams can move left and right independently of each other; therefore, they copy the right envelope of the polyomino they are pushed up against. This requires $\mathcal{O}(wh)$ space, because there must be a horizontal distance of at least w between the vertical parts of each expansion cam to allow them to move horizontally without influencing each other. To the right of the expansion cams, there is one stage for each right envelope E in $\mathcal{F}$. At the left end of each stage, there is a horizontal *driver* cam for each row of the polyomino. Let d_j be the distance between the right envelope and base line in row j, and let d'_j be the distance between right envelope and base line in row j in the previous stage, or 0 for the first stage. Note that due to the polyomino having width w, for at least one j we have $d_j = 0$. The driver in row j has width $3w + d_j - d'_j$. When we push all rows left against the polyomino, this ensures that the ends of all drivers are at the same width iff its right envelope is E. We prevent any vertical motion of the drivers using rigid obstacles placed between the stages. Right of the drivers of each stage we place the *plug* of the stage. The *plug* consists of one *interlocking cam* of height 3 for each row; see Fig. 9 for its dimensions. The parts of each plug can move horizontally according to the right envelope of the polyomino without blocking each other. However, if one of the cams is blocked w.r.t. motion to the top, it blocks all other cams. Let $y_\top, y_\bot$ be the leftmost and rightmost column of the plug if the polyomino has right envelope E. On the upper side of each stage, there is a horizontal wall of rigid obstacles with a window from $y_\bot$ to $y_\top$. On the bottom of each stage, we add a horizontal wall of rigid obstacles with windows of width one at $y_\bot$ and $y_\top$ and two vertical *barriers* extending through these windows. The barriers are long vertical cams that are fixed at their bottom end as depicted in Fig. 8 and cannot move horizontally; they can only move to the top if the interlocking cams are all at the same, correct width for the current stage, i.e., if the polyomino has right envelope E. In this case, the plug can move to the top into a *pocket* that prevents any motion other than to the bottom. The barriers move to the top with the plug, blocking a corridor that the polyomino travels through; their bottom end stays below the bottom wall of the stages, ensuring that the construction can

[1] https://roboticswarmcontrol.github.io/TiltSorting/index.html

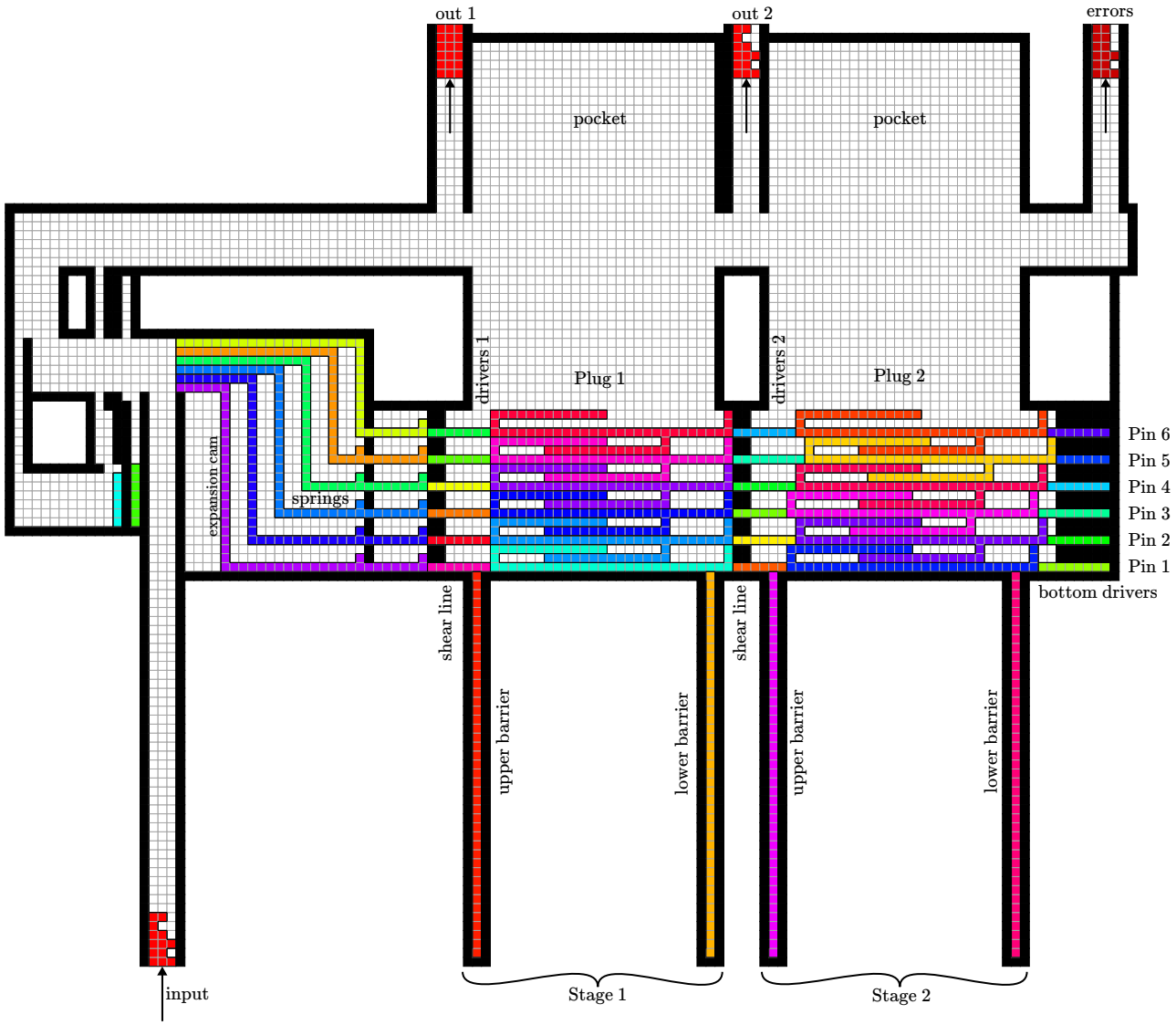

Fig. 8. Example of our construction that classifies polyominoes based on their right envelope. Using the control sequence ↑←↑←↑→↑ moves the red polyomino P (bottom left) out of an exit at the top depending on its right envelope. Afterwards, the sequence ←↓→↓ resets the cams in the workspace to their initial state. The right envelope of P matches the second stage and is moved to the corresponding exit (top center); the first stage is matched by a 6×3-rectangle, matching polyominoes leave through the first exit (top left). Any polyominoes with other envelopes leave through the last exit (top right). See https://roboticswarmcontrol.github.io/TiltSorting/index.html for an interactive visualization applet.

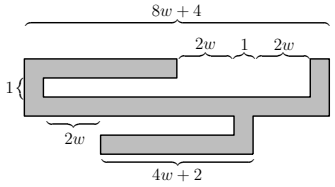

Fig. 9. Dimensions of interlocking cams used in our construction; each stage contains one of these cams for each row of the polyomino.

be reset by a downward move. Right of the last stage, there is one more group of drivers that are held in place by narrow horizontal pockets; see Fig. 8. □

Theorem 5. *For any completely filled polyomino P, errors that change any of the four envelopes can be detected in constant time; thus, for orthoconvex polyominoes, error detection can be done in constant time. Checking for other errors can be done in time linear in the perimeter of P, which is in $\mathcal{O}(wh)$; this is asymptotically optimal in the worst case.*

Proof. To perform error checking for a given completely filled polyomino, we have to make sure that there are no missing tiles along the boundary of the polyomino. The envelope of a completely filled polyomino can be error-checked in constant time, analogous to the construction in the sorting case. In the following, we prove that error-checking the remainder of the boundary requires $\Omega(wh)$ moves in the worst case. To prove this, we consider *comb* polyominoes with $\Omega(w)$ teeth of width five separated by gaps of width one; see Fig. 10. Consider a tile that is part of the right boundary of a tooth and at least three units away from the upper and lower end of the tooth. To verify that this tile is

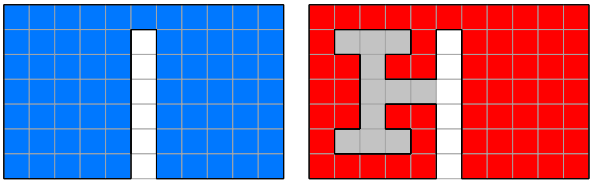

Fig. 10. A comb with 2 teeth of width 5 (left) and an erroneous polyomino (right) containing a trap (light gray). To check for missing tiles in $o(wh)$ moves, a cam would have to be moved into the trap by a $\leftarrow$ control. If the next control is not $\rightarrow$ (in which case the error is not detected), but either $\downarrow$ or $\uparrow$, the cam is trapped. Once a cam is trapped in the polyomino, no control sequence can transfer the workspace into a state without trapped cams.

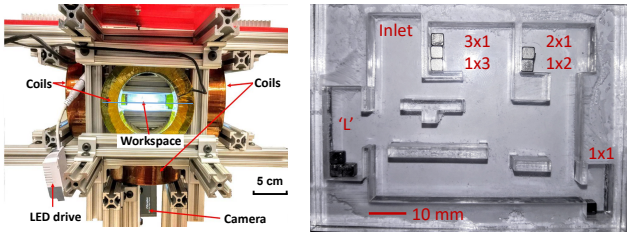

Fig. 11. (left) Magnetic manipulation system used to demonstrate polyomino sorting and error detection. (right) 77 mm $\times$ 56 mm workspace used to sort all polyominoes with 1, 2, or 3 tiles.

present in the given polyomino P, at some point, there must be a probe of size 1×1 to the right of this tile. This probe can either be a rigid obstacle or a cam; if it is a rigid obstacle, for some point in time where the probe is present, there must not be any other obstacle restricting the movement of P to the right. In particular, we can only check one tile on the right boundary of a tooth at each point in time. Therefore to use $o(wh)$ moves to error-check P, we have to use a cam for at least one tile on the right boundary of a tooth. However, it is impossible to do this in general, because there are errors that can *trap* any cam of size 1×1; see Fig. 10. □

VI. Experimental Demonstration

We demonstrated tilt sorting and error detection at the milli-scale using a customized setup that generates a uniform magnetic field to keep the parts aligned in the commanded direction and gravity is used to manipulate the parts.

a) Experimental Platform: The customized electromagnetic system in Fig. 11 has two pairs of coils (18 AWG, 1200 turns, Custom Coils, Inc) arranged orthogonally and powered by four SyRen10-25 motor drivers. Tekpower HY3020E is used as DC power supply. The system can generate up to 101 Gauss uniform fields on the horizontal plane of the workspace center. The coil current was controlled using an Arduino Mega 2560.

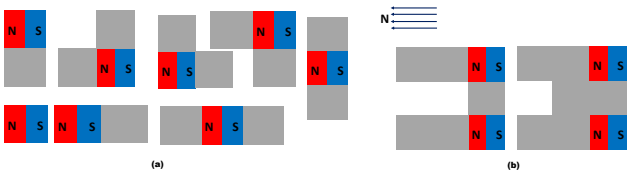

Fig. 12. (a) Schematic of the eight polyominoes used to demonstrate sorting and their alignment in a uniform magnetic field. (b) Schematic of two polyominoes fabricated for error detection.

Each workspace used to demonstrate tilt sorting and error detection was designed in AutoCAD and then cut using a Universal Laser Cutter. Two layers of transparent acrylic were glued together (Gorilla Super Glue) to make a workspace. One layer of 2.0 mm thickness was used as the base and another 5.5 mm thick layer made the obstacle boundary. To perform an experiment, the workspace was placed in the center of the magnetic platform and observed with an IEEE 1394 camera, captured at 60 fps. The polyominoes used for the experiments were fabricated from nickel-plated neodymium cube-shaped magnets (supermagnetman.com C0010 and C0030).

b) Static Workspace Experiments: To show sorting for static workspaces, we designed two workspaces at two scales. The first system used a workspace of 77 mm width and 56 mm length and sorted polyominoes composed of 3.0 mm^3 neodymium cube magnets. The second, smaller system used a workspace of 44 mm width and 35 mm length and sorted polyominoes made of 1.0 mm^3 neodymium cube magnets. An approximate uniform field of 30 Gauss was employed to keep the polyominoes aligned, and the workspace was tilted in the direction sequence $\{\downarrow, \rightarrow, \uparrow, \leftarrow\}$. The direction inputs were applied until the polyomino touched a layout wall. Fig. 12(a) shows the eight polyominoes which were sorted in these experiments. To make a polyomino, one magnetic cube was attached to one or more cubes that had been demagnetized using a blow torch. Fig. 13 shows the four different polyomino shapes in their respective bins inside the workspace and the results after applying control sequence for sorting three polyominoes.

c) Dynamic Workspace Experiments: This experiment used a 59 mm $\times$ 52 mm workspace. A cross-shaped moving cam was used to detect the inner shape of two polyominoes. One polyomino had a dent one-tile deep while the other had a dent two-tiles deep. The cam and the workspace in Fig. 14 are designed so that the polyomino with a one-tile dent is stored in a bin and the other polyomino is rejected. The direction sequence for the parts and the cam is $\{\downarrow, \rightarrow, \uparrow, \leftarrow\}$. Each of the two polyominoes contains two magnetic cubes, one in the top row and the other in the bottom, attached to the demagnetized cubes as shown in Fig. 12(b). Fig. 14 shows the polyomino with a one-tile dent inside an output region and the other polyomino exiting the workspace after applying the control input sequence $\{\downarrow, \rightarrow, \uparrow, \leftarrow\}$. See video attachment for experimental demonstrations.

VII. Conclusions

In this work, we presented algorithms to construct workspaces that are able to sort polyominoes based on their shape. Among others, the following open questions remain. Is it NP-hard to decide whether we can sort a given family of polyominoes using static or dynamic workspaces or is there an efficient algorithm? Are there any simple polyominoes which cannot be error-checked or sorted by dynamic workspaces? How hard is it to find a shortest possible sorting sequence or a smallest possible workspace?

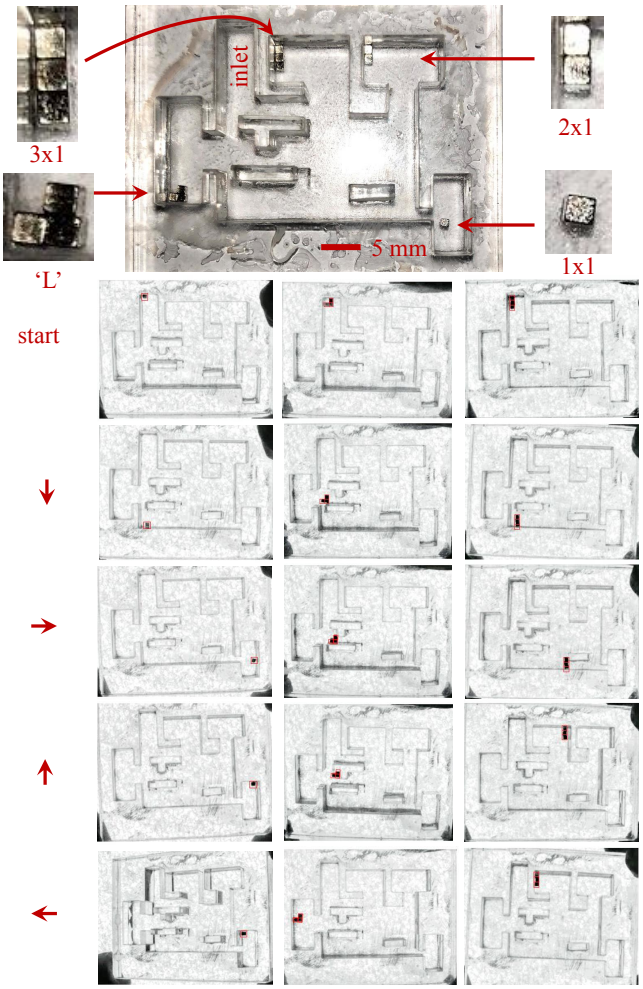

Fig. 13. Frames from video demonstration of sorting polyominoes using the static workspace for 1 mm tiles. See attachment for larger images and video or at https://youtu.be/ZeBur5F7sIo.

If we are able to scale down the size of the tiles for rigid obstacles and cams by a constant factor, we can sort larger classes of polyominoes in a more efficient manner. For instance, static workspaces can measure the depth of dents if the obstacles are a factor of two smaller than the polyomino. In general, how does scaling down the size of the obstacles influence what we can sort and the length of an optimal sorting sequence?

In our model, we put a single polyomino into our workspace and apply the complete sorting sequence before putting the next polyomino in. To increase throughput we could introduce pipelining in the sense that after a certain initial part of the sorting sequence, the next polyomino could be introduced into the workspace. How much throughput can we gain by this, in particular for static workspaces? It may also be feasible to use other polyominoes instead of cams in an otherwise static workspace.

Currently, we only apply a force in one of four directions at the same time. What changes if we were allowed to apply forces in two perpendicular directions at the same time? Our hardware setups used magnetic fields to align the polyominoes and gravity for actuation. As Salmanipour and Diller exploit in [12], the magnetic fields generated by electromagnetic coils are relatively uniform across micro-

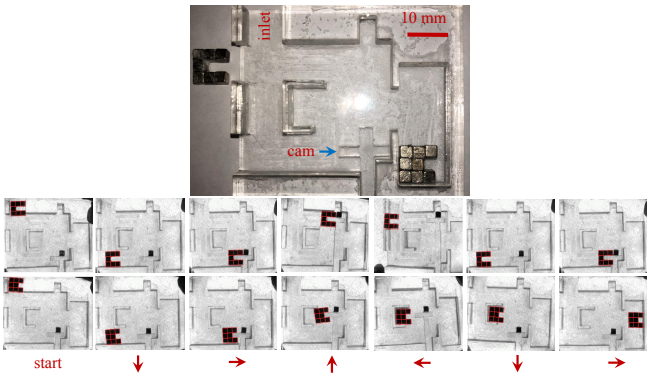

Fig. 14. (Top) A dynamic workspace with one sliding cam, designed for error detection using the sequence $\{\downarrow, \rightarrow, \uparrow, \leftarrow\}$. Only 3×3 polyominoes with a one-tile dent on the right side, as in Fig. 2, are delivered to the right output region. (Bottom) Results for error detection with two polyomino shapes, a two-tile dent (top) and a one-tile dent (bottom). See attachment for larger images and video or at https://youtu.be/ZeBur5F7sIo.

particles, so for sufficiently tiny workspaces the magnet orientation and gradient are approximately uniform. This means the magnetic forces and torques are approximately equal across a sufficiently small workspace. Future work should explore magnetic-based sorting at smaller size scales.

References

[1] Sheryl Manzoor, Samuel Sheckman, Jarrett Lonsford, Hoyeon Kim, Min Jun Kim, and Aaron T. Becker. Parallel self-assembly of polyominoes under uniform control inputs. *IEEE Robotics and Automation Letters*, 2(4):2040–2047, 2017.

[2] Aaron T Becker, Sándor P Fekete, Phillip Keldenich, Dominik Krupke, Christian Rieck, Christian Scheffer, and Arne Schmidt. Tilt Assembly: Algorithms for Micro-Factories that Build Objects with Uniform External Forces. In *The 28th International Symposium on Algorithms and Computation (ISAAC)*, 2017. To appear.

[3] Mengjia Xu, Dimitrios P Papageorgiou, Sabia Z Abidi, Ming Dao, Hong Zhao, and George Em Karniadakis. A deep convolutional neural network for classification of red blood cells in sickle cell anemia. *PLoS Computational Biology*, 13(10):e1005746, 2017.

[4] D. J. Lee, J. K. Archibald, and G. Xiong. Rapid color grading for fruit quality evaluation using direct color mapping. *IEEE Transactions on Automation Science and Engineering*, 8(2):292–302, April 2011.

[5] Michael A Peshkin and Arthur C Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal on Robotics and Automation*, 4(5):524–531, 1988.

[6] Onno C. Goemans, Ken Goldberg, and A. Frank Van Der Stappen. Blades: A new class of geometric primitives for feeding 3D parts on vibratory tracks. *Proceedings - IEEE International Conference on Robotics and Automation*, 2006(May):1730–1736, 2006.

[7] Srinivas Akella, W Huang, Kevin M Lynch, and Matthew T Mason. Sensorless parts feeding with a one joint robot. *Algorithms for Robotic Motion and Manipulation*, pages 229–237, 1996.

[8] Y. Zhang, X. Chen, H. Qi, and D. Balkcom. Rearranging agents in a small space using global controls. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3576–3582, Sept 2017.

[9] K.-F. Böhringer, V. Bhatt, B. R. Donald, and K. Goldberg. Algorithms for sensorless manipulation using a vibrating surface. *Algorithmica*, 26(3):389–429, Apr 2000.

[10] Robert-Paul Berretty, Ken Goldberg, Mark H Overmars, and A Frank van der Stappen. Trap design for vibratory bowl feeders. *The International Journal of Robotics Research*, 20(11):891–908, 2001.

[11] Pankaj K Agarwal, Anne D Collins, and John L. Harer. Minimal trap design. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2243–2248, 2001.

[12] Sajad Salmanipour and Eric Diller. Eight-degrees-of-freedom remote actuation of small magnetic mechanisms. In *IEEE International Conference on Robotics and Automation*, 2018.