On Designing 2D Discrete Workspaces to Sort or Classify 2D 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*— The paper studies the general problem of physically sorting according to shape a polyomino, a 2D structure composed of square tiles joined along edges, 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, deliver the polyomino 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 size of workspace required for the workspace as a function of w and h. By allowing moving polyomino cams that assist in the sorting, we can design dynamic workspaces that call sort all polyominoes that are "completely filled". 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 polyominos 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. Our model assumes individual tiles that stick together when brought in contact. 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.

[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. {atbecker, smanzoor2}@uh.edu. Work from these authors were partially supported by National Science Foundation IIS-1553063 and IIS-1619278.

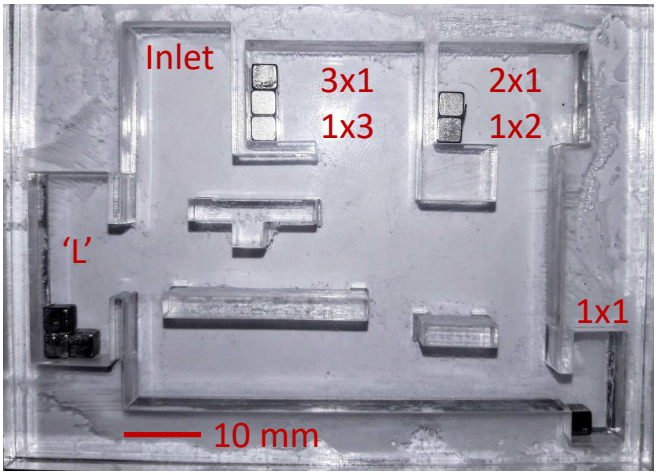

Fig. 1. A 2D workspace designed to sort all polyominos with 1, 2, or 3 tiles into bins that correspond to shape.

II. RELATED WORK

Error detection and shape recognition is a fundamental skill at many size scales in biology, from error detection in DNA strands to how antibodies bind to a specific shape of antigen. Similar skills are used for industrial processes that de-stone and clean grains and that sort the grains into grades TODO: cite paper or online video. While these traditionally use sieves of various sizes and blowing air, machine vision is being increasingly used. Similar processes are used to process minerals.

Sensorless manipulation has a rich history in robotics. Peshkin introduced a framework for designing stationary fences along a conveyor belt to align objects [3]. Early work by Akella et al. demonstrated using a single-joint robot arm mounted above a moving conveyor belt to position and orient planar parts [4]. Recent work by Zhang et al. [5] 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.

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 polyominos 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 polyominos; we know in advance that only polyominos 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 $\mathcal{F}$ is entering 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 polyominos are not wider or higher than P; filtering polyominos 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 polyominos. If P is placed in the input region, it must be moved to an accepting output region; other polyominos must be moved to a rejecting output region. A general limitation of our approach is that we cannot detect holes in polyominos; 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.

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. On the other hand, there are limits to what kind of polyominos can be sorted using static workspaces. For instance, it is impossible to decide 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 pixels not belonging to* P*, where the first pixel is adjacent to exactly three pixels of* P *and all remaining pixels are adjacent to exactly two pixels of* P*.*

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

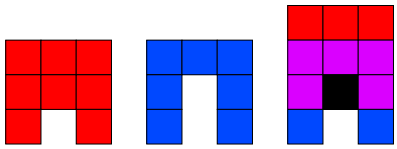

Fig. 2. Two polyominos with a (vertical) dent of depth 1 (left) and 2 (middle). Static workspaces cannot distinguish these polyominos. Introducing a difference between the top-left corners of the polyominos can only be done by moving the polyominos downwards onto a rigid obstacle (right). However, the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

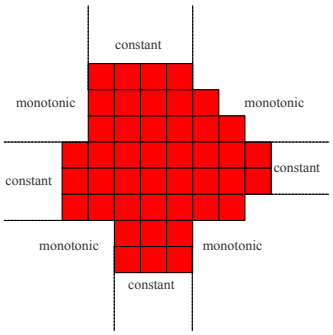

Fig. 3. Decomposition of the boundary of an orthoconvex polyomino.

Theorem 1. *Families* $\mathcal{F}$ *of orthoconvex polyominos 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 of size* $\mathcal{O}(|\mathcal{F}|wh) \times \mathcal{O}(|\mathcal{F}|wh)$*. For orthoconvex polyominos, error detection can be done with a static workspace of size* $\mathcal{O}(w(w + h)) \times \mathcal{O}(h(w + h))$ *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 polyominos 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 orthoconvex; this can be done by checking each individual

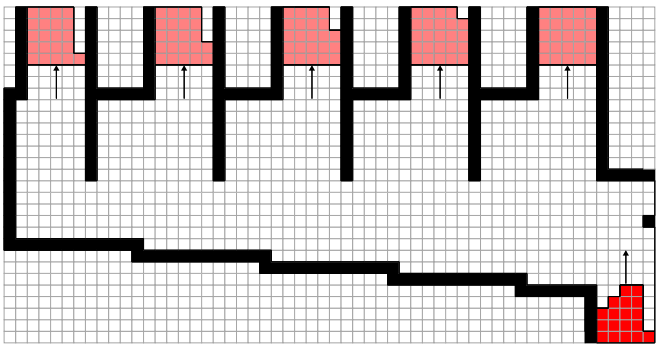

Fig. 4. Applying the control sequence $\uparrow\leftarrow\uparrow\rightarrow\uparrow$ classifies a polyomino based on the row of the top end of the rightmost constant piece of the boundary.

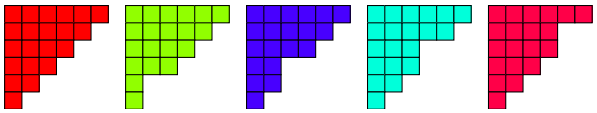

Fig. 5. A family of orthoconvex polyominos that require linear time to sort in static workspaces can be created by removing single tiles from a staircase polyomino (left).

row and column in total time $\mathcal{O}(w+h)$ with a workspace of dimensions $\mathcal{O}(w(\sqrt{w}+\sqrt{h})) \times \mathcal{O}(h(\sqrt{w}+\sqrt{h}))$. □

Moreover we can show that we cannot hope to sort orthoconvex polyominos with fewer than $\Omega(\min(w+h, \mathcal{F}))$ moves in static workspaces. In other words, static workspaces cannot sort orthoconvex polyominos 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 polyominos 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 polyominos in $\mathcal{G}_n$ are identical, only right and down moves can differentiate between the polyominos. 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 $\sigma_{j+1} = \rightarrow$. □

We also consider the following more general class of polyominos.

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.*

Because these polyominos can have dents, static workspaces are not sufficient to sort or error detect every family of completely filled polyominos. 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.

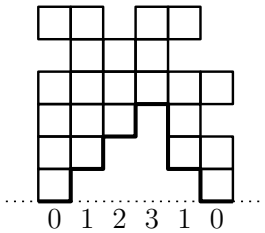

Fig. 6. A completely filled polyomino, its lower envelope and base line and the distance between base line and envelope.

Theorem 3. *A family $\mathcal{F}$ of completely filled polyominos can be sorted iff no pair of polyominos in $\mathcal{F}$ differs only by the depth of a dent. 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 could change the depth.*

V. DYNAMIC WORKSPACES

Dynamic workspaces are composed of rigid obstacles that do not move and moving *cams*. Cams are affected by the global controls in the same manner that input polyominos 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 polyominos 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 polyominos with a control sequence of constant length. You can get an intuition of the construction using our interactive visualization applet on `https://www.ibr.cs.tu-bs.de/projects/tilt-sort/tilt-sort-dynamic-construction.html` In a first step, our procedure groups the polyominos from $\mathcal{F}$ according to their height and width; we handle each group separately. Therefore we assume in the following that all polyominos 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 lower envelope is as follows; the construction for the other envelopes is analogous. We use a set of *pins*, one for each column of the polyomino. To sort a polyomino, the pins are pushed against the polyomino from below. 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 *plug* unlocks and can be moved to the right, thereby extending a *barrier* that we then use to move the polyomino to the right position. Refer to Fig. 7 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 columns; therefore, as a first technical step, we use one *expansion cam* per column as depicted in Fig. 7 to introduce additional horizontal space. These cams can move up and down independently of each other; therefore, they copy the lower envelope of the polyomino they are pushed up against. This requires $\mathcal{O}(wh)$ space, because there must be a vertical distance of at least h between the horizontal parts of each expansion cam to allow them to move vertically without influencing each other. Below the expansion cams, there is one stage for each lower envelope E in $\mathcal{F}$. At the top of each stage, there is a vertical *driver* cam for each column of the polyomino. Let d_j be the distance between the lower envelope and base line in column j, and let d'_j be the distance between lower envelope and base line in column j in the previous stage or 0 for the first stage. Note that due to the polyomino having height h, for at least one j we have $d_j = 0$. The driver in column j has height $3h + d_j - d'_j$. When we push all columns up against the polyomino, this ensures that the lower ends of all drivers are at the same height iff its lower envelope is E. We prevent any horizontal motion of the drivers using rigid obstacles placed between the stages. Below the drivers of each stage we place the *plug* of the stage. The *plug* consists of one *interlocking cam* of width 3 for each column; see Fig. 8 for its dimensions. The parts of each plug can move vertically according to the lower envelope of the polyomino without blocking each other. However, if one of the cams is blocked w.r.t. motion to the right, it blocks all other cams. Let $y_\top, y_\bot$ be the topmost and bottommost row of the plug if the polyomino has lower envelope E. On the right side of each stage, there is a vertical wall of rigid obstacles with a window from $y_\bot$ to $y_\top$. On the left side of each stage, we add a vertical wall of rigid obstacles with windows of height one at $y_\bot$ and $y_\top$ and two horizontal *barriers* extending through these windows. The barriers are long horizontal cams that are fixed at their left end as depicted in Fig. 7 and cannot move vertically; they can only move to the right if the interlocking cams are all at the same, correct height for the current stage, i.e., if the polyomino has lower envelope E. In this case, the plug can move to the right into a *pocket*, preventing any motion other than to the left. The barriers move to the right with the plug, blocking a corridor that the polyomino travels through; their left end stays left of the left wall of the stages, ensuring that the construction can be reset by a move to the left. Below the last stage, there is one more group of drivers that are held in place by narrow vertical pockets; see Fig. 7. □

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 polyominos, 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* polyominos with $\Omega(w)$ teeth of width five separated by gaps of width one; see Fig. 9. Consider a pixel 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. In order to verify that this pixel is actually present in the given polyomino P, at some point, there has to be a probe of size 1×1 to the right of this pixel. 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 pixel 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 pixel 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. 9. □

VI. TRADEOFFS

A. sorting vs. error-detection

Sorting polyominos by type becomes unfeasible because the number of polyominos as a function of the number of tiles n grows rapidly. The asymptotic growth of the number of polyominos was estimated by The complexity of error detection for an n tile polyomino is a function of the perimeter, which grows at ?

The problem can be simplified if we assume polyominos are generated by assembly steps that combine two polyominos

B. Pipelining

VII. EXPERIMENTAL DEMONSTRATION

We demonstrated tilt sorting and error detection at the milli scale using a customized setup which 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. 10 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 Arduino Mega 2560. Each workspace used to demonstrate tilt sorting and error detection is designed in AutoCAD and then cut using a Universal Laser Cutter. Two

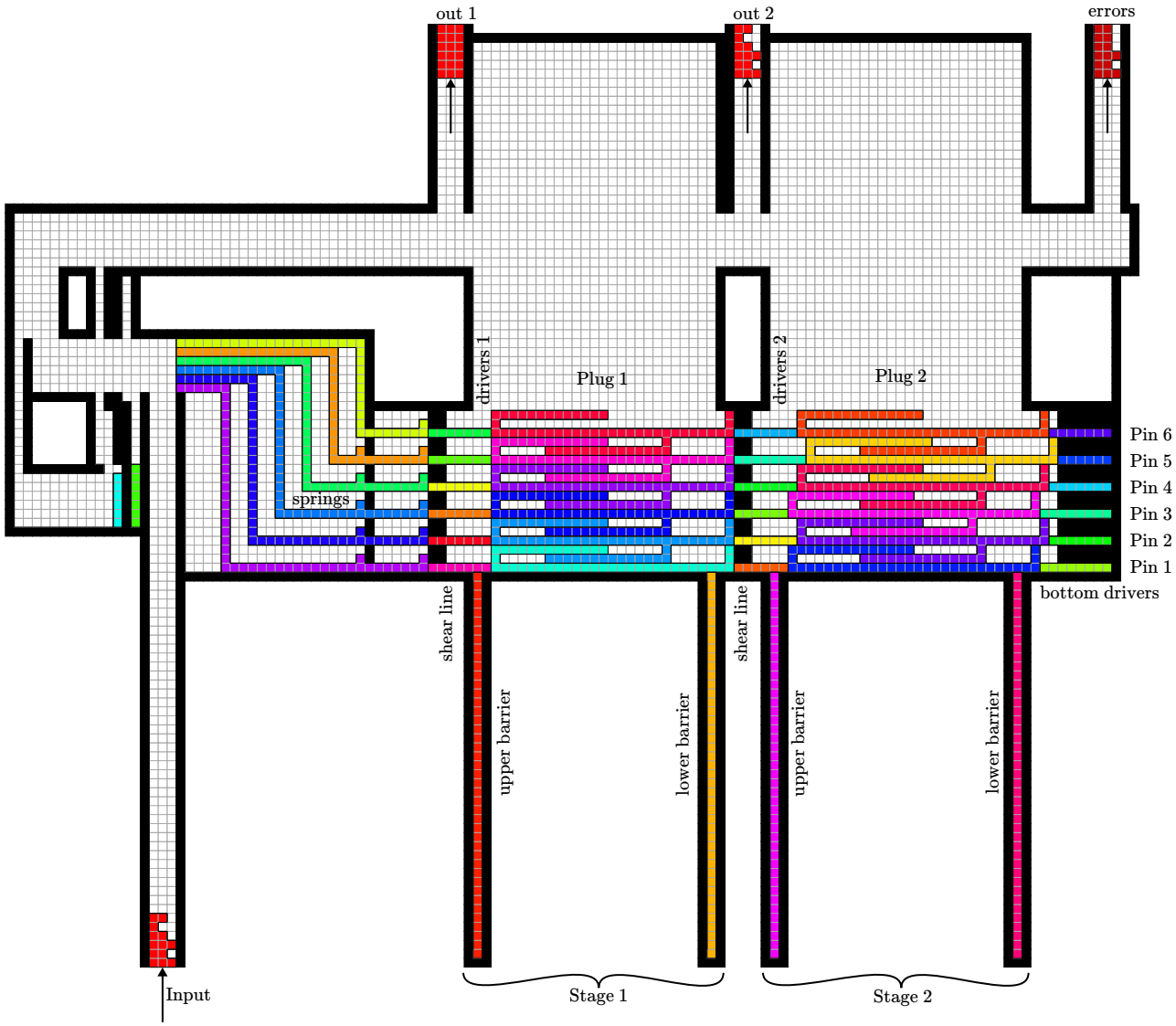

Fig. 7. Example of our construction that classifies polyominos based on their right envelope. Using the control sequence ↑←↑←↑→↑ moves the red polyomino P (bottom left) out of an exit at the top depending on its right envelope. Afterwards, the sequence ←↓→↓ resets the cams in the workspace to their initial 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 polyominos leave through the first exit (top left). Any polyominos with other envelopes leave through the last exit (top right).

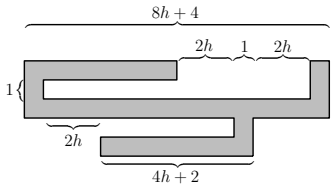

Fig. 8. Dimensions of interlocking cams used in our construction; each stage contains one of these cams for each column of the polyomino.

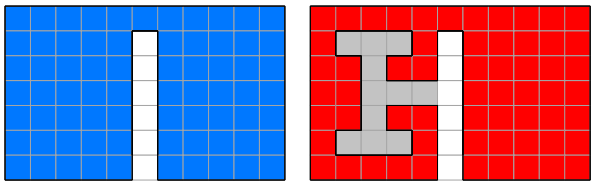

Fig. 9. A comb with 2 teeth of width 5 (left) and an erroneous polyomino (right) containing a trap (light gray). To check for missing pixels in $o(wh)$ moves, a cam would have to be moved into the trap by a ← control. If the next control is not → (in which case the error is not detected), but either ↓ or ↑, 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.

layers of transparent acrylic are glued together to make a workspace. One layer of 2 mm thickness is used as the base and another 5.5 mm thick layer makes the obstacle boundary. To perform an experiment, the workspace was placed in the center of the magnetic platform and observed with a IEEE 1394 camera, captured at 60 fps . The parts used for the experiments are comprised of nickel-plated neodymium cube-shaped magnets (supermagnetman.com C0010).

b) Static Workspace Experiments: To show sorting for static workspaces, we designed two workspaces at two scales. The first system used a workspace of XX mm width and

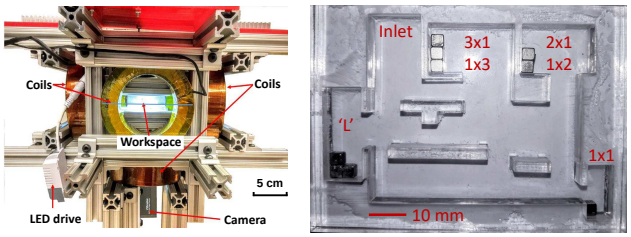

Fig. 10. (left) Magnetic manipulation system used to demonstration polyomino sorting and error detection. (right) XX mm × XX mm workspace used to sort all polyominoes with 1, 2, or 3 tiles.

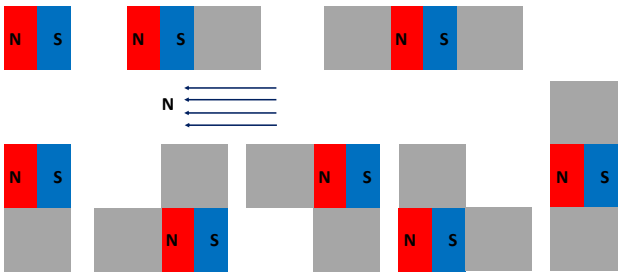

Fig. 11. Schematic of the eight polyominoes used to demonstrate sorting and their alignment in a uniform magnetic field.

XX mm length and sorted polyominoes composed of 2.88 mm^3 neodymium cube magnets. The second, smaller system used a workspace of 20 mm width and 39.5 mm length and sorted polyominoes made of 1 mm^3 neodymium cube magnets. An approximate uniform field of 30 Gauss was employed to keep the parts aligned, and the workspace was tilted in the direction sequence $\{\downarrow, \rightarrow, \uparrow, \leftarrow\}$. For each part a direction input was applied until it touched the layout wall. Fig. 11 shows the eight polyominoes which were sorted in these experiments. To make a part, one magnetic cube was attached to one or more demagnetized cube(s) and the cubes were demagnetized using a blow torch. The workspace in Fig. 12 shows the four different polyomino shapes in their respective bins.

c) Dynamic Workspace Experiments: The workspace used for dynamic workspace experiments is XX mm × XX mm. In this experiment a cross-shaped moving cam is used to detect the inner shape of two polyominoes. One polyomino has one pixel deep dent while the other has two pixel deep dent. The cam and the workspace is designed so

Fig. 12. Frames from video demonstration of sorting polyominoes using the static workspace for 1 mm tiles.

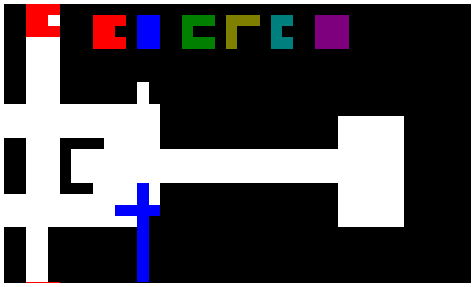

Fig. 13. A dynamic workspace with one sliding cam, designed to sort polyominoes of the type shown in 2 using the sequence $\langle d, r, u, l \rangle$. TODO: label the inlet, the collectors, etc.

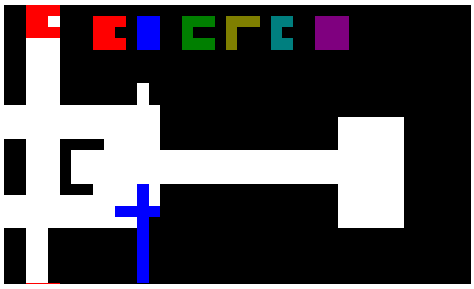

Fig. 14. Frames from video demonstration of the dynamic workspace. The image is from Matlab simulation and it'll be replaced after we have experimental results for it. TODO: label the inlet, the collectors, add scale bar

that the polyomino with a one-pixel 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. Fig. 13 shows the polyomino with one pixel dent inside a bin and the other polyomino exiting the workspace. See video attachment for experimental demonstrations.

VIII. CONCLUSIONS

Open questions: How hard is it to decide whether we can do sorting/error correction using static or dynamic workspaces? Are there any polyominos for which dynamic workspaces cannot do sorting/error detection? Can we always build a pixel probe for any reachable pixel? Can we improve the situation for static workspaces with pipelining? What happens if we allow movement in 8 directions instead of 4 (problematic because of speed/collision)?

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] Michael A Peshkin. Planning robotic manipulation strategies for sliding objects. 1986.

[4] 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.

[5] 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.