Distributed Aggregation for Modular Robots in the Pivoting Cube Model

Sebastian Claici, John Romanishin, Jeffrey I. Lipton, Stephane Bonardi, Kyle W. Gilpin, and Daniela Rus

Abstract— **We present a distributed control strategy for the aggregation of multiple modular robots into one connected structure optimized for use with 3D modular pivoting cube robots such as the 3D M-Blocks [1]. We use the intensity from a light source as input to a decentralized control algorithm that drives the robots together. We describe the algorithm, give provable guarantees on convergence, and discuss experiments carried out in simulation and with a hardware platform of ten 3D M-Blocks modules. In this paper we contribute provably correct algorithms for the aggregation of generic modular robots; we show how these algorithms can be applied on real hardware by evaluating them on the 3D M-Blocks platform.**

I. INTRODUCTION

We wish to develop modular robotic systems capable of responding to global external stimuli through local interactions and distributed control. One implementation of this vision involves using a stimulus source to form an aggregate structure that could be used later as the basis of forming a robot with a desirable shape or as the foundation for performing a group locomotion task. Extensive prior work has addressed the design and control of systems of modular self-reconfigurable robots [2], [3]. In this paper we build on our previous work that introduced the M-Block [4] and 3D M-Block [1] hardware modules, in addition to theoretical work describing reconfiguration algorithms following the pivoting cube model [5]. In the pivoting cube model (PCM), modules or groups of modules are able to rotate about any edge on a cubic lattice, while this framework is not as generally applicable as the more well known sliding cube model, the authors believe it is more practical to implement with real world robotic hardware. We previously presented the 3D M-Blocks [1], a 50mm characteristic length self-reconfigurable modular robot able to implement the (PCM) on a cubic lattice in three dimensions through angular momentum actuation. With this work we extend the capabilities of the 3D M-Blocks modular robots in three areas: (1) hardware, (2) reasoning, control algorithms and simulations and (3) experimental evaluation. The 3D M-Blocks described previously have been augmented with environmental sensing and custom IR-based communication electronics which support cube to cube communications on each face. This new communication system enables decentralized algorithms which require information exchange between modules. The ability to communicate with and detect adjacent modules coupled with environmental sensing abilities allow the new 3D M-Blocks to exhibit group behaviors with local interactions with global guarantees. We are especially interested in the ability of a scattered group of robots to find each other and form an aggregate using the individual modules pivoting motion for locomotion. For rotating modules moving independently on a lattice of robots the challenge is in the localization, coordination and control algorithms for directing module movements. When the modules pivot into the next lattice block, their location can be determined when the robots attach to the next lattice point. However when the modules are in an unstructured environment moving independently of a lattice there is an additional challenge since the location of each module after a pivoting step is uncertain due to bounces subsequent to the pivoting step caused by to the angular momentum transfer involved with movement.

In this paper we consider decentralized control and aggregation for modular robots. We present an algorithm that solves this problem when there is a signal in the environment that follows a gradient that can be sensed by all of the robots at all times. The algorithm is a decentralized optimization with a cost function. We describe the algorithm, prove its convergence to a single aggregate (independent of the initial starting positions of the robots) and present simulation results and describe experiments with a group of ten 3D M-Block modules that are capable of detecting and following light gradients. The algorithm in this paper builds on our prior work on adaptive coverage for mobile robots [6] and extends that body of work to forming aggregates for modular robots that can move independently on the ground, and on a lattice of modules following the pivoting cube model.

Specifically this paper contributes:

- A theoretical discussion of the problem of aggregation in arbitrary configuration settings. Provably correct and efficient algorithms are described in a generic setting that extends to the pivoting cube model naturally.
- A provably correct decentralized control algorithm for aggregation modules following the pivoting cube model, where the robot modules can move independently on a lattice of modules or on the ground.
- An extension of the 3D-M-Blocks hardware to include environmental sensing and an IR based system for face-to-face communications.
- Experiments demonstrating a hardware implementation of several different distributed aggregation algorithms using the 3D M-Blocks where light is the global external stimulus.

A. Related Work

We build on prior work in aggregation, self-reconfiguration and decentralized control optimization. Recent work in cost-

S. Claici, J. Romanishin, J. Lipton, S. Bonardi, and D. Rus are with the Computer Science and Artificial Intelligence Lab, MIT, Cambridge, MA, 02139. {sclaici, johnrom, jlipton, sbonardi, rus}@csail.mit.edu

effective swarm robotics has shown aggregation and reconfiguration in swarms of up to thousands of robots [7], [8]; however, this work has been restricted to 2D structures. An example of aggregation for modular robots that are collectively capable of locomotion in 3D environments has been given in [9] but the modules are not capable of individual motion and must be ferried to their docking locations by wheeled robots. In addition several other systems are able to aggregate swarms of smaller robots into larger structures, including [10] and cluster flow control locomotion using Atron modules [11]. Most similar to our approach is that taken by the SYMBRION project which envisions a swarm robotic system wherein all robots are capable of autonomous motion, and can configure into larger aggregates through reversible connections [12]. To the best of our knowledge, the SYMBRION system has not been implemented in practice at a similar scale to the M-Blocks.

The immediate extension of aggregation is reconfiguration: the planning of a series of moves that transforms one aggregate into another of a different shape. For example, 27 modules can reconfigure into a 3x3x3 cube to move around the environment easily, and reconfigure into a different morphology when faced with obstacles. Research in this area has often assumed connectivity as a strong requirement of the process. For 2D shapes, an algorithm that uses $O(n^2)$ moves for a configuration with n modules has been proposed in the pivoting cube model under a weak connectivity constraint [13]. More recent work showed that under certain conditions on the shape of the initial and final configurations, reconfiguration is possible in $O(n^2)$ moves in both 2D and 3D, and $O(n)$ parallel moves in 2D. Particularly relevant in the context of this paper, reconfiguration can also be achieved through self-disassembly of modules from a regular structure [14], [15].

Related to the problem of aggregation is distributed coverage control. The goal in distributed coverage control is to maximize coverage of an area by finding controllers for individual robots that are designed to function with only local information. While aggregation into a single structure is not the end goal, problems in distributed coverage control share many similarities with aggregation: the control algorithms must be distributed and simple as the robots have limited capabilities; the algorithms must have convergence guarantees; there is often a sensory stimulus that must be maximized. An example of a distributed coverage problem solved through sensory feedback is given in [6], [16]. This problem concerns robots moving in a 2D plane, but the algorithm can be extended to 3D. Recently, these techniques have been applied to create spoof-resilient multi-robot networks [17].

II. DECENTRALIZED AGGREGATION WITH SENSORY STIMULI

In this section we establish theoretical guarantees on convergence under a general framework. Specifically, we assume our robots have integrator dynamics, and can sense the gradient of a sensory function ϕ. We show that for convex ϕ or for unbounded communication radii, aggregation is always achieved, and the convergence rates are bounded. We also explore a constrained version of the problem where robots are restricted to move along low dimensional subspaces of the configuration space, and provide convergence guarantees for such situations as well.

A. Problem Statement and Assumptions

We study the problem of aggregating n robots in a bounded set. Let $\mathcal{W}$ be a closed bounded subset of $\mathbb{R}^N$, and let $\mathbf{p}_i$ denote the position of the i^{th} robot in $\mathcal{W}$.

Assume the i^{th} robot is capable of communication within a ball of radius r_i denoted by $\mathcal{B}(\mathbf{p}_i, r_i)$. Define the sensory function $\phi : \mathcal{W} \to \mathrm{R}$, where $\phi(\mathbf{q})$ is the value of the sensor at position $\mathbf{q}$.

In what follows we assume the robots follow integrator dynamics

$$\dot{\mathbf{p}}_i = u_i.$$

where u_i is the control vector of the i^{th} robot, and $\dot{\mathbf{p}}_i$ is the usual velocity vector.

This assumption is common in the coverage control literature [6], [16], [18], [19]. We further assume that each robot has knowledge of the gradient of the sensory function $\nabla\phi(\mathbf{q})$ at each point $\mathbf{q} \in \mathcal{W}$.

Unlike in coverage control, we do not require that each robot be able to compute its Voronoi cell, nor does each robot need to have any knowledge of surrounding robots.

Under this setting, the goal of each robot is to minimize its distance to every other robot:

$$J_i(\mathbf{p}_{-i}) = \frac{1}{2}\sum_{j\neq i} \|\mathbf{p}_i - \mathbf{p}_j\|_2^2 \tag{1}$$

where $\mathbf{p}_{-i} = (\mathbf{p}_1, \ldots, \mathbf{p}_{i-1}, \mathbf{p}_{i+1}, \ldots, \mathbf{p}_n)$. The global cost function of the system $(\mathbf{p}_1, \ldots, \mathbf{p}_n)$ can be written

$$J(\mathbf{p}_1, \ldots, \mathbf{p}_n) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \|\mathbf{p}_i - \mathbf{p}_j\|^2. \tag{2}$$

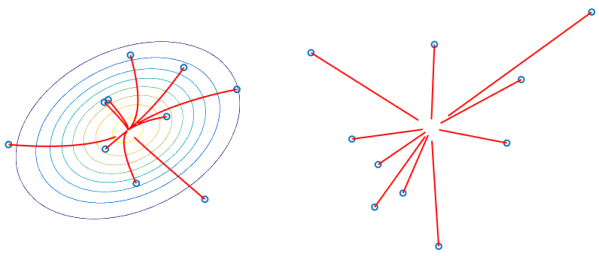
Equations (1) and (2) cannot be optimized directly without knowledge of the position of every robot.

Since $\|\cdot\|_2$ is a metric, we can exploit the triangle inequality and write

$$\sum_{j\neq i} \|\mathbf{p}_i - \mathbf{p}_j\|^2 \leq \sum_{j\neq i} (\|\mathbf{p}_i - \mathbf{q}\|_2^2 + \|\mathbf{p}_j - \mathbf{q}\|_2^2)$$

for any point $\mathbf{q} \in \mathcal{W}$. It is clear that a minimizer for the latter problem is to set $\mathbf{p}_i = \mathbf{q}$ for all i. Thus, finding a control law that drives all robots to the same position $\mathbf{q}$ will also minimize equation (1).

We assume that $\nabla\phi$ is Lipschitz, that is, it satisfies $\|\nabla\phi(\mathbf{x}) - \nabla\phi(\mathbf{y})\| \leq C\|\mathbf{x} - \mathbf{y}\|$ for some constant C and all $\mathbf{x}, \mathbf{y} \in \mathcal{W}$. The Lipschitz condition can be understood intuitively as a bound on the rate of change in a small time frame. We require this condition for several reasons. First, from a practical standpoint there will always be errors in the position estimation and control law of the robot. If ϕ were to vary unboundedly, small errors can compound rapidly. Second, there is a strong correlation between aggregation of robots, and the gradient descent algorithm for finding local minima of functions. Lipschitz continuity of ϕ follows from the local bound on the $\nabla\phi$, and allows us to give qualitative convergence guarantees.



(a) (b)

Fig. 1: Simulation of aggregation control laws. (a) Control law (3); the robots' initial positions are shown as circles and the trajectories shown as red lines; we draw the contour lines of the sensory function. (b) Control law (4); the robots' initial positions are shown as circles with the trajectories shown as red lines.

B. Decentralized Aggregation Control

We wish to derive a control law that will drive the robots together in a reasonable manner. By reasonable we mean here the following:

1) If ϕ is strictly convex, the control law will drive the robots to the same position. While this may seem a stringent requirement on ϕ, note that most point source stimuli (e.g. light or sound) satisfy convexity
2) If for each robot $\mathcal{W} \subseteq \mathcal{B}(\mathbf{p}_i, r_i)$, then the control law will drive the robots to the same position. Effectively, this condition states that the communication radius of the robot includes the entire configuration space. All robots must be able to communicate with all other robots.

The first condition motivates the following control law:

$$u_i = -\alpha \nabla \phi(\mathbf{p}_i) \tag{3}$$

where α is a line search parameter.

Control law (3) is a gradient descent method and is guaranteed to converge to a local minima if ϕ is differentiable, $\nabla\phi$ is Lipschitz, and α is chosen to satisfy the Wolfe conditions [20]. The intuition here is that we want the robot to move towards a direction that has the largest drop in intensity of ϕ, thus moving towards the minimum of ϕ. If ϕ is globally convex, control law (3) converges to the global minimum for each robot. That is to say, we chose $\mathbf{q}$ to satisfy

$$\min_{\mathbf{x} \in \mathcal{W}} \phi(\mathbf{x}).$$

We call ϕ strongly convex if it is twice differentiable and satisfies $\nabla^2\phi \succeq dI$ for some constant d where I is the identity matrix, and $\nabla^2\phi$ is the Hessian matrix $\left(\frac{\partial \phi}{\partial \mathbf{p}_i \mathbf{p}_j}\right)_{i,j}$ for $i, j \in \{1, \ldots, n\}$. If ϕ is strongly convex, then control law (3) converges at exponential rate.

However, we also require that the robots are able to aggregate if they can communicate over the entire environment (the second requirement). Equation (3) does not achieve this for functions that are not globally convex.

To motivate the second control law, consider again the cost function (2). The derivative of J with respect to $\mathbf{p}_i$ is given by

$$\frac{\partial J}{\partial \mathbf{p}_i} = (n-1)\mathbf{p}_i - \sum_{j \neq i} \mathbf{p}_j.$$

This suggests the following control law to force robots together:

$$u_i = -\mathbf{p}_i + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{p}_j. \tag{4}$$

where $\mathcal{N}(i)$ is the set of robots that robot i can directly communicate with; that is, those robots j for which $\mathbf{p}_j \in \mathcal{B}(\mathbf{p}_i, r_i)$.

Note that in the case of $\mathcal{W} \subseteq \mathcal{B}(\mathbf{p}_i, r_i)$, equation (4) reduces to

$$u_i = -\mathbf{p}_i + \frac{1}{n-1} \sum_{j \neq i} \mathbf{p}_j.$$

We recall that a fixed point of a dynamical system is a point where all first derivatives vanish. It is clear that $\mathbf{p}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{p}_j$ is a fixed point of the system. We also recall that a fixed point is asymptotically stable if, as time goes to infinity, the system approaches the fixed point. Formally, we can write:

Definition 2.1: A fixed point x_e of a system $\dot{x} = f(x(t))$ with $x(0) = x_0$ is asymptotically stable if it is Lyapunov stable and there exists a δ such that if $\|x_0 - x_e\| < \delta$ then $\lim_{t \to \infty} \|x(t) - x_0\| = 0$.

Here we have used the definition of Lyapunov stability. Intuitively Lyapunov stability is a bound on how far away a system can get from a fixed point if it starts within some distance δ from the fixed point.

We will use Lyapunov functions to verify the asymptotic stability of our control law [21]. We propose the following Lyapunov function:

$$V_i = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \|\mathbf{p}_i - \mathbf{p}_j\|^2 \tag{5}$$

We need to check the two Lyapunov requirements for asymptotic stability:

- $V_i(\mathbf{x}) > 0$.
- $\dot{V}_i(\mathbf{x}) < 0$ for some neighborhood around the mean position.

The first condition is immediate from the definition of V_i. For the second condition, we have

$$\begin{aligned} \dot{V}_i(\mathbf{p}_i) &= \nabla V_i(\mathbf{p}_i) \cdot u_i \\ &= -|\mathcal{N}(i)| \left(\mathbf{p}_i - \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{p}_j \right)^2. \end{aligned}$$

Since $\dot{V}_i(\mathbf{x}) < 0$ for $\mathbf{x} \neq \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{p}_j$. By LaSalle's invariance principle, the system is asymptotically stable to the mean position of the neighborhood.

C. Stochastic Control

We consider now the problem of finding a control law for robots that have restricted dynamics. Consider the following model:

Definition 2.2: A robot is said to have partially stochastic dynamics if at any time it can select between two actions:

- Move according to the control law $(\dot{\mathbf{p}}_i|V) = (u_i|V)$ for some $V \subset \mathcal{W}$ with $\dim V \leq \dim \mathcal{W}$. That is to say, the motion of the robot is restricted to a subset of free space: for example, a robot moving in two dimensions can be restricted to move only along a straight line.
- Uniformly sample a subset $V' \subset \mathcal{W}$ to move in. Note that since the position of each robot is a continuous function of time, the current position $\mathbf{p}_i$ must be included in the subset V'.

This definition is prompted by particular behavior in the 3D M-Blocks, and a use case will be shown in Section III.

The conditions on V are that it be connected, closed, and convex for the control law to make sense, and that ϕ restricted to V remain continuous. Moreover, if $v_1, \ldots, v_m$ form a basis of V, we require that the partial derivatives $\frac{\partial \phi}{\partial v_i}$ exist for all i.

We consider again the problem of driving the modules together. We wish to minimize the global cost (2) under this new setting. We first note that if ϕ is convex over W, then it is convex when restricted over V. Since V is convex, $\lambda\mathbf{x} + (1-\lambda)\mathbf{y} \in V$ for all $\mathbf{x}, \mathbf{y} \in V$, and convexity of $\phi|V$ follows from convexity of ϕ over $\mathcal{W}$.

We will use the notation $V' \sim \mathcal{U}(\mathbf{x}, \mathcal{W})$ to denote a uniform sampling from $\mathcal{W}$ of subsets that contain the point $\mathbf{x}$. For a concrete example, consider sampling the set of all lines passing through a point $\mathbf{x}$ in two dimensions. This can be accomplished by sampling uniformly a direction vector $\mathbf{y}$ and taking the line $\mathbf{x} + \mathbf{y}$.

Therefore, we can adapt the control law (3) to

$$u_i = \begin{cases} -\beta\nabla(\phi|V)(\mathbf{p}_i), & \text{if} \quad \nabla(\phi|V)(\mathbf{p}_i) \neq 0, \\ V \sim \mathcal{U}(\mathbf{p}_i, \mathcal{W}), & \text{otherwise.} \end{cases} \tag{6}$$

The sampling step is only performed after the robot has reached a minimum of $\nabla(\phi|V)$. The procedure is detailed in Algorithm 1.

Algorithm 1 Stochastic Control

```
repeat
    if there is a direction of decrease in the stimulus then
        set  u_i = −β∇(φ|V)(p_i)
    else
        sample new subspace in which to move
until good enough solution is reached
```

Algorithm 1 is not guaranteed to converge for arbitrary ϕ and sampling procedures. However, under certain assumptions on the sampling procedure, we can provide convergence guarantees of (6) to the true global minimum of the function. By convergence, we mean that there exists some time T such that for all $t > T$, we have $u_i = 0$. Examples include moving only along different coordinate axes (a variant of coordinate descent [20]).

Let us restrict ourselves to the case where V is sampled uniformly over all m-dimensional hyperplanes where $m < n$ (also called m-dimensional flats). To sample an m-dimensional flat that contains $\mathbf{p}_i$, we can sample m points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ from $\mathcal{W}$, and define the flat as intersection between the linear span of $\{\mathbf{p}_i, \mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $\mathcal{W}$.

We will prove the following:

Theorem 2.1: If V is sampled uniformly over the set of m-dimensional flats, and ϕ is convex over $\mathcal{W}$, then control law (6) converges.

We will make use of the monotone convergence theorem which we restate here in a simple form (see [22]):

Theorem 2.2: (Monotone convergence) If $(\mathbf{a_n})$ is a monotone sequence of real numbers, then this sequence has a finite limit if and only if the sequence is bounded.

We are now ready to prove Theorem 2.1.

Proof: Let the $\mathbf{p}_i^1, \ldots, \mathbf{p}_i^n$ be the sequence of points for which $\nabla\phi(\mathbf{p}_i^j) = 0$. By the convexity of ϕ, we have $\phi(\mathbf{p}_i^1) \geq \phi(\mathbf{p}_i^2) \geq \ldots \geq \phi(\mathbf{p}_i^n)$. This series is monotone and bounded since ϕ is bounded. Applying the monotone convergence theorem, we can conclude that control law (6) converges. ■

We conclude this section with a simulation result. For the simulation, we assumed there are $n = 10$ robots placed according to a normal distribution in $\mathbb{R}^2$. We use a Gaussian function

$$\phi(\mathbf{x}) = e^{(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)^T}$$

as the sensory stimulus with $\mu = 0$ and Σ equal to the identity plus a small off-diagonal term.

We use control laws (3) and (4) in two simulation experiments. For control law (3), we fix $\alpha = 1$ as the step size, and perform gradient ascent. Simulation runs are shown in Fig. 1. Both control laws converge to a single point, thus achieving aggregation. We note here that convergence rate depends on the choice of parameters and control law. If the robots can communicate with all other robots, then moving towards a shared center can be significantly faster.

Of course, it is always possible that the sensory function has multiple local minima, and after aggregation the robots are split into distant clusters (such an example is shown in Fig. 2). However, in such cases it is impossible to achieve a single aggregate without assuming prior knowledge about the structure of the environment, sensory function, or knowledge of the number of robots in the system. Such assumptions are untenable in our physical implementation using the M-Blocks, and generally untenable for modular robotics systems.

In the sections that follow, we explore a specific example of aggregation in the pivoting cube model.

III. AGGREGATION OF MODULAR ROBOTS IN THE PIVOTING CUBE MODEL

Following Section II, we implement the algorithms described in the pivoting cube model. We showcase an example arising from our hardware implementation where the stochastic control described in the previous chapter is important.

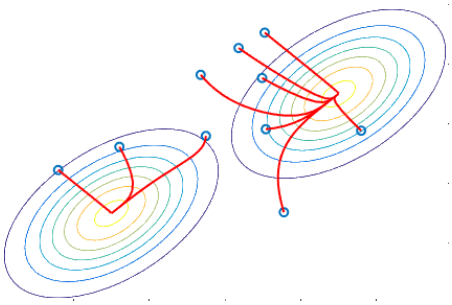


Fig. 2: A single aggregate is impossible in this scenario. The robots are distributed in an environment with two local maxima and have very limited communication capabilities. As can be observed, two aggregates are formed. Without prior knowledge of the number of modules, such scenarios cannot be resolved into a single aggregate.

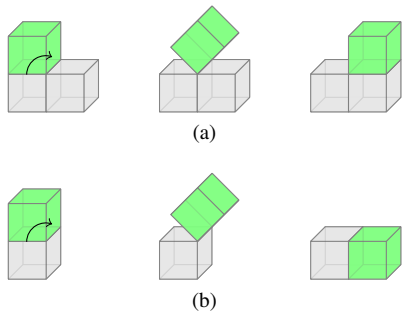


Fig. 3: Pivoting moves. In these examples the green module pivots around the gray modules. (a) Pivot by $\pi/2$ moves the M-Block horizontally. (b) Pivot by π moves the M-Block diagonally.

A. *Theoretical Results*

Let $\mathcal{W}$ be a bounded two dimensional region into which we embed a lattice structure $\mathcal{L}$, for example a 2D grid over $\mathbb{R}^2$. Assume there are n modules, and let $\mathbf{p}_1(t), \ldots, \mathbf{p}_n(t) \in \mathbb{N} \rightarrow \mathcal{L}$ denote the positions of the modules at time t. All modules have the same mass and the laws of gravity hold. Each module can pivot about its edges to reach new positions on the lattice. The modules move independently of one another and require only information about edge connected neighbors to determine whether pivot moves are valid. We assume it takes unit time for a cube to execute a move from one lattice position to another.

Since our focus is on aggregation and not path planning, we assume there are no obstacles in the environment. To achieve aggregation, each module will independently follow the maximum direction of a stimulus located at position $\mathbf{p}_L$. We say that the process has converged if there are no viable moves that would move any module in the environment closer to the stimulus source. We assume the modules are initially scattered throughout the environment, which allows us to restrict our attention to the two-dimensional plane.

The 3D M-Blocks have sensors embedded on each of the six faces that output a number from 0 to 1024 representing the intensity of the sensor readings on that particular face. Let $I_1, \ldots, I_6$ be the intensity readings on each of the 6 faces.

If the algorithm converges at time t_f, we try to minimize the maximum distance between any module and the stimulus source at t_f:

$$\min \max_i \|\mathbf{p}_i(t_f) - \mathbf{p}_L\|. \tag{7}$$

where $\mathbf{p}_L$ is the position of the stimulus source and $i \in \{1, \ldots, n\}$.

Since $\mathbf{p}_L$ is difficult to estimate, each module will attempt to maximize a fitness function that uses the stimulus intensity readings:

$$\frac{1}{|\{I_j | I_j \neq 0\}|} \sum_{I_j \neq 0} I_j \tag{8}$$

Equation (8) is an average over faces that have non-zero stimulus intensity readings. As we only count non-zero intensities, the maximum is achieved when a module is directly below the source, as then Equation (8) reduces to $\max_j I_j$.

We will use the notation $I_{\mathbf{n_j}}$ to denote the stimulus intensity reading on the face with surface normal $\mathbf{n_j}$.

To drive towards a stimulus source, each module must first estimate the direction of the source. This is the purpose of Algorithm 2. This direction is used as a gradient in Algorithm 3 to drive the module towards the stimulus. Algorithm 3 can be interpreted as greedily selecting the move that moves the robot closest to the stimulus source and repeating until convergence.

Any move is a translation that can be written as a sum of at most two face normals (e.g. the move that takes a module from $(0,0,0)$ to $(1,1,0)$ can be written as a sum of $(1,0,0)$ and $(0,1,0)$). A move is *possible* if the volume swept by the module during rotation does not collide with any other module or the environment. We say a move is *weakly admissible* if the intensity reading on at least one of the faces corresponding to the normals is greater than 0. We call a move *strongly admissible* if the intensity reading on all of the faces corresponding to the normals is greater than 0 (see also Fig. 4).

We can use this information to provide a estimated direction towards the stimulus source. Using the notation described above of $\mathbf{n_j}$ for the normal of a face, we can estimate the direction of the stimulus source as

$$\sum_{j \in \text{faces}} I_j \mathbf{n_j}.$$

Algorithm 2 implements the stimulus direction estimation strategy discussed above.

Algorithm 2 Estimate direction of stimulus source

```
1: function EstimateStimulusDirection(Cube i)
2:     for each face j do
3:         I_j ← StimulusIntensity(j)
4:         n_j ← surface normal to face j
5:     return Σ_faces I_j n_j
```

Algorithm 3 acts as a controller for each module. Note the following: we choose the move whose direction is closest to the direction of the stimulus source estimated using Algorithm 2. Also, note that we only iterate over feasible moves in line 4.

Algorithm 3 Drive cube towards estimated direction

```
1:  function STEP(Cube i)
2:      d ← ESTIMATEDSTIMULUSDIRECTION(Cube i)
3:      sort moves by distance to d
4:      for each move M do
5:          let n_1, ..., n_k s.t. Σ_{i=1}^{k} n_i = M
6:          if ∃j ∈ 1, ..., k s.t. I_{n_j} > 0 then
7:              return M
8:      return NIL
9:
10: function DRIVE(Cube i)
11:     M ← STEP(Cube i)
12:     while M ≠ NIL do
13:         apply move M
14:         M ← STEP(Cube i)
```

We wish to prove the following:

Theorem 3.1: For a cube with initial position $\mathbf{p}_i(0)$ that only performs *weakly admissible* moves there is only a finite set of coordinates at which it can later reside. This set is the sphere in the l_1 norm with radius

$$\|\mathbf{p}_L - \mathbf{p}_i(0)\|_1$$

and center $\mathbf{p}_L$.

Proof: For our hardware platform, the stimulus is a light intensity reading. In this case we can use Lambert's law

$$I_{\mathbf{n_j}} \propto \mathbf{n_j} \cdot \mathbf{d_j}. \tag{9}$$

to ensure that intensity readings on faces that are oriented away from the stimulus source read 0. In cases where this does not hold (e.g. sound intensity), we can use just the largest two values and all proofs follow.

As we assume *weak admissibility*, the module cannot take a move that would place it farther away in the l_1 norm from the stimulus source than $\mathbf{p}_i(0)$, for such a move would have a component oriented away from the stimulus source, and would thus not be chosen by Algorithm 3. ■

For the 2D case, Figure 4 shows the set of coordinates at which the blue cube can reside. In yellow are shown the points satisfying *weak admissibility*, while in green are shown those satisfying *strong admissibility*.

Strong admissibility guarantees convergence as the module can only move closer to the stimulus source. To ensure convergence with *weak admissibility*, we require an extra $O(1)$ memory per cube to detect two step cycles.

We now show that Algorithm 3 converges to the global maximum of Equation (8) for one module.

We can prove the following:

Theorem 3.2: Algorithm 3 converges to a single aggregate. Moreover, the following two properties will be satisfied:

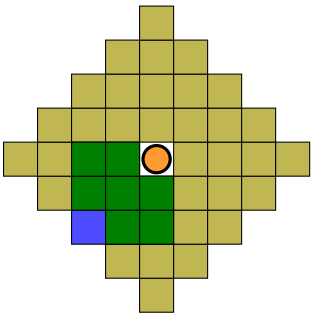


Fig. 4: Coordinates at which the blue module can reside. In green are positions that are *strongly admissible*, while *weakly admissible* positions are in yellow. The circle represents the projection of the stimulus source onto the plane.

1) There will be a module in the final configuration that is at the closest lattice point to the projection of the stimulus source on the plane.
2) The final configuration has no holes.

To give an example of Property 1, consider several 3D M-Blocks under a light source. Property 1 states that there will always be an M-Block on the lattice point closest to the light source.

Proof: We first prove Property 1. Assume there is no such module. For a module to be unable to move closer to the stimulus source, there must be other modules one step closer across both directions in the Manhattan norm. This chain of modules is finite and terminates either with a module located closest to the projection of the stimulus source, or with a module that is able to take a move closer to the stimulus source. But we can take this move and repeat the argument. Since the set of possible moves is finite, this process will eventually terminate when there is a module that is located at the closest lattice point to the projection of the stimulus source.

Assume now that the algorithm converges but there are multiple aggregates. Let C_L be the module in Property 1. Find the closest module to C_L that is not part of the same aggregate as C_L. If this module cannot move closer to the stimulus source, there are modules blocking it, but these modules are then closer to the stimulus source, and thus closer to C_L while still part of a different assembly contradicting our assumption that we chose the closest module.

Property 2 follows by an analogous argument to the above. ■

To determine whether a move is possible only requires local information (knowledge of modules connected on each face) about a cube's neighbors in the plane. Each module requires $O(1)$ time to make a decision about which move to take. Algorithm 3 thus scales to arbitrarily many modules and converges in time proportional to the maximum l_1 distance between any module and the projection of the stimulus source on the plane.

B. Stimulus Tracking with Stochastic Control

The algorithms developed above guarantee convergence when the modules are restricted to move along a lattice

structure. For example, if there is a base of "scaffold" modules that are not capable of actuating, but provide support for modules that can actuate, the algorithms above apply. One example is in automatic manufacture of buildings: a few modules can be active at a time, and reside on a base of inactive modules. Selectively aggregating just a few modules at a time, simple shapes such as cuboids and prisms can be constructed. However, when the modules are moving on an arbitrary surface, the inertial forces generated during reorientation and motion are sufficiently high to cause the robot to swerve. This motion often reorients the forward direction of the M-Block; as such, the algorithms described previously cannot be applied as is. We show, however, that this situation is captured by Algorithm 1.

We assume the M-Block can be modeled as a point $\mathbf{x}$ endowed with a direction vector $\mathbf{u}$. The stimulus source is a circle centred at $\mathbf{x_L}$ with radius R_L (write $\mathcal{C}_L$ for this circle). The M-Block has reached the goal if $\|\mathbf{x} - \mathbf{x_L}\|_2 \leq R_L$. In continuous space, the control strategy is:

1) Move along $\mathbf{u}$ or $-\mathbf{u}$ to closest position to light source. This effectively projects $\mathbf{x_L}$ onto the line passing through $\mathbf{x}$ that has direction $\mathbf{u}$.

$$\mathbf{x}' = \mathbf{x} + \mathbf{u}^T(\mathbf{x_L} - \mathbf{x})\mathbf{u}$$

2) If the goal is not reached, sample a new direction vector $\mathbf{u}$ uniformly at random.

Intuitively, steps 1 and 2 describe a situation where the module moves as much as possible towards the stimulus source along the line it is currently oriented towards, and when it cannot improve its position further, the module performs a reorientation maneuver.

The equivalence to control law (6) follows immediately. If the gradient of the stimulus function restricted to the line of motion of the robot is non-zero, we simply move along this line until we reach a maximum (step 1). Otherwise, we reorient the robot, effectively sampling a new line along which to travel (step 2).

As a line is an m-dimensional flat in $\mathbb{R}^2$, we can apply Theorem 2.1 to prove convergence.

IV. RESULTS AND EXPERIMENTAL DATA

To verify the correctness of our algorithms, we implemented them in simulation and on our hardware platform, the 3D M-Blocks. As a stimulus, we use the light intensity from a point light source located above and to the side of the modules.

A. Simulation Results

To test aggregation in free space, we implemented a Monte Carlo simulation of the algorithm. We averaged 100 simulation runs and plotted the expected number of moves until convergence on a 100×100 grid with $R_L = 1$, and $R_x = 1$ (Fig. 5). The stimulus source is placed at the origin. Recall that R_L is the radius of a circle around the stimulus source at which we consider the modules to have converged. We can define R_L with respect to the size of the modules, but we must have $R_L > 0$ as otherwise the convergence set has measure 0. We can observe that there is a quadratic convergence rate to the origin from all points which is independent of R_L.

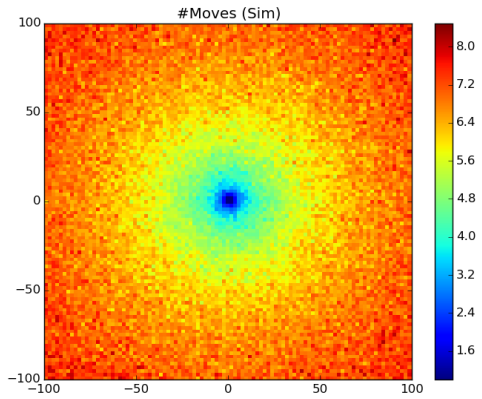


Fig. 5: We measured the number of moves until convergence starting from each grid cell and averaged over 100 experimental runs. A quadratic convergence rate can be observed.

B. Hardware Experiments

We tested the light tracking algorithm running in a distributed fashion on ten 3D M-block modules [1], which have been upgraded to include an ambient light sensor on each face, an additional IMU on the module's frame, and an additional Arduino programmable processor. The experiment was performed with no centralized information, controller or module-to-module communication, with the algorithm running on the embedded Arduino programmed processor. The modules were free to move in an environment 0.9 m by 0.6 m with foam covered floor and walls. In every experimental run the modules moved closer towards the light source on average, and the majority of the modules joined aggregates (Fig. 6)

The goal of this experiment is two-fold; first to minimize the distance of each module from the stimulus source; second to form the smallest number of aggregates with the maximum number of cubes per aggregate in order to eventually facilitate shape reconfiguration. There are several minor differences between the algorithm running on the 3D M-Blocks and the simulations, which are artifacts the hardware. As opposed to the theoretical implementation which exhibits predictable dynamics, the 3D M-Blocks modules exhibit complex dynamics and motion, especially when considering interactions involving magnetic and inertial forces with neighboring modules. Additionally due global communication constraints, the modules stop moving when they connect to at least one other module, forming an aggregate. This is in contrast to the theoretical model where the modules are able to move independently regardless of the positions of neighboring modules. In future work we home to add the ability for aggregates of modules to work together to move as a group.

V. CONCLUSIONS

In this paper, we presented a decentralized control algorithm for the aggregation of modular robots following the

(a) 0 s (b) 30 s (c) 60 s (d) 90 s (e) 120 s

Fig. 6: Still frames illustrating one representative run of the distributed light tracking algorithm running on eight 3D M-Blocks modules. (a) Shows the starting locations of the modules. (e) Shows the final location after the completion of the experiment.

TABLE I: Experimental data from our hardware experiment. We measured the following statistics: 1) Average distance of the center positions of the modules as measured from an overhead camera. 2) maximum distance traveled by a single module. 3) number of modules in the largest aggregate (defined as physically connected modules, either edge to edge or face to face connections). 4) number of modules that are in an aggregate of at least two modules compared to the total number of active modules.

	Mean Distance (m)	**Maximum Distance (m)**	**Largest Aggregate**	**Modules Aggregated**
1	0.25	0.64	3	5 / 8
2	0.13	0.58	3	7 / 10
3	0.106	0.37	2	6 / 10

pivoting cube model. We derived an algorithm for generic modular robots following integrator dynamics that tracks a sensory stimulus (in our case a light source) and gave provable guarantees for its correctness and convergence. We implemented the control scheme on the 3D M-Blocks, our hardware platform for the pivoting cube model, which demonstrates the practicality of the scheme.

There are many potential extensions to our work. We are investigating a scheme for reconfiguration. Note that, unlike the algorithm presented in [5], we are faced with several problems: all intermediary structures must be stable and not topple during a move; some moves are difficult for the 3D M-Blocks hardware and are to be avoided; and removing the need to use an intermediate structure (in [5] a line structure) for reconfiguration is desirable for efficiency.

ACKNOWLEDGMENT

This work is supported by the NSF through grants 1240383 and 1138967. The authors would like to thank Elizabeth Mittmann for her help with the experiments, and the NDSEG fellowship.

REFERENCES

[1] J. W. Romanishin, K. Gilpin, S. Claici, and D. Rus, "3D M-Blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1925–1932.

[2] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, pp. 43–52, 2007.

[3] J. Neubert, A. Rost, and H. Lipson, "Self-soldering connectors for modular robots," *Robotics, IEEE Transactions on*, vol. 30, no. 6, pp. 1344–1357, 2014.

[4] J. W. Romanishin, K. Gilpin, and D. Rus, "M-blocks: Momentum-driven, magnetic modular robots," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4288–4295.

[5] C. Sung, J. Bern, J. Romanishin, and D. Rus, "Reconfiguration planning for pivoting cube modular robots," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. Seattle, WA: IEEE, May 2015.

[6] M. Schwager, J. McLurkin, and D. Rus, "Distributed coverage control with sensory feedback for networked robots," in *robotics: science and systems*, 2006.

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

[8] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3293–3298.

[9] P. Levi, E. Meister, and F. Schlachter, "Reconfigurable swarm robots produce self-assembling and self-repairing organisms," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1371–1376, 2014.

[10] M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, and C. J. Taylor, "Towards robotic self-reassembly after explosion," *Departmental Papers (MEAM)*, p. 147, 2007.

[11] A. Shokri and E. Masehian, "A meta-module approach for cluster flow locomotion of modular robots," in *Robotics and Mechatronics (ICROM), 2015 3rd RSI International Conference on*. IEEE, 2015, pp. 425–431.

[12] W. Liu and A. F. Winfield, "Distributed autonomous morphogenesis in a self-assembling robotic system," in *Morphogenetic Engineering*. Springer, 2012, pp. 89–113.

[13] N. M. Benbernou, "Geometric algorithms for reconfigurable structures," Ph.D. dissertation, Massachusetts Institute of Technology, 2011.

[14] K. Gilpin, A. Knaian, and D. Rus, "Robot pebbles: One centimeter modules for programmable matter through self-disassembly," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2485–2492.

[15] K. Gilpin, K. Koyanagi, and D. Rus, "Making self-disassembling objects with multiple components in the robot pebbles system," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3614–3621.

[16] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[17] S. Gil, S. Kumar, M. Mazumber, D. Katabi, and D. Rus, "Guaranteeing spoof-resilient multi-robot networks," in *Robotics Science and Systems*, 2015.

[18] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1327–1332.

[19] L. C. Pimenta, V. Kumar, R. C. Mesquita, and G. A. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 3947–3952.

[20] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[21] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. prentice-Hall Englewood Cliffs, NJ, 1991.

[22] E. M. Stein and R. Shakarchi, *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009.