Aggregation for Modular Robots in the Pivoting Cube Model

by

Sebastian Claici

B.S., University of Southampton (2014)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

Author . . . . . Signature redacted . . . . . .
Department of Electrical Engineering and Computer Science
August 30, 2016

Certified by . . . Signature redacted . . . . . .
Daniela Rus
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . . . Signature redacted . . . . . .
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Theses

Aggregation for Modular Robots in the Pivoting Cube Model

by

Sebastian Claici

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 2016, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

In this thesis, we present algorithms for self-aggregation and self-reconfiguration of modular robots in the pivoting cube model.

First, we provide generic algorithms for aggregation of robots following integrator dynamics in arbitrary dimensional configuration spaces. We describe solutions to the problem under different assumptions on the capabilities of the robots, and the configuration space in which they travel. We also detail control strategies in cases where the robots are restricted to move on lower dimensional subspaces of the configuration space (such as being restricted to move on a 2D lattice).

Second, we consider the problem of finding a distributed strategy for the aggregation of multiple modular robots into one connected structure. Our algorithm is designed for the pivoting cube model, a generalized model of motion for modular robots that has been effectively realized in hardware in the 3D M-Blocks. We use the intensity from a stimulus source as a input to a decentralized control algorithm that uses gradient information to drive the robots together. We give provable guarantees on convergence, and discuss experiments carried out in simulation and with a hardware platform of six 3D M-Blocks modules.

Thesis Supervisor: Daniela Rus
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I gratefully acknowledge the contributions of my advisor, Daniela Rus, and labmates in the Distributed Robotics Lab. In particular, John Romanishin has been incredibly helpful in this project, as his mastery of mechanical engineering gave birth to the M-Blocks without which this project would not have existed. I also appreciate the thoughtful discussions on reconfiguration shared with Cynthia Sung and Jingjin Yu.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

The complexity of biological structures across scales from the molecular to the macroscopic is the result of self-organization processes. Central to these processes is the aggregation and reconfiguration of matter into a variety of morphologies. Self-assembly confers on biological cells the ability to produce an almost unlimited variety of structures [39, 40]. It is no surprise that the flexibility, reliability and speed of self-assembly has attracted interest from an engineering perspective [41, 43].

One clear advantage self-assembling systems have over purpose built robots is their tolerance of failure. Faulty modules can be swapped out on the fly and replaced by healthy modules. This almost infinite adaptability allows not only for a great number of morphologies, but also ensures the robustness of the system as a whole.

If such modular robots are programmable, can communicate, and are capable of detaching formed connections, they can be exploited to form almost any multi-module configuration. Such abilities would greatly facilitate fabrication techniques such as material printing, would allow for targeted complex anti-viral systems to be deployed seamlessly, and would yield heterogeneous tool sets thus facilitating ease of use. Nature's molecular self-assembly systems do not have the same freedom, but are able to exploit conformational dynamics to switch between states and can thus achieve hierarchical sequencing of assembly steps. It is a challenge to

reproduce the staggering complexity of natural self-assembly systems in modular robotic systems with only single state modules.

The problem we address in this thesis is aggregation. Aggregation is the process by which a group of scattered robots gathers in a single location to be able to share information or overcome larger obstacles. The aggregation process can be performed relying on long range communication between robots, but as modular robots are meant to be deployed in remote and unknown scenarios, relying on communication is untenable. Instead, we develop the problem assuming all robots are capable of sensing a environmental stimulus. For example, in planetary exploration applications, the robots may have a light sensor to detect the position of the sun, or a compass to detect changes in the magnetic field. Such sensors are usually small, lightweight, and fault tolerant. The robots may even be endowed with several redundant sensors to further increase robustness.

In this thesis, we develop the theory of self-assembly by providing algorithms for the aggregation of modular robots under mild conditions. To the best of our knowledge, these algorithms are the first of their kind to be developed for the specific task of aggregation. We build on our previous work that introduced the M-Block [26] and 3D M-Block [25] hardware modules (Fig. 1-1), in addition to theoretical work describing reconfiguration algorithms in the pivoting cube model [36].

1.1 Challenges

In an attempt to build a self-sufficient modular robotics system there are many challenges that need to be addressed.

From an algorithms perspective, aggregation must be performed in a fully decentralized manner without assumptions about terrain topology, inter-module communication, or reliability. The algorithms must be agnostic to the underlying hardware. As the number of modules in a self-assembly system can grow to the millions, our algorithms must also scale. In particular, we cannot rely on global information to make local decisions as limited communication capabilities make

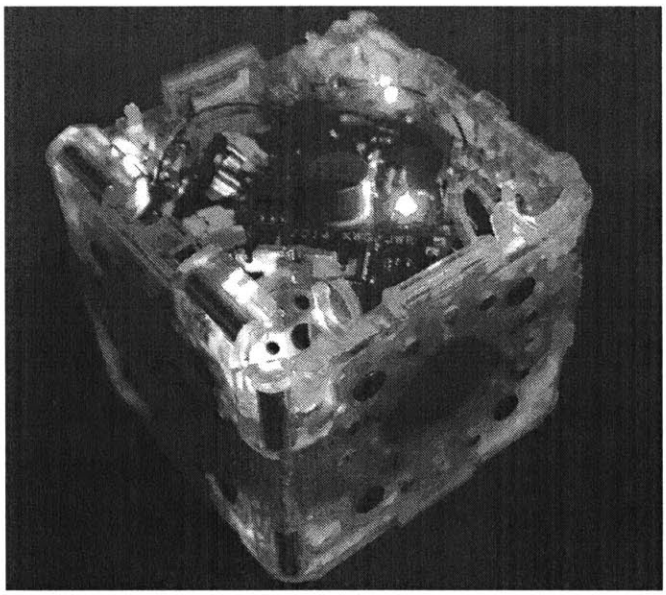


Figure 1-1: The 3D M-Blocks form a self-reconfigurable system of modules that are capable of individual movement through the use of an inertial actuator, and can bond to neighbors through connectors built from permanent magnets.

this infeasible. Once large scales are reached, defective modules also become an issue. The algorithms must be able to deal with unresponsive modules in a clean way. Furthermore, the algorithms must be both space and compute efficient due to hardware constraints.

Hardware limitations must also be taken into consideration. The limited processing power and memory storage available on the hardware is a limiting factor when designing algorithms for aggregation and reconfiguration. The modules cannot rely on external tools to function. Each module must have adaptable control to allow them to function in a wide range of environments.

This thesis provides a resolution for the problem of aggregation of the 3D M-Blocks.

1.2 Our Approach

Our approach improves upon existing hardware and algorithms. We provide theoretical guarantees for the general problem of aggregating a group of n robots in

$\mathbb{R}^d$ assuming only integrator dynamics. We prove convergence to a single aggregate assuming only knowledge of a single sensory function over the environment. We discuss a related problem of aggregation when the modules can only move along a low dimensional subspace of configuration space. Such problems appear when, for instance, turning the robot to a new orientation is expensive, requiring a significant time or power investment.

We provide provably correct algorithms for modular robot aggregation based on simple sensory information. For the discrete case in which modules are constrained to move along a lattice, we give a move and time optimal algorithm for the case in which the stimulus is a convex function over the environment. As an example, consider robots capable of sensing light intensity gradients spread in a closed convex environment. The light intensity values define a convex function over the domain, and the end goal is for the robots to converge to a position as close as possible to the light source. For the continuous case, we give a probabilistically complete algorithm that performs very well empirically.

We have developed the 3D M-Blocks to test our algorithms on a physical platform. The 3D M-Blocks as their name suggests are capable of locomotion across all three axes of a cube. The motion primitive in the 3D M-Blocks consists of a pivot by π or $\pi/2$ about an edge shared with a neighboring module or the ground plane (Figure 1-2). While pivoting, a module will sweep out a volume that is greater than the union of the volumes of the starting and final positions.

Figure 1-2: 3D M-Blocks pivoting moves.

We take inspiration from gradient descent algorithms. Our algorithm is a decentralized optimization with a cost function that tracks a signal in the environment that can be sensed by all robots. Leveraging information on directional signal strength, we show how our algorithm can provably achieve aggregation.

1.2.1 Outline of Technical Approach

Our algorithms are based on gradient methods. We assume the existence of a sensory stimulus that can be sensed by the robot. Specifically, we assume the robot can determine the gradient of the stimulus function either directly, or by estimating it in a neighborhood around its current position.

We present decentralized provably correct algorithms for aggregation under the following assumptions. First, we assume that the stimulus function is convex, and provide convergence guarantees by reducing to gradient descent. Second, we assume that the robots are capable of communicating with one another, and provide a control law and convergence guarantees by leveraging Lyapunov theory.

We further extend this model to allow for situations in which the robot has a restricted move set, and provide similar convergence guarantees in this case.

We apply these algorithms to the pivoting cube model. Leveraging the more structured environments on which lattice based systems operate, we can improve the convergence rates to linear if we assume the robots are restricted to move on a lattice. Finally, we show how we can efficiently perform aggregation when changing the direction of the robot is expensive.

1.3 Thesis Contributions

This thesis makes the following contributions:

1. A discussion of aggregation for robots following integrator dynamics. We provide optimal or nearly optimal control laws that drive modules together.

2. A provably correct decentralized algorithm for aggregating modules following the pivoting cube model, where the robot modules can move independently on a lattice of modules or on the ground. To the best of our knowledge, this is the first work that discusses general strategies for aggregation of modular robots.

3. Extensive simulation experiments for aggregation of randomly scattered modules following global environmental gradients using a decentralized algorithm.

4. Experiments demonstrating a hardware implementation of several different distributed aggregation algorithms using the 3D M-Blocks where light is the global external stimulus.

1.4 Thesis Outline

The remainder of this document is organized as follows. In Chapter 2, we survey related work, and discuss limitations with respect to the problem we address. In Chapter 3, we present the pivoting cube model of self-assembly, and provide background on the aggregation problem. We discuss the sister problem of reconfiguration for completeness. Finally, we give a detailed discussion of our current hardware implementation of the 3D M-Blocks hardware, including limitations.

Chapter 4 provides detailed theoretical results for the general aggregation problem in arbitrary dimensional configuration spaces. We give fast convergence guarantees under different assumptions (e.g. convex stimulus function, or unbounded communication radius), and discuss extensions of the theory to situations where the modules are restricted to move only along a subspace of the configuration space.

Chapter 5 applies the theoretical work of Chapter 4 to the pivoting cube model. We discuss how to overcome the limitations of the pivoting cube model to achieve the same fast convergence rates. In this Chapter, we also provide experimental data (both simulation and hardware) to back our theoretical claims).

Finally, Chapter 6 discusses the lessons we have learned while developing the theory and system, and provides suggestions for avenues of future research.

Chapter 2

RELATED WORK

Modular robotics (or programmable matter) define any system where the individual modules are capable of interacting with one another in non-trivial ways to form increasingly complex systems. Systems that we would describe as modular today have been around for decades. Perhaps the first such system was developed by Penrose in 1957 and 1959 [21, 22].

Primarily, the term modular robotics has come to imply two themes: (1) systems consist of several robots designed to function together, and (2) the systems address shape formation as the main problem to solve.

Our work builds upon the vast modular robotics literature, from the Penrose tiles, to the ground-breaking work of Fukuda et al [12, 13, 14], to the present day work on various self-reconfigurable models and platforms.

2.1 Lattice Modular Robotic Systems

We situate our work in the context of lattice-based modular robotic systems. Several models have emerged for self-reconfigurable cubic modular systems [1, 9, 15, 18, 25, 26, 28, 29]. The most prevalent of these is the sliding cube model (SCM) for which efficient and universal algorithms exist for reconfiguration. The SCM allows modules to slide along their neighbors to achieve motion. In particular, a common feature of the SCM is a move that slides a cube about two faces of one of

its neighbors to achieve a corner turn. This move proves difficult to implement in practice.

There are however, many attractive theoretical results in the SCM. When strong connectivity (face connectivity) is enforced, there exist $O(n^2)$ algorithms for reconfiguring arbitrary 2D shapes [3, 7]. For 3D configurations, algorithms have been developed that achieve reconfiguration if the start and goal configurations do not have topological holes [23]. This requirement is imposed as [23] assume that modules that have reached their position in the target configuration will no longer move. If we relax this constraint, we can achieve $O(n^2)$ reconfiguration centralized, and $O(n^3)$ distributed reconfiguration in 3D shapes [11]. In spite of this attractive body of work, we are not aware of any systems that implement the sliding cube model in 3D. Systems implementing a 2D version of the sliding cube model have been given by [2, 16, 37].

Because the sliding cube model is difficult to implement in practice, prompting a move to pivoting cube models of reconfiguration [25, 26]. In the pivoting cube model, modules pivot about one of their edges. Since the volume swept during pivoting is greater than the union of the start and end positions, we must be careful with translating SCM algorithms to this new model.

An $O(n^2)$ algorithm for reconfiguration was given by [5] assuming edge connectivity constraints. In [19], the authors considered pivoting hexagons in 2D and gave a $O(n^{5/2})$ algorithm for reconfiguration of a restricted set of configurations. An algorithm for reconfiguring pivoting cubes in both 2D and 3D under a strong connectivity assumption was given in [36], however the space of configurations that can be reconfigured is limited.

The pivoting cube model has had successful physical implementations in [26] for modules capable of motion in 2D, and refined in [25] for 3D motion.

While most modular robotic systems have focused on reconfiguration, the 3D M-Blocks are capable of motion in unstructured environments and while untethered. This allows us to talk of aggregation as the problem of localizing and connecting multiple untethered robots to a single connected aggregate. Recent work

in swarm robotics has demonstrated aggregation and reconfiguration in swarms of up to thousands of robots [27]. However, this work has been restricted to 2D structures, and it is unclear how to develop the model further to allow for 3D motion. An example of aggregation in 3D environments is given in [17], but the modules are not capable of self-locomotion and rely on wheeled robots to perform the aggregation. Approaches to aggregation that do not rely on external systems exist [42], but rely on on-board cameras which are still impractically large to be integrated with most modular robots.

2.2 Coverage Control

A problem that bears significant relevance to our work is that of coverage control. Briefly, coverage control represents broadly a set of algorithms for optimizing the location of multiple robots to maximize coverage with respect to a function defined over the environment. For example, we might wish to communicate through ad-hoc wireless networks, in which case we would want to maximize the signal strength obtained through the network. Such work is situated at the intersection of multi-robot path planning [4, 6] and locational optimization [10, 38].

Distributed coverage control has efficient solutions when the distribution of the stimulus over the environment is assumed to be known *a priori* by all robots [8, 24, 30]. This assumption was relaxed first in [31], and improved upon in future work [32, 33].

Problems in distributed coverage control share many similarities with aggregation: the algorithms must be distributed, and very frequently the robots have only limited sensing and localization capabilities. We point specifically to [31, 32] as examples of distributed coverage control algorithms that share features with our aggregation algorithms.

The problem we solve is fundamentally different, however. We wish to control the final shape of the robots and their relation to one another. Specifically, we aim to minimize the distance between robots (thus form an aggregate), without

considering the fitness of the robots' position with respect to a sensory function as is the case with coverage control algorithms.

Chapter 3

BACKGROUND

Table 3.1: List of Symbols from Chapters 3–5

Symbol	Definition
n	Number of robots.
$\mathbf{p}_i$	Position of the i^{th} robot.
$\dot{\mathbf{p}}_i$	Velocity of the i^{th} robot.
$\mathbf{p}_L$	Position of the stimulus source.
u_i	Control input of i^{th} robot.
$\mathcal{N}(i)$	Neighbors (in communication radius) of i^{th} robot.
$\mathcal{W}$	Configuration space of the robots.
ϕ	Sensory function.
J	Global cost function.
I_j	Intensity value on the j^{th} face of a robot.
$\mathbf{n_j}$	Outward facing normal of the j^{th} face of a robot.
$\mathcal{C}_L$	Circle centered at projection of stimulus source.
$\mathcal{C}_x$	Circle centered at point x.

We begin by describing the underlying assumptions in this thesis in more detail. Specifically, we describe the pivoting cube model: the motion model that our hardware implementation is based on. We describe the problems of aggregation and reconfiguration to provide intuition into the theoretical contributions that follow. Finally, we conclude this chapter with a description of the 3D M-Blocks hardware and its limitations.

Figure 3-1: Admissible pivoting moves.

3.1 Pivoting Cube Model

As its name suggests, the pivoting cube model comprises cubic modules that are move by pivoting about their edges. In the sliding cube model a move is possible if the start and end positions are adjacent, and the end position is unoccupied. By contrast, in the pivoting cube model, the volume swept by a module during a move must also be unoccupied since a cube is longer along a diagonal. This greatly restricts the set of moves allowed.

In our model, a cube pivots the maximum possible before it comes into face contact with another module. Two modules are connected if they share a face.

The pivoting cube model is attractive because it implements a realistic move set that can easily be translated into a hardware implementation. For example, the corner step move shown in Figure 3-1b is a pivot by π radians, while in the sliding cube model, a similar move is unrealistic.

We define the ground plane $y = 0$ and restrict ourselves to configurations $C \in \mathrm{R}^3_{y\geq 0}$.

3.2 Aggregation

Aggregation is the task of connecting modules that are initially scattered in an environment into one single connected whole. This task is easy when the modules are capable of relative localization through sensory input (e.g. message passing), but significantly more difficult when such information is not readily available.

In the 3D M-Blocks (see Figure 3-2 and also 3.4), communication is only guaranteed in point-to-point contact. Therefore, instead of relying on communication,

we formulate the problem as an optimization over a sensory stimulus and use gradient ascent strategies to perform aggregation. Each face of a 3D M-Block has an ambient light sensor that outputs a scalar value in a bounded range. These readings can be used to obtain a direction of steepest increase and to steer the module towards the aggregate.

We generalize this setting as follows. Let $\mathcal{W}$ be a bounded two dimensional region, and let $\phi : \mathcal{W} \rightarrow \mathrm{R}$ be a sensory function that returns the intensity of the stimulus at each point in $\mathcal{W}$. For aggregation of n robots into a single connected component, we require that ϕ be convex. Let $\mathbf{p_i} : [0, \mathrm{inf}) \rightarrow \mathcal{W}$ be the position function of each module i for $1 \leq i \leq n$.

We distinguish between two problems:

Lattice Aggregation. The modules are restricted to lie on a lattice embedded in $\mathcal{W}$. The modules are capable of motion along the x and y axes of the lattice.

Free Space Aggregation. To better model the actual behaviour of the 3D M-Blocks, we endow each module with a direction vector $\mathbf{u_i(t)}$. Modules can move along the direction vector $\mathbf{u_i}(t)$, or they can attempt plane changes which reorient the module along a different direction vector $\mathbf{u_i}'(t)$ while also adding random noise to its position vector $\mathbf{p_i}'(t) = \mathbf{p_i}(t) + \xi$ where ξ is drawn from a distribution over $\mathcal{W}$. For probabilistic completeness, we require that ξ have non-zero density almost everywhere in $\mathcal{W}$.

3.3 Reconfiguration

Once aggregation has been achieved and the modules are in a single connected component, we turn our attention to the problem of reconfiguration.

A configuration C is a set of unit cube modules on a cubic lattice. Let $|C|$ be the number of modules in configuration $|C|$. We will identify a module with an index i into this set. We can assign a coordinate system such that each module i is at integer coordinates (x_i, y_i, z_i).

Two modules i and j are connected if they share a face. We can create a con-

figuration graph $G = (V, E)$ with vertices $V = \{i | i \text{ is a module in } C\}$ and edges $E = \{(i, j) | i, j \text{ are connected in } C\}$. The configuration C is connected if the underlying graph G is connected. As a surface in R^3, C has a boundary ∂C, and a complement $\mathrm{R}^3 \setminus C$.

The problem of reconfiguration can be formulated as follows. Given configurations C_s and C_f where $|C_s| = |C_f|$, find a set of feasible configurations $(C_s, C_1, \ldots, C_T)$ such that the following conditions hold:

- *Termination*: $C_T = C_f$.
- *Validity*: For each t, C_t can be obtained from C_{t-1} by pivoting a set of modules in C_{t-1}.
- *Collision Avoidance*: For each set of moves, the volumes swept by the modules during execution must not overlap.
- *Connectivity*: For each t, the configuration graph of C_t is connected.

In addition, we can distinguish between solutions by counting the number of moves required to achieve reconfiguration. Formally, a move is any single pivot by a single module. For instance, if several modules are required to move to achieve reconfiguration, we count one move for each. A solution that requires fewer moves overall is better than one that requires more moves. For practical purposes, we would also want to distinguish between moves based on their empirical difficulty. This can be achieved by assigning non-uniform weight to each move and preferring moves with smaller cost.

3.4 Hardware

The 3D M-Blocks are modular self-reconfigurable robots which use inertial actuation to reconfigure by pivoting about the edges of the modules (Figure 3-2). 3D M-Blocks move by applying a torque generated by an internal flywheel. To reach

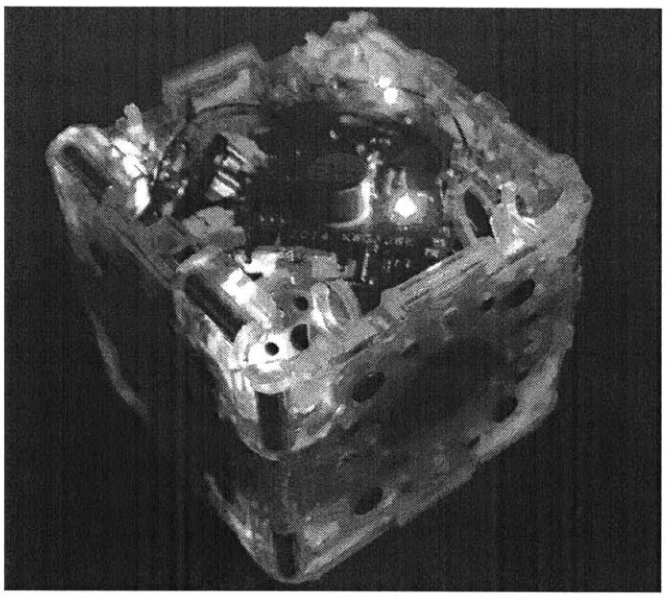


Figure 3-2: Photo showing a 3D M-Block with current hardware and electronics.

very high torque impulses, we employ a mechanical braking system that can generate torques in excess of 2.5Nm in a 10ms window (Figure 3-3).

An internal mechanism allows the flywheel to realign to a set of orthogonal axes X, Y, Z. Each face has eight face magnets and four edge magnets. All faces are rotationally symmetric making the 3D M-Blocks symmetric about three orthogonal planes.

While designed to work best on a structured lattice formed of other modules, the internal actuation mechanism allows the modules to move freely both individually and as assemblies.

Each face of the 3D M-Blocks contains a microprocessor that manages an ambient light sensor, two RGB LEDs and IR communication electronics. One face contains an accelerometer to allow the module to determine its orientation with respect to the central actuator. The modules are capable of communication through an infrared light system. Each face can transmit and receive IR messages on a range of up to 200cm.

When transmitting messages between adjacent modules, the IR light will refract through the received face and reflect on the interior of the body, messages

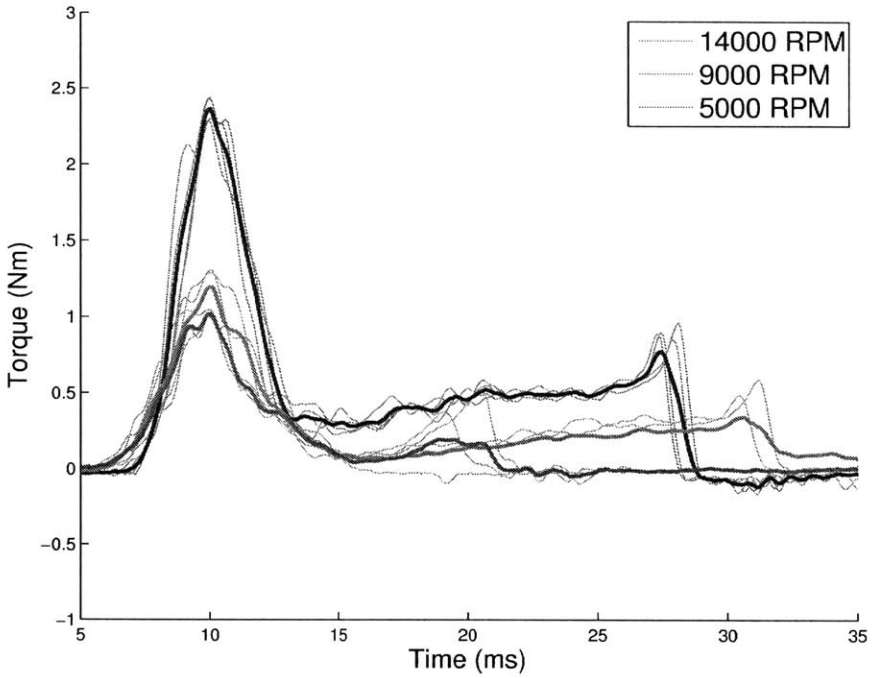


Figure 3-3: Torque generated by applying the mechanical brake. Plot shows torque (in Nm) against RPM of the flywheel when the brake is applied. Several experiment runs (shown in faint colors) are averaged and the mean is displayed in solid colors.

need to be signed with an initial pattern that can only be detected by sampling the ambient light sensor of the adjacent face.

Chapter 4

DECENTRALIZED AGGREGATION WITH SENSORY STIMULI

In this chapter, 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.

4.1 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 $\phi : W \to \mathrm{R}$, 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 ith robot, and $\dot{\mathbf{p}}_i$ is the usual velocity vector.

This assumption is common in the coverage control literature[8, 24, 31, 32]. We further assume that each robot has knowledge of the gradient of the sensory function $\nabla\phi$ 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 have to have any knowledge of surrounding robots.

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

$$J_i(\mathbf{p}_{-i}) = \frac{1}{2}\sum_{j\neq i} \|\mathbf{p}_i - \mathbf{p}_j\|_2^2 \tag{4.1}$$

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

$$J(\mathbf{p}_1, \ldots, \mathbf{p}_n) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \|\mathbf{p}_i - \mathbf{p}_j\|^2. \tag{4.2}$$

Equations (4.1) and (4.2) cannot be optimized directly without knowledge of the position of every robot.

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

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

for any point $\mathbf{q} \in \mathcal{W}$. Finding a control law that drives all robots to the same position $\mathbf{q}$ will also minimize equation (4.1) for all robots.

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 al-

ways be errors in the position estimation and control law of the robot. If ϕ were to vary unboundedly, small errors can compound rapidly. Second, as we shall see soon, there is a strong correlation between aggregation of robots, and the gradient descent algorithm for finding local minima of functions. To guarantee convergence of the gradient descent procedure, we require ϕ Lipschitz continuous.

4.2 Decentralized Aggregation Control

We want 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 like a stringent requirement, 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.

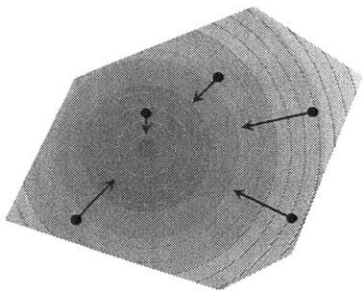


Figure 4-1: Aggregation of robots distributed in a convex region using sensory information.

This motivates the following control law:

$$u_i = -\alpha \nabla \phi(\mathbf{p}_i) \tag{4.3}$$

where α is a line search parameter.

Control law 4.3 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, 4.3 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 (4.3) 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 (4.3) does not achieve this for functions that are not globally convex.

To motivate the second control law, consider again the cost function (4.2). 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:

$$u_i = -\mathbf{p}_i + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\mathbf{p}_j. \tag{4.4}$$

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 (4.4) 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 4.2.1. *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 used the definition of Lyapunov stability. We will not repeat it, but note that intuitively Lyapunov stability is a bound on how far away a system can get from the 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 [34]. We propose the following Lyapunov function:

$$V_i = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \|\mathbf{p}_i - \mathbf{p}_j\|^2 \tag{4.5}$$

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

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

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.

4.3 Stochastic Control

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

Definition 4.3.1. *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 the M-Blocks, and expanded further in chapter 5.

The conditions on V are that it be connected and closed, 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. If $e_1, \ldots, e_m$ are the basis vectors of $\mathbb{R}^m$, the last condition is equivalent to the fact that the linear map $\frac{\partial \mathbf{u}}{\partial \mathbf{v}}$ is an isomorphism.

We consider again the problem of driving the modules together. We wish to minimize the global cost (4.2) 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 (4.3) to

$$u_i = \begin{cases} -\beta \nabla(\phi|V)(\mathbf{p}_i), & \text{if} \quad \nabla(\phi|V)(\mathbf{p}_i) \neq 0, \\ V \sim \mathcal{U}(\mathbf{p}_i, \mathcal{W}), & \text{otherwise.} \end{cases} \tag{4.6}$$

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

1: **repeat**
2: **if** there is a direction of decrease in the stimulus **then**
3: set $u_i = -\beta \nabla(\phi|V)(\mathbf{p}_i)$
4: **else**
5: sample new subspace in which to move
6: **until** good enough solution is reached

Note that Algorithm 1 is not guaranteed to converge. However, under certain assumptions on the sampling procedure, we can provide convergence guarantees of (4.6) 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 4.3.1. *If V is sampled uniformly over the set of m-dimensional flats, and ϕ is convex over $\mathcal{W}$, then control law (4.6) converges.*

We will make use of the monotone convergence theorem which we restate here in a simple form (see [35]):

Theorem 4.3.2. *(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 4.3.1.

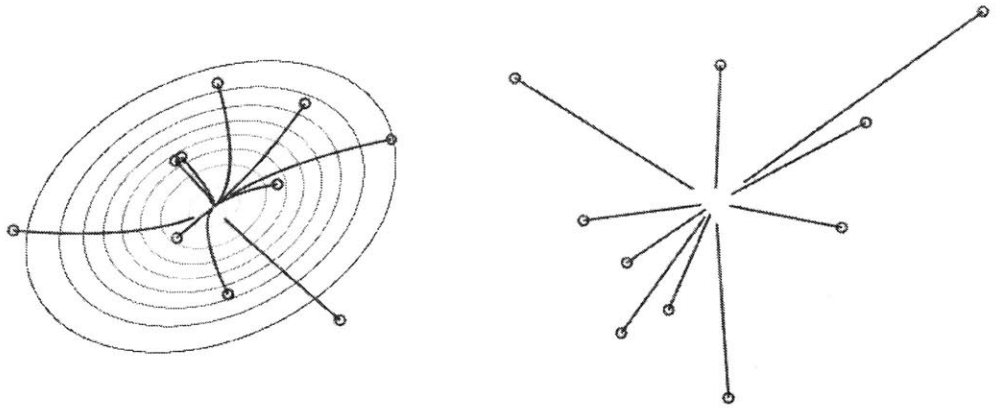
Proof. Let the $\mathbf{p}_i^1, \ldots, \mathbf{p}_i^n$ be the sequence of points for which $\nabla\phi(\mathbf{p}_i^j) = 0$. By the convexity of ϕ, we have $\phi(\mathbf{p}_i^1) \geq \phi(\mathbf{p}_i^2) \geq \ldots \geq \phi(\mathbf{p}_i^n)$. This series is monotone and bounded since ϕ is bounded. Applying the monotone convergence theorem, we can conclude that control law (4.6) converges. □

We conclude this chapter with a simulation result. For the simulation, we assumed there $n = 10$ 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 (4.3) and (4.4) in two simulation experiments. For control law (4.3), we fix $\alpha = 1$ as the step size, and perform gradient ascent. Simulation runs are shown in Fig. 4-2. Both control laws converge to a single point, thus achieving aggregation. We note here that convergence rate depends on the choice



(a) (b)

Figure 4-2: Simulation of aggregation control laws. (a) Control law (4.3); 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 (4.4); the robots' initial positions are shown as circles with the trajectories shown as red lines.

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. 4-3). 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 this chapter, we have looked at the problem of aggregation in a generic fashion, assuming integrator dynamics on the robots. In the chapters that follow, we explore a specific example of aggregation in the case of the 3D M-Blocks.

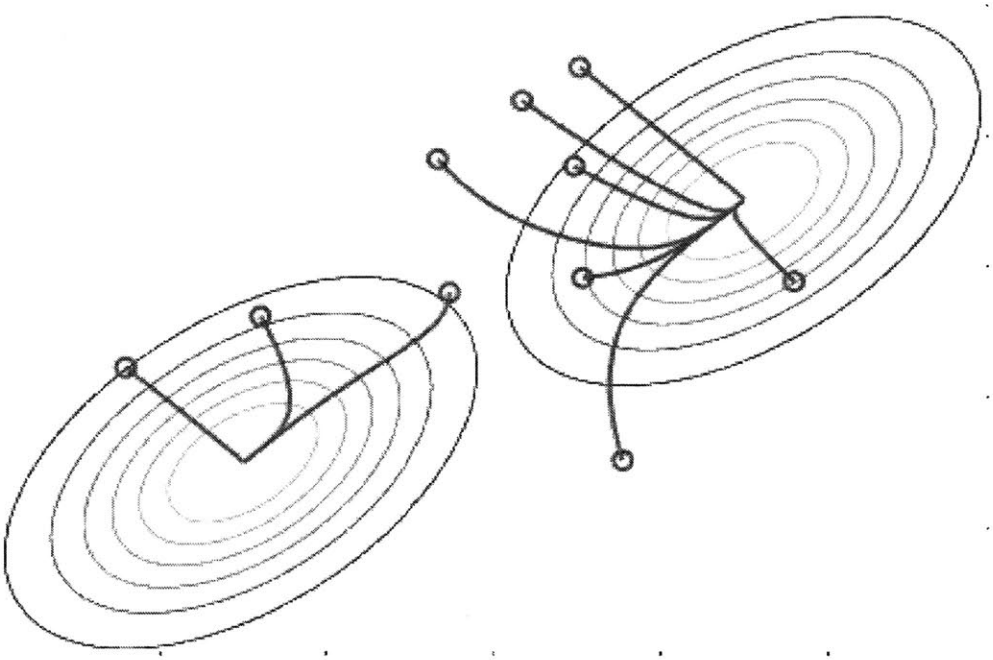


Figure 4-3: A single aggregate is impossible in this scenario. The robots are distributed in an environment with two local maxima and have very limited communication capabilities. As can be observed, two aggregates are formed. Without prior knowledge of the number of modules, such scenarios cannot be resolved into a single aggregate.

Chapter 5

AGGREGATION OF MODULAR ROBOTS IN THE PIVOTING CUBE MODEL

Following Chapter 4, we implement the algorithms described in the pivoting cube model. The restrictions presented by the pivoting cube model are dealt with new theoretical results. We showcase an example arising from our hardware implementation where the stochastic control described in the previous chapter is important. We conclude this section with experimental results including simulations and hardware experiments of aggregation.

5.1 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

Figure 5-1: Pivoting moves.

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:

$$\min \max_i \|\mathbf{p}_i(t_f) - \mathbf{p}_L\|. \tag{5.1}$$

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:

$$\frac{1}{|\{I_j | I_j \neq 0\}|} \sum_{I_j \neq 0} I_j \tag{5.2}$$

Equation 5.2 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 5.2 reduces to $\max_j I_j$.

5.1.1 Driving Modules Towards the Maximum Direction of the Stimulus

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. 5-2).

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 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 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 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 5.1.1. *For a cube with initial position $p_i(0)$ 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

$$I_{\mathbf{n_j}} \propto \mathbf{n_j} \cdot \mathbf{d_j}. \tag{5.3}$$

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 $p_i(0)$, 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 5-2 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*.

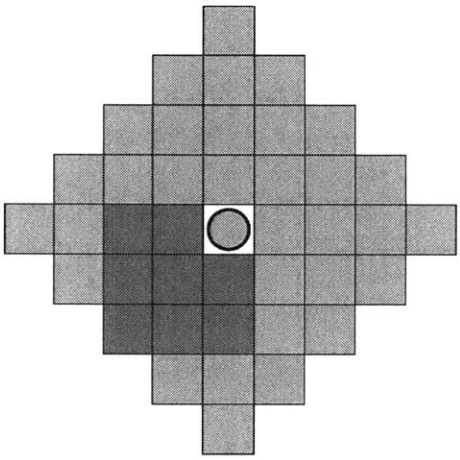


Figure 5-2: Coordinates at which the blue module can reside. In green are positions that are *strongly admissible*, while *weakly admissible* positions are in yellow. The circle represents the projection of the stimulus source onto the plane.

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.

5.1.2 Driving Modules Together

We now show that Algorithm 3 converges to the global maximum of Equation 5.2 for one module.

We can prove the following:

Theorem 5.1.2. *Algorithm 3 converges to a single aggregate. Moreover, the following two properties will be satisfied:*

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.

5.1.3 Stimulus Tracking in $\mathbb{R}^2$

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 (Fig. 5-3).

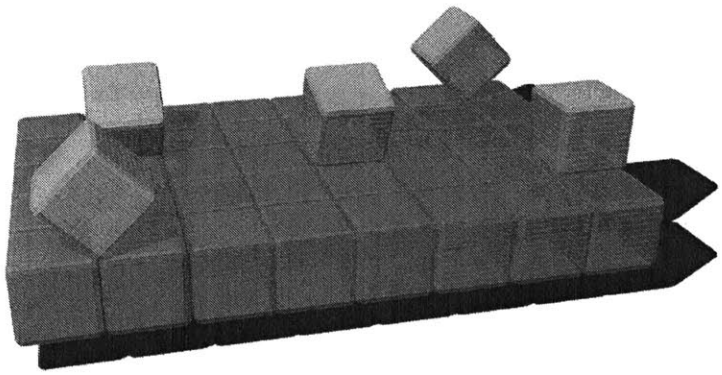


Figure 5-3: Modules moving on fixed scaffolding. Green modules above are mobile, while the orange modules forming the bottom layer are not capable of motion, but form a lattice on which the former move. In this scenario, algorithms (5.1)(5.2) can be used as the scaffolding acts as a lattice.

However, if there is no lattice base, the M-Blocks develop a very high torque when performing a plane change. The forces generated during this maneuver will cause the M-Block to shake violently and lose its initial orientation. Thus, we cannot rely on the lattice assumption and must develop algorithms to deal with the

We relax our requirement of deterministic convergence and give a probabilistically complete algorithm that at every step with high probability will drive the 3D M-Block closer to the stimulus source.

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 uniformly from a circle centred at $\mathbf{x}'$ with radius R_x (write $\mathcal{C}_x$ for this circle), and sample a new direction vector $\mathbf{u}$ uniformly at random (equivalent to sampling the slope of the line that passes through the new point).

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. Recall from Section 4.3 that this procedure is guaranteed to converge eventually. In what follows, we provide stronger guarantees and empirical results for the particular case of the M-Blocks.

We call one complete execution of 1 and 2 (moving along line and sampling new position) a single step. For the very first jump we assume the first step has been performed.

We care about the expected number of steps needed for the M-Block to reach the goal. The probability that the goal can be reached in one step is given by:

$$\Pr[\text{one step}] = \frac{|\mathcal{C}_L \cap \mathcal{C}_x|}{\pi R_c^2} + \left(1 - \frac{|\mathcal{C}_L \cap \mathcal{C}_x|}{\pi R_c^2}\right) \frac{2}{\iint\limits_S \mathrm{d}S} \iint\limits_S \frac{\arcsin \frac{\|\mathbf{x}_S - \mathbf{x}_L\|_2}{R_L}}{\pi} \mathrm{d}S \quad (5.4)$$

where $S = \mathcal{C}_x \setminus \mathcal{C}_L$.

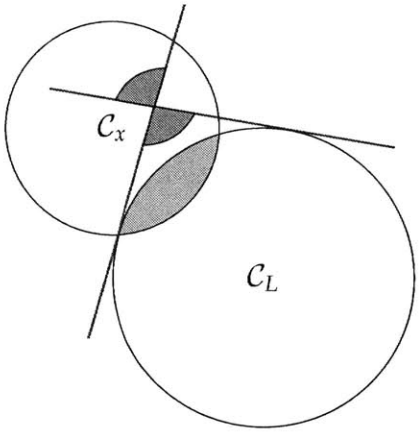


Figure 5-4: Geometric interpretation of eq. 5.4. Either the new position is in the grey intersection, or averaged over all other positions, the new direction vector is within the blue angles.

The geometric interpretation of eq. 5.4 is that the goal is reached if either the new random position is within $\mathcal{C}_L$ (the first term in eq. 5.4), or the new direction vector will move the cube into $\mathcal{C}_L$. This only happens if the direction vector (or its inverse) is within the two tangents from the new point to the circle.

A new position $\mathbf{x}'$ is better than $\mathbf{x}$ if $\|\mathbf{x}' - \mathbf{x}_L\|_2 \leq \|\mathbf{x} - \mathbf{x}_L\|_2$. Notice that this is the same as the problem of reaching the goal in one step with R_L replaced by $\|\mathbf{x} - \mathbf{x}_L\|_2$. We are trying to move into the circle centred at $\mathbf{x}_L$ with radius $\|\mathbf{x} - \mathbf{x}_L\|_2$ (the old distance). Call this circle $\mathcal{C}_I$ and its radius R_I. The probability of improving the position is then given by

$$\Pr[\text{improve}] = \Pr[\text{one step}] + (1 - \Pr[\text{one step}]) \cdot \left(\frac{|\mathcal{C}_I \cap \mathcal{C}_x|}{\pi R_c^2} + \left(1 - \frac{|\mathcal{C}_I \cap \mathcal{C}_x|}{\pi R_c^2}\right) \frac{2}{\iint_S \mathrm{d}S} \iint_S \frac{\arcsin \frac{\|\mathbf{x}_S - \mathbf{x}_L\|_2}{R_I}}{\pi} \mathrm{d}S \right) \tag{5.5}$$

This is much larger than $\Pr[\text{one step}]$ as the function is increasing with distance when

$$C_x \cap C_L = \emptyset.$$

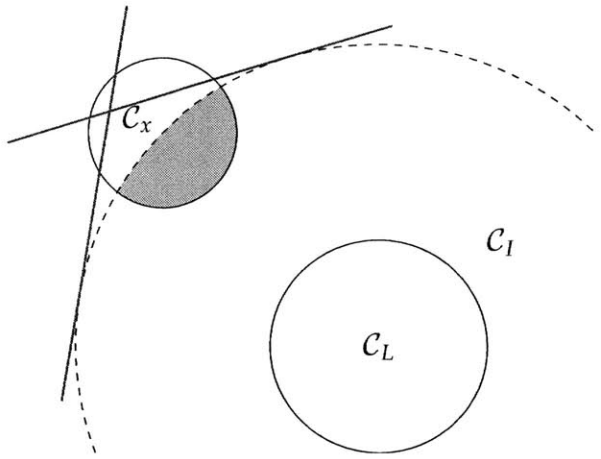


Figure 5-5: Geometric interpretation of eq. 5.5.

Since the surface integral in both equations is difficult to evaluate, we lower bound the probabilities by assuming all points within the regions being intergrated over are distance

$$\|\mathbf{x} - \mathbf{x}_L\|_2 + R_x$$

away from the stimulus source.

We now establish the following results.

Proposition 5.1.1. Pr[*one step*] > 0 *if the space is convex and bounded.*

If we show that Proposition 5.1.1 is true, then we have effectively shown probabilistic convergence of our algorithm, as everywhere on the set, the probability that we will converge to the optimum solution is non-zero.

We now proceed with the proof:

Proof. If C_L and C_x intersect, then the first term is strictly positive. Otherwise $\|\mathbf{x}_s - \mathbf{x}_L\|_2 > R_L$ and the integral term will be strictly positive. □

We note here that the algorithm also extends for bounded sets that are not necessarily convex in R^2 but on which the stimulus function ϕ is concave. The modification necessary is to replace the walk along $\mathbf{u}$ or $-\mathbf{u}$ with a walk along the gradient $\nabla\phi$.

If ϕ is not concave, we can only guarantee convergence to local maxima of the stimulus function. As the modules are initially randomly distributed in the environment, a single aggregate is no longer a feasible end result.

Proposition 5.1.1 establishes the following:

Corollary 5.1.1. *The algorithm is probabilistically complete on bounded sets assuming a concave stimulus function ϕ.*

A simple example is a point light source (concave function) in the middle of a room (bounded set).

Proposition 5.1.2. $\Pr[\textit{improve}]$ *is strictly increasing with* $\|\mathbf{x} - \mathbf{x}_L\|_2$ *when* $\mathcal{C}_x \cap \mathcal{C}_L = \emptyset$.

Proof. This can be proved formally using the area formula for the intersection of two circles, but a simple argument is to consider the intersection area as the distance between the stimulus source and the current position grows.

The larger $\|\mathbf{x} - \mathbf{x}_L\|_2$, the more the intersection resembles a semicircle. The first term in Pr[improve] thus increases with distance. The second term will always be strictly smaller than

$$1 - \frac{|\mathcal{C}_I \cap \mathcal{C}_x|}{\pi R_c^2}$$

which completes the proof. □

Intuitively, what Proposition 5.1.2 implies is that the farther away from the stimulus source the module is, the more likely it is that it will be able to improve its position. On the other hand, if the module is close to the stimulus source, it is less likely to be able to improve its position. But since the probability of improvement only becomes negligible when very close to the stimulus, aggregation is still achievable.

5.2 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 used the light

intensity from a point light source located above the modules.

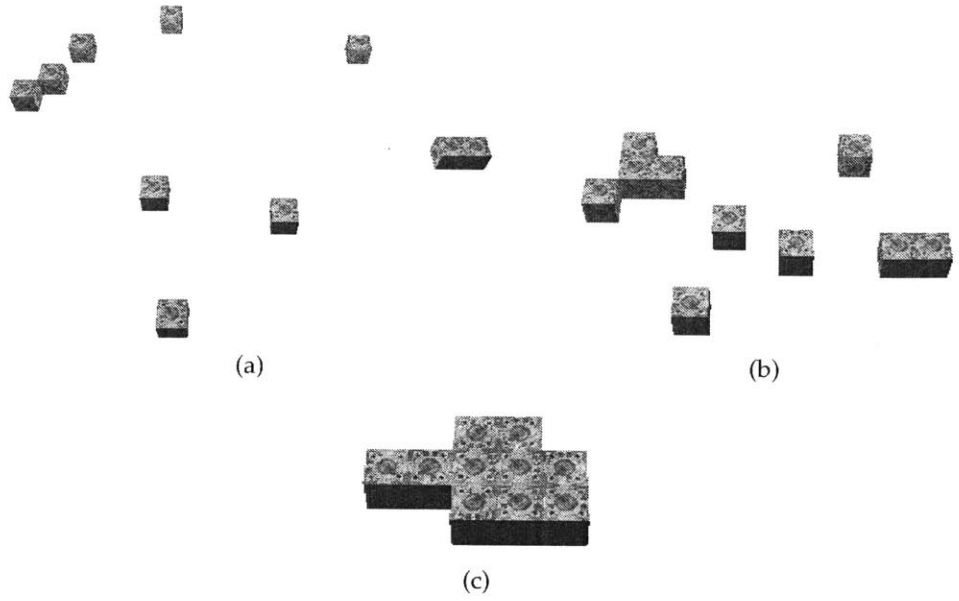


Figure 5-6: One simulation run with 10 modules. (a) Starting configuration with modules scattered randomly. (b) Halfway through execution. (c) After aggregation.

5.2.1 Simulation Results

We have written a simulation using C++ and OpenGL. To compute the light intensity on a face, we used a flat shading model and assumed that light intensity is proportional to the inverse square of the distance from the light source to the surface.

Figure 5-6 shows a typical simulation run with 10 modules using Algorithm 3. The initial configuration is shown in Figure 5-6a. The final aggregate is shown in Figure 5-6c. We ran 1000 such simulations using up to 100 modules. When running a simulation with 100 modules the main bottleneck is in rendering, as the algorithm requires only $O(1)$ operations per module to find the next move. All simulation runs converged to the optimum configuration described by Equation 5.2 where no module can move closer to the projection of the light source on the plane.

To test aggregation in free space, we implemented a Monte Carlo simulation of the algorithm. We averaged 50 simulation runs and plotted the expected number of moves until convergence on a 10×10 grid with $R_L = 1$, and $R_x = 1$ (Fig. 5-7). 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.

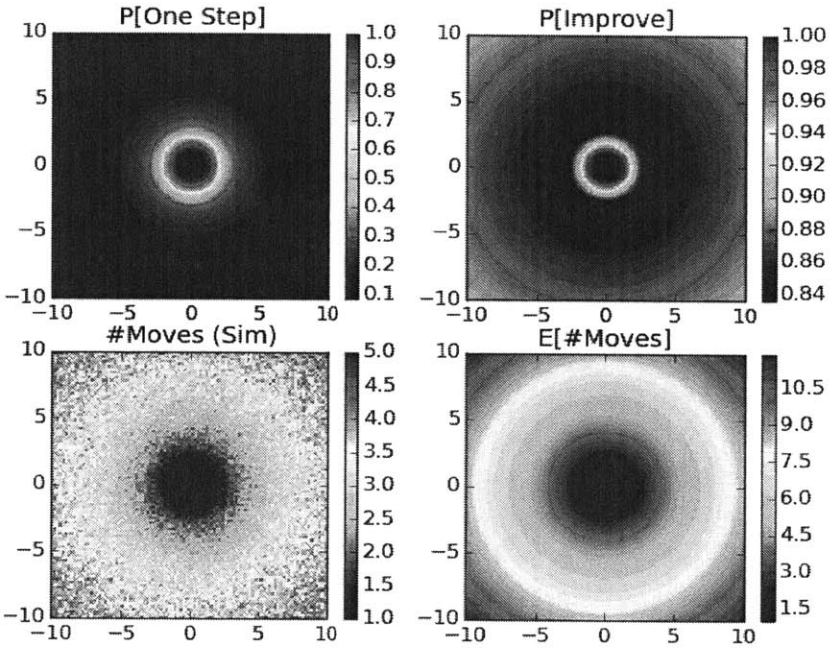


Figure 5-7: Simulating aggregation in free space. The top plots represent equations 5.4 and 5.5. Notice how Pr[improve] is increasing with distance. The bottom left plot is the result of the Monte Carlo simulation, while the right is the expected number of moves until convergence obtained by considering a geometric distribution with $p = \Pr[\text{one step}]$ at each point.

We can obtain a coarse upper bound on the expected number of moves by assuming a module is fixed in place until it samples a direction vector that is aligned with the stimulus source (the fact that $\Pr[\text{improve}] > 0.5$ guarantees this will be a lower bound). This generates a geometric distribution at each point in R^2, and we can obtain the expected number of tries until success which establishes an upper

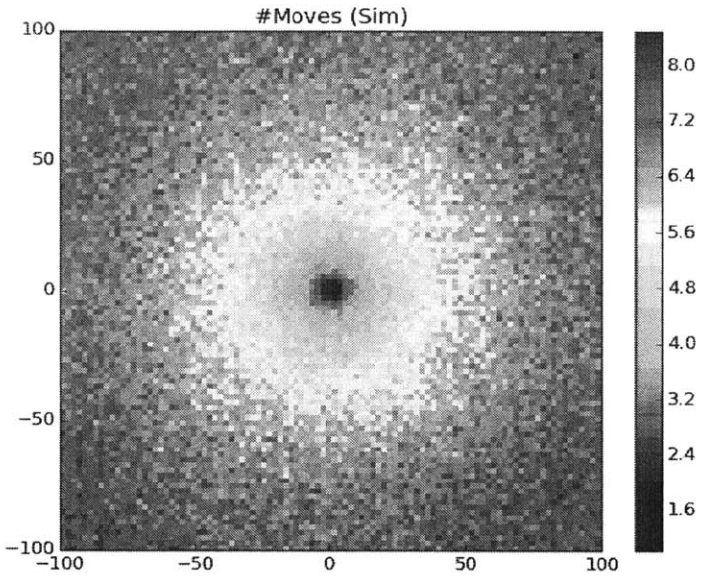


Figure 5-8: Expected number of moves until convergence in a 100×100 grid averaged over 100 runs.

bound.

However, empirically this upper bound is very rough. In Fig. 5-8, we can see that the expected number of moves until convergence grows roughly as the square root of $\|\mathbf{x} - \mathbf{x}_L\|_2$, while the lower bound we establish grows linearly with $\|\mathbf{x} - \mathbf{x}_L\|_2$.

5.2.2 Hardware Results

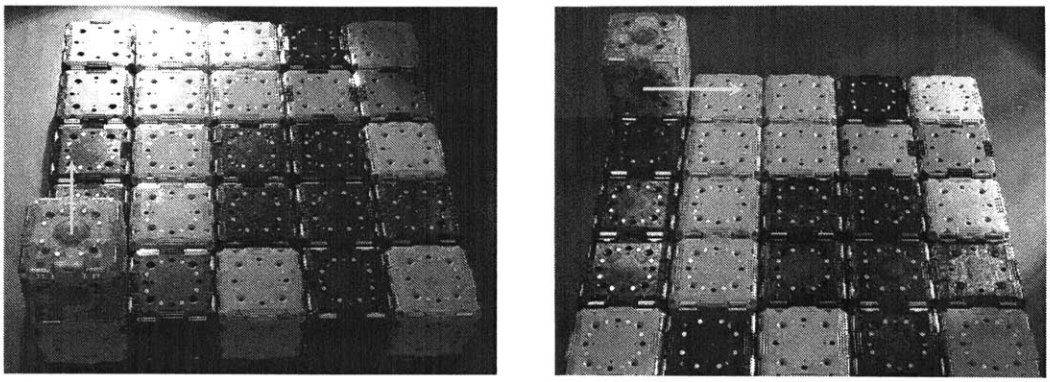
To implement the algorithms on the 3D M-Blocks, each module had to be capable of performing two motion primitives: a traverse move that would move the module forward or backward along its plane of orientation, and a plane changing operation to realign the flywheel with a different orientation. To verify the feasibility of the experiments, we first tested these two primitives. We measured both reliability and the time between the module receiving the command and execution. The results are shown in Table 5.1. Plane changing is reliable as it retries until the desired plane is reached; on average, it takes 3.2 attempts, and each attempt takes

approximately 10s. The inertial actuation move has low variance (<0.1s), but plane changing has high variance: over 50 runs, the most attempts one module required was 13, but the mean number of retries was 3.

Table 5.1: Reliability and time required to execute the two motion primitives.

Primitive	Reliability	Mean Time
Traverse	79/80	3.36s
Change Plane	50/50	31.5s

We verified the algorithm would track a moving light source by using two light sources orthogonal to one another, and switching between them. We performed this test 10 times on a 5x5 regular lattice (see Figure 5-6). This involved 80 traversal moves, and 10 plane changing operations. During the test, only two traversals failed due to a hardware issue that caused the IMU readings to lag, and caused the module to have an incorrect notion of which direction was forward. As computing the next move requires $O(1)$ time and memory, this shows our algorithm is both efficient and practical.



(a) (b)

Figure 5-9: Experimental setup to verify that Algorithm 3 works on our hardware platform. Simulated lattice aggregation can be seen in Fig. 5-6. (a) Initial configuration and expected direction of motion. (b) Intermediate configuration with expected direction of motion; here the module needs to perform a plane change.

Finally, to test aggregation, we arranged six modules in free space and five on a lattice and ran Algorithm 3. Due to the magnetic forces acting on a module when

it is connected to another module, once a two module configuration is formed it cannot be broken by simple traversal moves. For this reason, we only run function STEP in Algorithm 3 a finite number of times and report the number of aggregates, the size of the largest aggregate formed, and the number of modules that were not part of a larger aggregate.

Throughout ten experiments in free space running 5 steps of Algorithm 3, the modules formed an average of two aggregates, though there were two runs where only one was formed. The largest size of any single aggregate was three modules (half of the initial six). The maximum number of individual cubes observed was four. We discovered that a simple method of controlling two connected modules in free space is to use one as steering module by orienting its flywheel parallel to the ground plane and the other as a driving module. The steering module can then be used to slide the configuration across the ground, while the other module can perform two traverse moves to ensure that no change plane operations need to be performed (see Figure 5-10). This allows almost perfect control over the aggregate. This aggregate can then be used to "pick up" single modules.

Throughout five experiments on the lattice, the modules formed into either one or two aggregates. In three of the five runs, the modules formed a single aggregate located under the light source. While the algorithm guarantees that every module will be part of an aggregate, the large number of plane changing operations required greatly increase the time required for the algorithm to converge. Moreover, a two module aggregate on a lattice only allows motion along one axis as opposed the equivalent in free space.

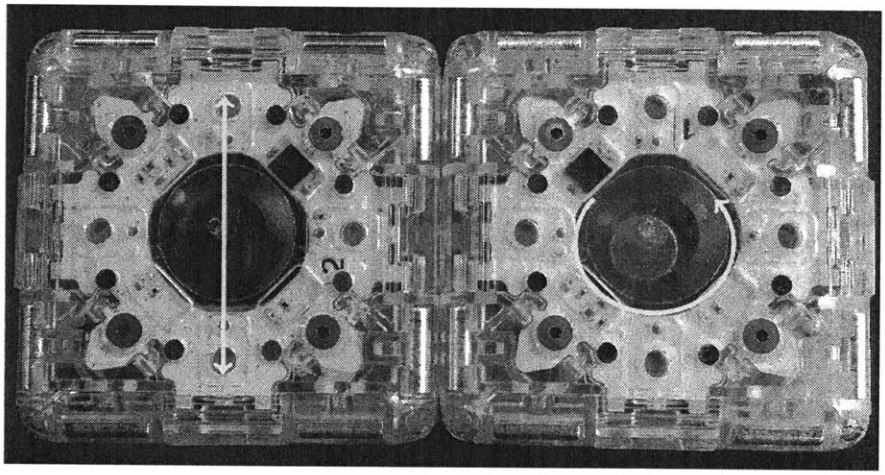


Figure 5-10: Control in free space using two modules. The right module can rotate the two cube configuration about its center by accelerating and braking the flywheel. The left module can move forward and backward: performing any two such moves will maintain the configuration.

Chapter 6

CONCLUSIONS

Our results in this thesis are twofold.

First, we have shown to our knowledge the first discussion of aggregation of multiple robots under the action of stimuli. We have provided convergence guarantees and showcased algorithms for restricted versions of the general problem. The algorithms, based on gradient descent, converge rapidly to a locally optimal solution. Under certain assumptions, we have shown that convergence to the global optimal is not only possible, but guaranteed.

Second, we have presented a decentralized control algorithm for the aggregation of modular robots following the pivoting cube model. We derived an algorithm that tracks a sensory stimulus (in our case a light source) and gave provable guarantees for its correctness and convergence. The control scheme was demonstrated in simulation. We also implemented the scheme on the 3D M-Blocks, our hardware platform for the pivoting cube model, which demonstrates the practicality of the scheme. To our knowledge, this is the first time aggregation, a necessary first step in many of the common use cases of modular robotics, has been discussed at length.

A similar approach can be used to extend this work to other application domains. We highlight the possibility of using the 3D M-Blocks to perform coverage control, or in the creation of sensor networks. This would require a significantly more robust and efficient communication scheme between modules.

6.1 Lessons Learned

We considered aggregation under very strict assumptions, and proved that good solutions can be obtained even with very primitive communication and sensing capabilities. Unfortunately, there are situations in which a single aggregate is impossible to obtain. For instance, if the robots can only communicate with each other when very close, the stimulus function has two maxima well separated, and the robots are initially distributed uniformly in free space, then a single aggregate cannot be obtained. Our approaches rely on either the convexity of the stimulus function, or long range communication capabilities.

If neither are present, it is possible that exploration/exploitation strategies will perform better than what we outlined in this thesis. There is an added cost to performing exploration/exploitation. We lose theoretical convergence guarantees unless we assume that the robots know *a priori* how many other robots are present in the environment. As modular robots need to be robust against individual failures, this requirement may be untenable. Second, convergence will be significantly slower due to the time spent in exploration stages.

On the practical side, future work is required for the system and algorithms to become practical for real world use. One issue is that of scale. The current 3D M-Blocks are impractically large for many of the applications modular robots are desired for. Second, the algorithms must be tested in real world scenarios with significantly more noisy stimuli than in closed controlled lab scenarios.

6.2 Future Work

There are several interesting avenues for future research. A sister problem of aggregation is reconfiguration. That is, once modules have been aggregated into a single connected component, what moves should the modules perform to change from one static configuration to another. Previous work [36] has addressed this issue in the pivoting cube model, but several questions remain.

We mention the following:

- Is there a tighter lower bound on the number of modules to be added for any configuration to be reconfigurable into a line? In particular, can we establish an $O(1)$ lower bound.
- Is there a polynomial time algorithm to determine whether two configurations are equivalent under the equivalence relation described previously?
- Is there an extended moveset that would yield universal reconfiguration.

These are all potential future directions for our work, and the variety of open problems show the incredible richness of the field of modular robotics.

Bibliography

[1] Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D Demaine, Robin Flatland, Stefan Langerman, Joseph O'Rourke, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhrer. Linear reconfiguration of cube-style modular robots. *Computational geometry*, 42(6):652–663, 2009.

[2] Byoung Kwon An. Em-cube: cube-shaped, self-reconfigurable robots sliding on structure surfaces. In *Robotics and Automation, 2008. Proceedings. 2008 IEEE International Conference on*, pages 3149–3155, 2008.

[3] Nora Ayanian, Paul J White, Adám Hálász, Mark Yim, and Vijay Kumar. Stochastic control for self-assembly of XBots. In *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 1169–1176. American Society of Mechanical Engineers, 2008.

[4] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, 14(6):926–939, 1998.

[5] Nadia M Benbernou. *Geometric algorithms for reconfigurable structures*. PhD thesis, Massachusetts Institute of Technology, 2011.

[6] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Optimizing schedules for prioritized path planning of multi-robot systems. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 271–276. IEEE, 2001.

[7] Chih-Jung Chiang and Gregory S Chirikjian. Modular robot motion planning using similarity metrics. *Autonomous Robots*, 10(1):91–106, 2001.

[8] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1327–1332. IEEE, 2002.

[9] Jay Davey, Ngai Kwok, and Mark Yim. Emulating self-reconfigurable robots-design of the smores system. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4464–4469. IEEE, 2012.

[10] Zvi Drezner. Dynamic facility location: The progressive p-median problem. *Location Science*, 3(1):1–7, 1995.

[11] Robert Fitch, Zack Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2460–2467. IEEE, 2003.

[12] Toshio Fukuda and Yoshio Kawauchi. Cellular robotic system (cebot) as one of the realization of self-organizing intelligent universal manipulator. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 662–667. IEEE, 1990.

[13] Toshio Fukuda and Seiya Nakagawa. Dynamically reconfigurable robotic system. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 1581–1586. IEEE, 1988.

[14] Toshio Fukuda, Seiya Nakagawa, Yoshio Kawauchi, and Martin Buss. Structure decision method for self organising robots based on cell structures-cebot. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 695–700. IEEE, 1989.

[15] Kyle Gilpin, Ara Knaian, and Daniela Rus. Robot pebbles: One centimeter modules for programmable matter through self-disassembly. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2485–2492. IEEE, 2010.

[16] Kazuo Hosokawa, Takehito Tsujimori, Teruo Fujii, Hayato Kaetsu, Hajime Asama, Yoji Kuroda, and Isao Endo. Self-organizing collective robots with morphogenesis in a vertical plane. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 2858–2863. IEEE, 1998.

[17] Paul Levi, Eugen Meister, and Florian Schlachter. Reconfigurable swarm robots produce self-assembling and self-repairing organisms. *Robotics and Autonomous Systems*, 62(10):1371–1376, 2014.

[18] Yan Meng, Yuyang Zhang, Abhay Sampath, Yaochu Jin, and Bernhard Sendhoff. Cross-ball: a new morphogenetic self-reconfigurable modular robot. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 267–272. IEEE, 2011.

[19] An Nguyen, Leonidas J Guibas, and Mark Yim. Controlled module density helps reconfiguration planning. In *Proc. of 4th International Workshop on Algorithmic Foundations of Robotics*, pages 23–36, 2000.

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

[21] Lionel S Penrose. Self-reproducing machines. *Scientific American*, 200(6):105–114, 1959.

[22] LS Penrose and Roger Penrose. A self-reproducing analogue. *Nature*, 179:1183–1183, 1957.

[23] Daniel Pickem, Magnus Egerstedt, and Jeff S Shamma. Complete heterogeneous self-reconfiguration: deadlock avoidance using hole-free assemblies. In

Estimation and Control of Networked Systems, volume 4, pages 404–410. Citeseer, 2013.

[24] Luciano CA Pimenta, Vijay Kumar, Renato C Mesquita, and Guilherme AS Pereira. Sensing and coverage for a network of heterogeneous robots. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3947–3952. IEEE, 2008.

[25] John W Romanishin, Kyle Gilpin, Sebastian Claici, and Daniela 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*, pages 1925–1932. IEEE, 2015.

[26] John W Romanishin, Kyle Gilpin, and Daniela Rus. M-blocks: Momentum-driven, magnetic modular robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4288–4295. IEEE, 2013.

[27] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[28] Daniela Rus and Marsette Vona. Self-reconfiguration planning with compressible unit modules. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 4, pages 2513–2520. IEEE, 1999.

[29] Daniela Rus and Marsette Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, 2001.

[30] S Salapaka, A Khalak, and MA Dahleh. Constraints on locational optimization problems. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 2, pages 1741–1746. IEEE, 2003.

[31] Mac Schwager, James McLurkin, and Daniela Rus. Distributed coverage control with sensory feedback for networked robots. In *robotics: science and systems*, 2006.

[32] Mac Schwager, Daniela Rus, and Jean-Jacques Slotine. Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research*, 28(3):357–375, 2009.

[33] Mac Schwager, Jean-Jacques Slotine, and Daniela Rus. Consensus learning for distributed coverage control. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1042–1048. IEEE, 2008.

[34] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. prentice-Hall Englewood Cliffs, NJ, 1991.

[35] Elias M Stein and Rami Shakarchi. *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009.

[36] Cynthia Sung, James Bern, John Romanishin, and Daniela Rus. Reconfiguration planning for pivoting cube modular robots. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, Seattle, WA, May 2015. IEEE.

[37] Yosuke Suzuki, Norio Inou, Hitoshi Kimura, and Michihiko Koseki. Reconfigurable group robots adaptively transforming a mechanical structure-numerical expression of criteria for structural transformation and automatic motion planning method. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2361–2367. IEEE, 2007.

[38] Alfred Weber and Carl Joachim Friedrich. *Theory of the Location of Industries*. University of Chicago, 1929.

[39] George M Whitesides and Mila Boncheva. Beyond molecules: Self-assembly of mesoscopic and macroscopic components. *Proceedings of the National Academy of Sciences*, 99(8):4769–4774, 2002.

[40] George M Whitesides and Bartosz Grzybowski. Self-assembly at all scales. *Science*, 295(5564):2418–2421, 2002.

[41] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *Robotics & Automation Magazine, IEEE*, 14(1):43–52, 2007.

[42] Mark Yim, Babak Shirmohammadi, Jimmy Sastra, Michael Park, Michael Dugan, and Camillo J Taylor. Towards robotic self-reassembly after explosion. *Departmental Papers (MEAM)*, page 147, 2007.

[43] Mark Yim, Paul White, Michael Park, and Jimmy Sastra. Modular self-reconfigurable robots. *Encyclopedia of complexity and systems science*, pages 5618–5631, 2009.