SHAPING A SWARM USING A SHARED CONTROL INPUT

A Thesis

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

by
Shiva Shahrokhi
August 2018

SHAPING A SWARM USING A SHARED CONTROL INPUT

Shiva Shahrokhi

Approved:

Chair of the Committee
Aaron T. Becker, Assistant Professor
Department of Electrical and Computer Engineering

Committee Members:

David Mayerich, Assistant Professor
Department of Electrical and Computer Engineering

Rose Faghih, Assistant Professor
Department of Electrical and Computer Engineering

Suresh K. Khator, Associate Dean,
Cullen College of Engineering

Badrinath Roysam, Professor and Chair,
Electrical and Computer Engineering

To my wonderful parents, and my lovely husband.

SHAPING A SWARM USING A SHARED CONTROL INPUT

An Abstract

of a

Thesis

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

by
Shiva Shahrokhi
August 2018

Abstract

Micro-robots are small enough to move through the passageways of the body, therefore they are suited for targeted drug delivery and micro-scale manufacturing. Due to their small size, a single robot does not have enough force to deliver payloads, and it is prohibitively difficult to have onboard computation. Therefore, these robots are usually controlled by global inputs such as a uniform external magnetic field. This thesis presents controllers and algorithms for steering such an under-actuated swarm. This work first proves that the mean position of the swarm is controllable, and shows how an obstacle can make the variance controllable. Then it derives automatic controllers for these and a hysteresis-based switching control to regulate the first two moments of the swarm distribution. Finally, this work uses friction with boundary walls to break the symmetry caused by the global input and uses it to steer two particles to arbitrary positions.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Shaping a Swarm Using a Shared Control Input

Micro- and nanorobotics can be manufactured in large numbers. Our vision is for large swarms of robots remotely guided 1) through the human body, to cure disease, heal tissue, and prevent infection and 2) ex vivo to assemble structures in parallel. For each application, large numbers of micro robots are required to deliver sufficient payloads, but the small size of these robots makes it difficult to perform onboard computation. Instead, these robots are often controlled by a global, broadcast signal. The biggest barrier to this vision is a lack of control techniques that can reliably exploit large populations despite incredible under-actuation. Additionally, it is not always possible to gather pose information on each robot for feedback control. Robots might be difficult or impossible to sense individually due to their size and location. For example, micro-robots are smaller than the minimum resolution of a clinical MRI-scanner [1]. However, it is often possible to sense global properties of the group, such as mean position and variance. To make progress in automatic control with global inputs, we present swarm manipulation controllers requiring only mean and variance measurements of the robot's positions. These controllers are used as primitives to perform a block-pushing task illustrated in Fig. 1.1. A limitation was that variance control could only compress a swarm along the world x and y axes. This means the swarm could not navigate workspaces with narrow corridors with other orientations, such as those shown in Fig. 1.3. Challenges like these require a controller that regulates the swarm's position covariance, σ_{xy}. This thesis continues to prove that two orthogonal boundaries with high friction are sufficient to arbitrarily position two robots (Section 2.3), implements these position control algorithms in simulation

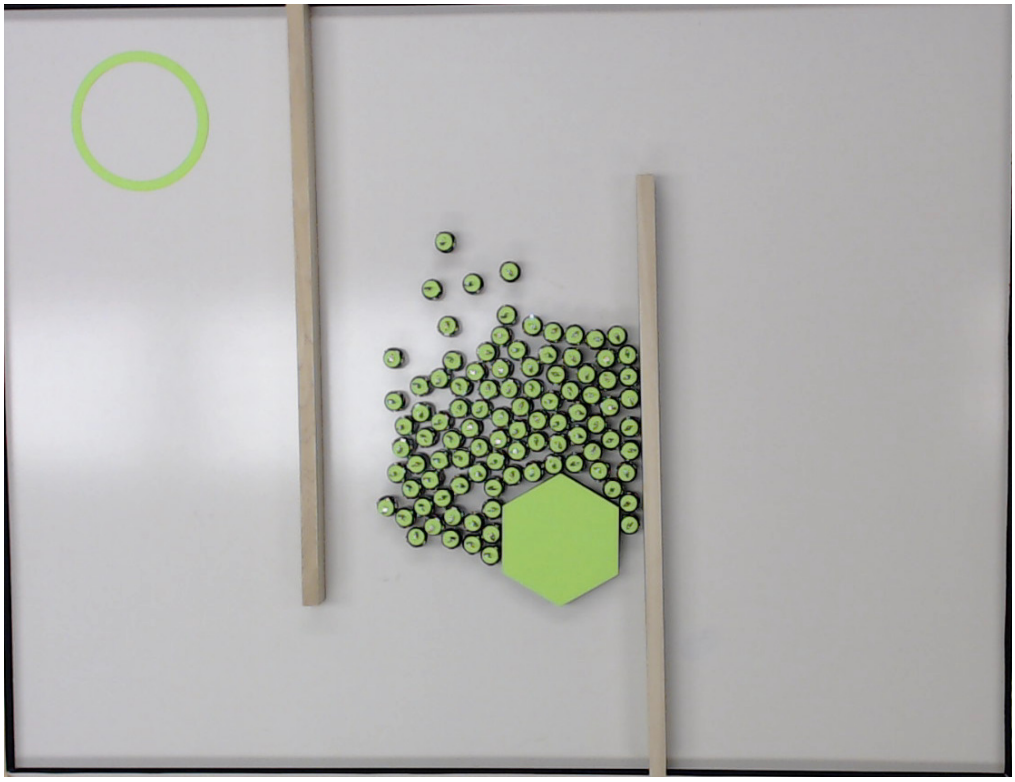

Figure 1.1: A swarm of robots, all controlled by a uniform force field, can be effectively controlled by a hybrid controller that knows only the first and second moments of the robot distribution. Here a mockup of a swarm of hardware robots(kilobots) that pushes a green block towards the goal is shown.

(Section 2.4) and on a hardware setup with up to 64 robots (Section 2.5), and ends with directions for future research.

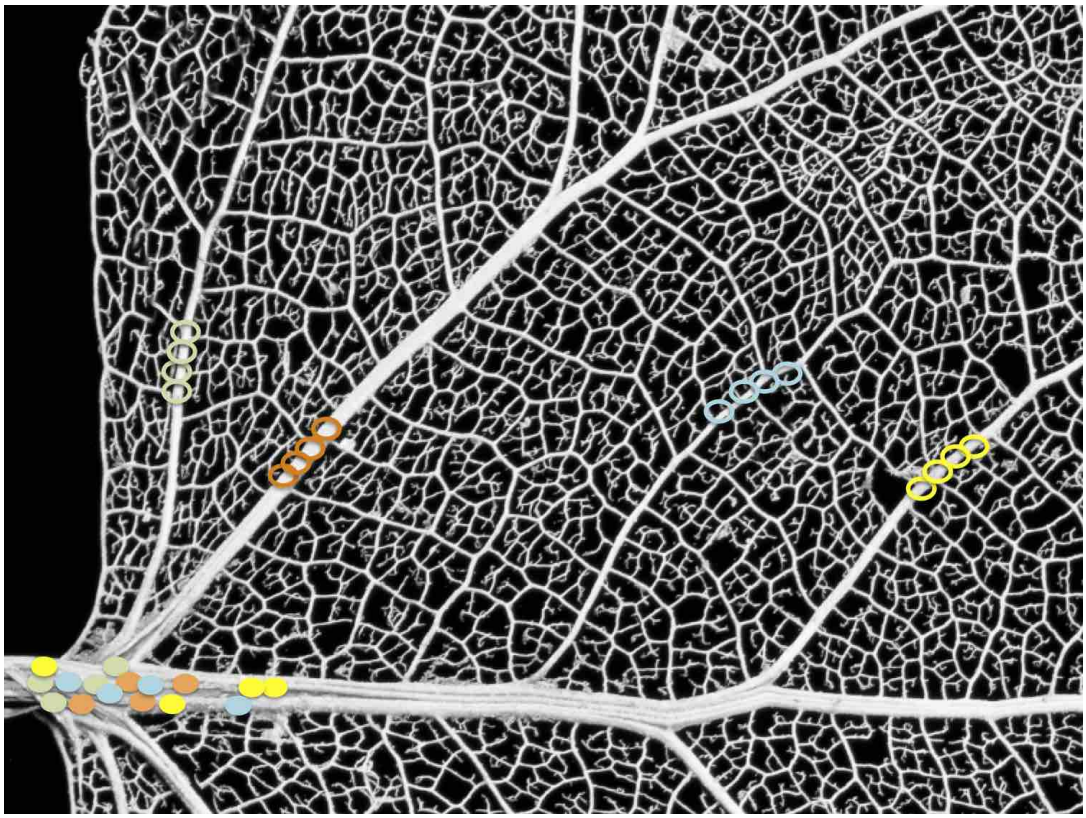

Figure 1.2: Navigating a swarm using global inputs, where each member receives the same control inputs, is challenging due to many obstacles. This thesis demonstrates how friction with walls can be used to change the shape of a swarm.

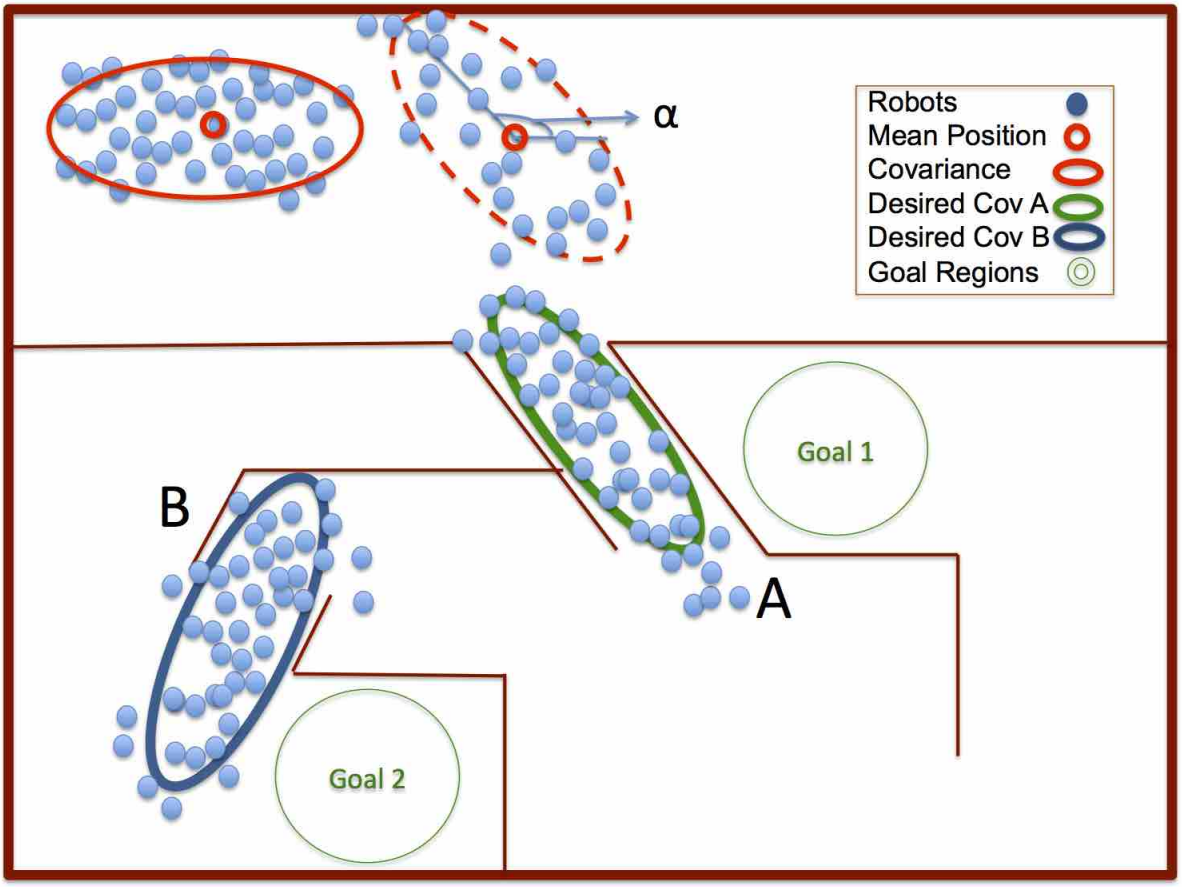

Figure 1.3: Maintaining group cohesion while steering a swarm through an arbitrary maze requires covariance control.

Chapter 2

Stochastic Swarm Control with Global Inputs

2.1 Related work

Controlling the *shape*, or relative positions, of a swarm of robots is a key ability for a myriad of applications. Correspondingly, it has been studied from a control-theoretic perspective in both centralized, e.g. virtual leaders in [2], and decentralized approaches, e.g. control-Lyapunov functions gradient based decentralized controllers in [3]. Most approaches assume a level of intelligence and autonomy in the individual robots that exceeds the capabilities of current micro- and nano-robots [4, 5].

Instead, this paper focuses on centralized techniques that apply the same control input to each member of the swarm, as in [6].

2.1.1 Global-control of micro- and nanorobots

We are particularly motivated by harsh constraints in micro- and nanorobotic systems. Small robots are often powered and steered by a global, broadcast control signal. Examples include *scratch-drive microrobots*, actuated and controlled by a DC voltage signal from a substrate [7, 8]; *light-driven nanocars*, synthetic molecules actuated by a specific wavelength of light [9], *MagMite* microrobots with different resonant frequencies controlled by a global magnetic field [10]; and magnetically controlled nanoscale helical screws [11, 12]. Large numbers of robots can be constructed, but the user interaction required to individually control each robot scales linearly with robot population. Instead, user interaction is often constrained to modifying a global input: while one robot is controlled, the rest are ignored. Making progress in targeted therapy and swarm

manipulation requires the coordinated control of large robot populations.

2.1.2 Human user studies with large swarms

There is currently no comprehensive understanding of user interfaces for controlling multi-robot systems with massive populations [13]. Our previous work with over a hundred hardware robots and thousands of simulated robots [6] demonstrated that direct human control of large swarms is possible. Unfortunately, the logistical challenges of repeated experiments with over one hundred robots prevented large-scale tests. To gather better data, we designed *SwarmControl.net*, a large-scale online game to test how humans interact with large swarms [14]. Our goal was to test several scenarios involving large-scale human-swarm interaction, and to do so with a statistically-significant sample size. These experiments showed that numerous simple robots responding to global control inputs are directly controllable by a human operator without special training, and that the visual feedback of the swarm state should be very simple in order to increase task performance. All code [15], and experimental results were posted online. The current paper presents motion primitives and an automatic controller that solves one of the games from SwarmControl.net.

2.1.3 Block-pushing and compliant manipulation

Unlike *caging* manipulation, where robots form a rigid arrangement around an object [16, 17], our swarm of robots is unable to grasp the blocks they push, and so our manipulation strategies are similar to *nonprehensile manipulation* techniques, e.g. [18], where forces must be applied along the center of mass of the moveable object. A key difference is that our robots are compliant and tend to flow around the object, making this similar to fluidic trapping [19, 20].

Our n-robot system with 2 control inputs and $4n$ states is inherently under-actuated,

and superficially bears resemblance to compliant, under-actuated manipulators [21, 22]. Like these manipulators, the swarm conforms to the object to be manipulated. However our swarm lacks the restoring force provided by flexures in [21] and the silicone in [22]. Our swarm tends to disperse itself, so we require artificial forces, such as the variance control primitives in Section 2.2.4, to regroup the swarm.

2.1.4 Using shear forces to shape a set of particles

Shear forces are unaligned forces that push one part of a body in one direction, and another part of the body in the opposite direction. These shear forces are common in fluid flow along boundaries, as described in introductory fluid dynamics textbooks [23]. Similarly, a swarm of robots under global control pushed along a boundary will experience shear forces. This is a position-dependent force, and so can be exploited to control the configuration or shape of the swarm. Physics-based swarm simulations have used these forces to disperse a swarm's spatial position for accomplishing coverage tasks [24].

More research has focused on generating artificial force-fields. Applications have included techniques to design shear forces to a single object for sensorless manipulation [25]. Vose et al. demonstrated a collection of 2D force fields generated by 6DOF vibration inputs to a rigid plate [26, 27]. This collection of force fields, including shear forces, could be used as a set of primitives for motion control for steering the formation of multiple objects.

2.2 Theory

2.2.1 Models

Consider holonomic robots that move in the 2D plane. We want to control position and velocity of the robots. First, assume a noiseless system containing one robot with mass m. Our inputs are global forces $[u_x, u_y]$. We define our state vector $\mathbf{x}(t)$ as the x

position, x velocity, y position and y velocity. The state-space representation in standard form is:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \tag{2.1}$$

$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

and our state space representation as:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix} [u_x, u_y]. \tag{2.2}$$

We want to find the number of states that we can control, which is given by the rank of the *controllability matrix*

$$\mathcal{C} = [B, AB, A^2B, ..., A^{n-1}B]. \tag{2.3}$$

$$\text{Here } \mathcal{C} = \left[\begin{array}{cc|cc|cc|cc} 0 & 0 & \frac{1}{m} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{m} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right], \tag{2.4}$$

and thus all four states are controllable.

2.2.2 Independent control with multiple robots is impossible

A single robot is fully controllable, but what happens with n robots? For holonomic robots, movement in the x and y coordinates are independent, so for notational convenience without loss of generality we will focus only on movement in the x axis. Given n robots to be controlled in the x axis, there are $2n$ states: n positions and n

velocities. Our state-space representation is:

$$\begin{bmatrix} \dot{x}_{1,1} \\ \dot{x}_{2.1} \\ \vdots \\ \dot{x}_{1,n} \\ \dot{x}_{2,n} \end{bmatrix} = \begin{bmatrix} 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{2,1} \\ \vdots \\ x_{1,n} \\ x_{2,n} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u_x. \tag{2.5}$$

However, just as with one robot, we can only control two states because $\mathcal{C}$ has rank two:

$$\mathcal{C} = \left[\begin{array}{c|c|c|c|c} 0 & 1 & 0 & 0 & \\ 1 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \dots \\ 0 & 1 & 0 & 0 & \\ 1 & 0 & 0 & 0 & \end{array}\right]. \tag{2.6}$$

2.2.3 Controlling mean position

This means *any* number of robots controlled by a global command have just two controllable states in each axis. We can not control the position of all the robots, but what states are controllable? To answer this question we create the following reduced order system that represents the average x position and x velocity of the n robots:

$$\begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 0 & 1 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{2,1} \\ \vdots \\ x_{1,n} \\ x_{2,n} \end{bmatrix}$$

$$+ \frac{1}{n} \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u_x. \tag{2.7}$$

Thus we have

$$\begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_x. \tag{2.8}$$

We again analyze $\mathcal{C}$:

$$\mathcal{C} = \left[\begin{array}{c|c} 0 & 1 \\ 1 & 0 \end{array}\right]. \tag{2.9}$$

This matrix has rank two, and thus the average position and average velocity are controllable.

Due to symmetry, only the mean position and mean velocity are controllable. However, there are several techniques for breaking symmetry, for example by allowing independent noise sources [28], or by using obstacles [6].

2.2.4 Controlling the variance of many robots

The variance, σ_x^2, σ_y^2, of the swarm's position is computed:

$$\begin{aligned} \overline{x}(\mathbf{x}) &= \frac{1}{n}\sum_{i=1}^{n} x_{1,i}, \qquad \sigma_x^2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_{1,i} - \overline{x})^2, \\ \overline{y}(\mathbf{x}) &= \frac{1}{n}\sum_{i=1}^{n} x_{3,i}, \qquad \sigma_y^2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_{3,i} - \overline{y})^2. \end{aligned} \tag{2.10}$$

Controlling the variance requires being able to increase and decrease the variance. We will list a sufficient condition for each. Both conditions are readily found at the micro and nanoscale. Real systems, especially at the micro scale, are affected by unmodelled dynamics. These unmodelled dynamics are dominated by Brownian noise. To model this (2.1) must be modified as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) + W\boldsymbol{\varepsilon}(t) \\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t), \end{aligned} \tag{2.11}$$

where $W\boldsymbol{\varepsilon}(t)$ is a random perturbation produced by Brownian noise. Given a large free workspace, a *Brownian noise* process increases the variance linearly with time:

$$\dot{\sigma_x^2}(\mathbf{x}(t), \mathbf{u}(t) = 0) = W\boldsymbol{\varepsilon}. \tag{2.12}$$

If robots with radius r are in a bounded environment with sides of length $[\ell_x, \ell_y]$, the unforced variance asymptotically grows to the variance of a uniform distribution,

$$[\sigma_x^2, \sigma_y^2] = \frac{1}{12}[(\ell_x - 2r)^2, (\ell_y - 2r)^2]. \tag{2.13}$$

A flat obstacle can be used to decrease variance. Pushing a group of dispersed robots against a flat obstacle will decrease their variance until the minimum-variance (maximum density) packing is reached. For large n, Graham and Sloan showed that the minimum-variance packing $\sigma^2_{optimal}(n, r)$ for n circles with radius r is $\approx \frac{\sqrt{3}}{\pi}(nr)^2 \approx 0.55(nr)^2$ [29].

We will prove the origin is globally asymptotically stabilizable by using a control-Lyapunov function [30]. A suitable Lyapunov function is squared variance error:

$$V(t, \mathbf{x}) = \frac{1}{2}(\sigma^2(\mathbf{x}) - \sigma^2_{goal})^2$$

$$\dot{V}(t, \mathbf{x}) = (\sigma^2(\mathbf{x}) - \sigma^2_{goal})\dot{\sigma^2}(\mathbf{x}). \tag{2.14}$$

We note here that $V(t, \mathbf{x})$ is positive definite and radially unbounded, and $V(t, \mathbf{x}) \equiv 0$ only at $\sigma^2(\mathbf{x}) = \sigma^2_{goal}$. To make $\dot{V}(t, \mathbf{x})$ negative semi-definite, we choose

$$u(t) = \begin{cases} \text{move to wall} & \text{if } \sigma^2(\mathbf{x}) > \sigma^2_{goal} \\ \text{move from wall} & \text{if } \sigma^2(\mathbf{x}) \leq \sigma^2_{goal}. \end{cases} \tag{2.15}$$

For such a $u(t)$,

$$\dot{\sigma^2}(\mathbf{x}) = \begin{cases} \text{negative} & \text{if } \sigma^2(\mathbf{x}) > \max(\sigma^2_{goal}, \sigma^2_{optimal}(n, r)) \\ W\epsilon & \text{if } \sigma^2(\mathbf{x}) \leq \sigma^2_{goal}, \end{cases} \tag{2.16}$$

and thus $\dot{V}(t, \mathbf{x})$ is negative definite and the variance is globally asymptotically stabilizable.

2.2.5 Controlling both mean and variance of many robots

The mean and variance of the swarm cannot be controlled simultaneously, however if the dispersion due to Brownian motion is much less than the maximum controlled speed,

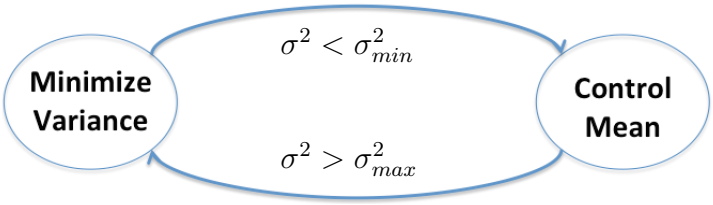

Figure 2.1: Two states for controlling the mean and variance of a robot swarm.

we can adopt a hybrid, hysteresis-based controller to regulate the mean and variance shown in Alg. 1. Such a controller normally controls the mean position according to (2.19), but switches to minimizing variance if the variance exceeds some σ^2_{max}. The variance is lowered to less than σ^2_{min}, and the system returns to controlling the mean position. This is a standard technique for dealing with control objectives that evolve at different rates [31, 32], and the hysteresis avoids rapid switching between control modes. The process is illustrated in Fig. 2.1.

Algorithm 1 Hybrid mean and variance control

Require: Knowledge of swarm mean $[\bar{x}, \bar{y}]$, variance $[\sigma^2_x, \sigma^2_y]$, the locations of the rectangular boundary $\{x_{min}, x_{max}, y_{min}, y_{max}\}$, and the target mean position $[x_{target}, y_{target}]$.

$flag_x \leftarrow$ **false**, $flag_y \leftarrow$ **false**
$x_{goal} \leftarrow x_{target}$, $y_{goal} \leftarrow y_{target}$
loop
  if $\sigma^2_x > \sigma^2_{max}$ **then**
    $x_{goal} \leftarrow x_{min}$
    $flag_x \leftarrow$ **true**
  else if $flag_x$ **and** $\sigma^2_x < \sigma^2_{min}$
    $x_{goal} \leftarrow x_{target}$
    $flag_x \leftarrow$ **false**
  end if
  if $\sigma^2_y > \sigma^2_{max}$ **then**
    $y_{goal} \leftarrow y_{min}$
    $flag_y \leftarrow$ **true**
  else if $flag_y$ **and** $\sigma^2_y < \sigma^2_{min}$
    $y_{goal} \leftarrow y_{target}$
    $flag_y \leftarrow$ **false**
  end if
  Apply (2.19) to move toward $[x_{goal}, y_{goal}]$
end loop

A key challenge is to select proper values for σ^2_{min} and σ^2_{max}. The optimal packing

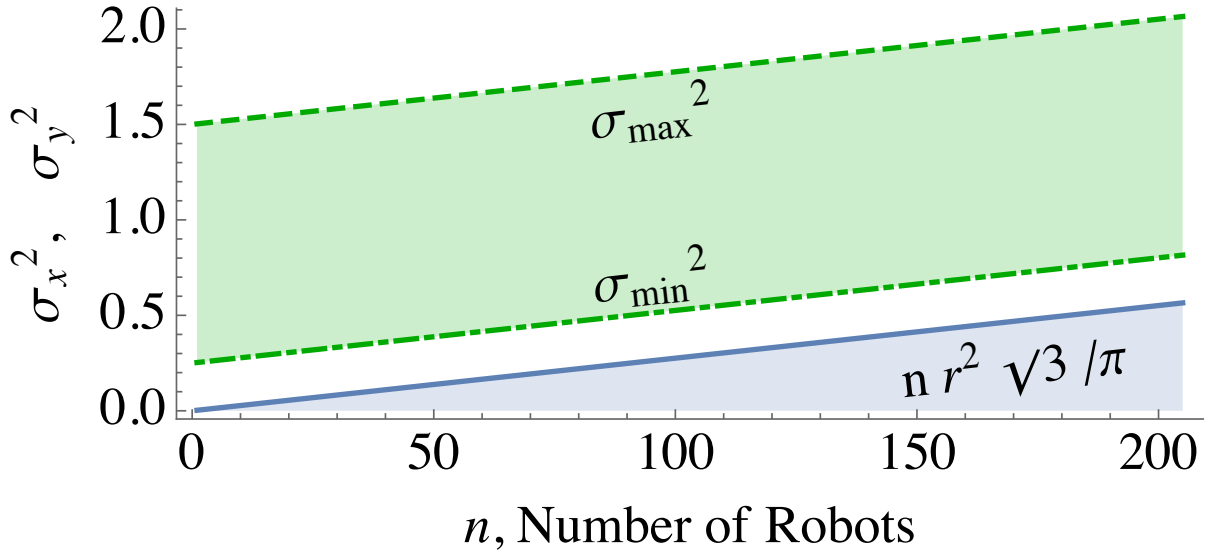

Figure 2.2: The switching conditions for variance control are set as a function of n, and designed to be larger than the optimal packing density. The above plot uses robot radius $r = 1/10$.

variance is $\sigma^2_{optimal}(n, r) = \frac{\sqrt{3}}{\pi} nr^2$. The random packings generated by pushing our robots into corners are suboptimal, so we choose the conservative values shown in Fig. 2.2:

$$\sigma^2_{min} = 2.5r + \sigma^2_{optimal}(n, r)$$

$$\sigma^2_{max} = 15r + \sigma^2_{optimal}(n, r). \tag{2.17}$$

2.2.6 Controlling covariance using friction

Global inputs move a swarm uniformly. Controlling covariance requires breaking this uniform symmetry. A swarm inside an axis-aligned rectangular workspace can reduce variance normal to a wall by simply pushing the swarm into the boundary. Directly controlling covariance by pushing the swarm into a boundary requires changing the boundary. An obstacle in the lower-right corner is enough to generate positive covariance. Generating both positive and negative covariance requires additional obstacles. Requiring special obstacle configuration also makes covariance control dependent on the local environment. Instead of pushing our robots directly into a wall, this paper examines an oblique approach, by using boundaries that generate friction with the robots.

These frictional forces are sufficient to break the symmetry caused by uniform inputs. Robots touching a wall have a negative friction force that opposes movement along the boundary, as shown in Eq. (2.18). This causes robots along the boundary to slow down compared to robots in free-space. This enables covariance control using boundaries with arbitrary orientations.

Let the control input be a vector force $\vec{F}$ with magnitude F and orientation θ. The force of friction F_f is

$$\begin{aligned} N &= F\cos(\theta) \\ F_f &= \begin{cases} \mu_f N, & \mu_f N < F\sin(\theta) \\ F\sin(\theta), & \text{else} \end{cases} \\ F_{forward} &= F\sin(\theta) - F_f. \end{aligned} \tag{2.18}$$

Fig. 2.3 shows the resultant forces on two robots when one is touching a wall. As illustrated, bot experiences different net forces although each receive the same inputs. For ease of analysis, the following algorithms assume μ_f is infinite and robots touching the wall are prevented from sliding along the wall. This means that if one robot is touching the wall and another robot is free, if the control input is parallel or into the wall, the touching robot will not move. The next section shows how a system with friction model (2.18) and two walls are sufficient to arbitrarily position two robots.

2.3 Position control of 2 robots using wall friction

This section describes an algorithm for positioning two robots and introduces concepts that will be used for multi-robot positioning. As we can see in Alg. 2, assume two robots are initialized at s_1 and s_2 with corresponding goal destinations e_1 and e_2. Denote the current positions of the robots r_1 and r_2. Let the subscripts x and y denote the x and y coordinates, i.e., s_{1x} and s_{1y} denote the x and y locations of s_1. The algorithm assigns a global control input at every instance. As a result, our goal is to adjust $\Delta r_x = r_{2x} - r_{1x}$

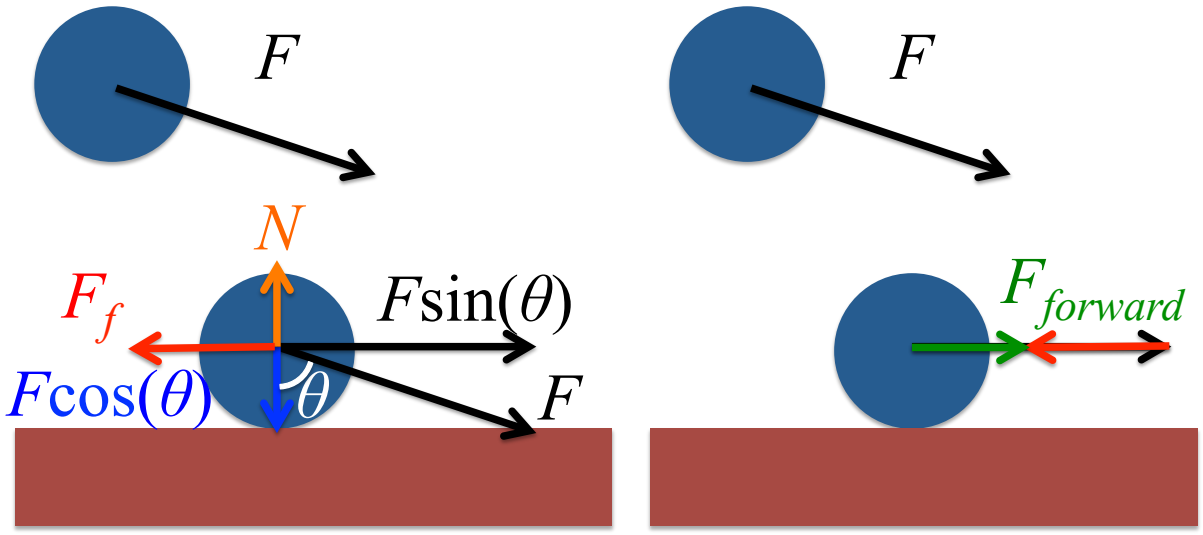

Figure 2.3: Wall friction reduces the force for going forward $F_{forward}$ on a robot near a wall, but not for a free robot.

from $\Delta s_x = s_{2x} - s_{1x}$ to $\Delta e_x = e_{2x} - e_{1x}$ and similarly adjust $\Delta r_y = r_{2y} - r_{1y}$ from $\Delta s_y = s_{2y} - s_{1y}$ to $\Delta e_y = e_{2y} - e_{1y}$ with one global input at every instance. The key to the algorithm is the position-dependent friction model (2.18).

Our algorithm uses a divide and conquer method to solve the positioning problem. It finds the final position of the robots in two steps: (i) First, $|\Delta r_x - \Delta e_x|$ is reduced to zero while Δr_y is kept constant in Alg. 3. (ii) Having fixed Δr_x to Δe_x as desired, the algorithm next keeps Δr_x constant and adjusts Δr_y to Δe_y, as desired in Alg.. 4. Though steps (i) and (ii) are similar from an algorithmic point of view, the following subsections describe the process in detail.

2.3.1 Step (i): Fixing Δr_x

- Define $e'_1 = (e_{1x}, s_{1y})$ and $e'_2 = (e_{2x}, s_{2y})$. Our goal for defining e'_1 and e'_2 is to understand the direction to which robots should move in order to adjust Δr_x. Let $e'_{\text{top}} = \arg\max_i e'_{iy}$ and $e'_{\text{bottom}} = \arg\min_i e'_{iy}$. Now if $e'_{\text{top,x}} - e'_{\text{bottom,x}} > 0$, then the

global input to both robots would be toward left direction and if $e'_{\text{top,x}} - e'_{\text{bottom,x}} < 0$, then the global input to both robots would be toward right direction. The two robots continue their horizontal path until one of them reaches the ϵ-neighborhood of one of the left or right walls.

- At this step, let $y_{\min} = \min_i r_{iy}$, i.e., $y_{\min}$ is the minimum height of the two robots. We move both robots downward by the amount of $y_{\min}$ such that one of the robots would touch the bottom wall and hence friction force will not let that robot to move left or right.

- The fact that the friction force of the bottom wall would not let the lower robot to move right or left will let the other robot to move to right and left freely to adjust Δr_x according to Δe_x.

- Finally, even if with the free move of the upper robot Δr_x is not set to the Δe_x, we can run the Step (i) (as described in the previous paragraphs) again to adjust the Δr_x. It is easy to show that it is guaranteed that we can adjust Δr_x to Δe_x in only two iterations.

2.3.2 Step (ii): Fixing Δr_y

Now that we have adjusted the difference in robots' positions along one axis, we focus to do the same on the other axis as well. Therefore, similar to Section 2.3.1, we employ the following steps:

- Let s'_1 and s'_2 be the points we derived at the end of the steps in Section 2.3.1.

- Define $e''_1 = (s'_{1x}, e_{1y})$ and $e''_2 = (s'_{2x}, e_{2y})$. We define e''_1 and e''_2 to understand the direction to which robots should move in order to adjust Δr_y. Let $e''_{\text{right}} = \arg\max_i e''_{ix}$ and $e''_{\text{left}} = \arg\min_i e'_{ix}$. Now if $e''_{\text{right,y}} - e''_{\text{left,y}} > 0$, then the global input to both robots would be toward down direction and if $e''_{\text{right,y}} - e''_{\text{left,y}} < 0$,

then the global input to both robots would be toward up direction. The two robots continue their vertical path until one of them reaches the ϵ-neighborhood of one of the top or bottom walls.

- At this step, let $x_{\min} = \min_i r_{ix}$, i.e., $x_{\min}$ is the minimum distance of the two robots from the origin along the x-axis. We move both robots to the left by the amount of $x_{\min}$ such that one of the robots would touch the left wall and hence friction force will not let that robot to move up or down.

- The fact that the friction force of the left wall would not let one of the robots to move up or down will let the other robot to move to up or down freely to adjust Δr_y according to Δe_y.

- Finally, even if with the free move of the robot which is not touching the wall Δr_y is not set to the Δe_y, we can run the Step (i) (as described in the previous paragraphs) again to adjust the Δr_y. It is easy to show that it is guaranteed that we can adjust Δr_y to Δe_y in only two iterations.

Once Δr_x and Δr_y are set to Δe_x and Δe_y, we can use global input to easily move both robots from r_1 and r_2 toward e_1 and e_2.

Algorithm 2 WallFrictionArrange2Robots(s_1, s_2, e_1, e_2, L)

Require: Knowledge of starting (s_1, s_2) and ending (e_1, e_2) positions of two robots. $(0, 0)$ is bottom corner, s_1 is rightmost robot, L is length of the walls. Current position of the robots are (r_1, r_2).

1: (r_1, r_2) = GenerateDesired$x$-spacing(s_1, s_2, e_1, e_2, L)
2: GenerateDesired$y$-spacing(r_1, r_2, e_1, e_2, L)

2.4 Simulation

Our simulations use a Javascript port of Box2D, a popular 2D physics engine with support for rigid-body dynamics and fixed-time step simulation [33]. All experiments ran on a Chrome web browser on a 2.6 GHz Macbook. All code is available at [34].

Algorithm 3 GenerateDesired$x$-spacing(s_1, s_2, e_1, e_2, L)

Require: Knowledge of starting (s_1, s_2) and ending (e_1, e_2) positions of two robots. $(0, 0)$ is bottom corner, s_1 is topmost robot, L is length of the walls. Current robot positions are (r_1, r_2).

Ensure: $r_{1y} - r_{2y} \equiv s_{1y} - s_{2y}$

$\epsilon \leftarrow$ small number
$\Delta s_x \leftarrow s_{1x} - s_{2x}$
$\Delta e_x \leftarrow e_{1x} - e_{2x}$
$r_1 \leftarrow s_1$, $r_2 \leftarrow s_2$
if $\Delta e_x < 0$ **then**
    $m \leftarrow (L - \epsilon - \max(r_{1x}, r_{2x}), 0)$ ▷ Move to right wall
else
    $m \leftarrow (\epsilon - \min(r_{1x}, r_{2x}), 0)$ ▷ Move to left wall
end if
$m \leftarrow m + (0, -\min(r_{1y}, r_{2y}))$ ▷ Move to bottom
$r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
if $\Delta e_x - (r_{1x} - r_{2x}) > 0$ **then**
    $m \leftarrow (\min(|\Delta e_x - \Delta s_x|, L - r_{1x}), 0)$ ▷ Move right
else
    $m \leftarrow (-\min(|\Delta e_x - \Delta s_x|, r_{1x}), 0)$ ▷ Move left
end if
$m \leftarrow m + (0, \epsilon)$ ▷ Move up
$r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
$\Delta r_x = r_{1x} - r_{2x}$
if $\Delta r_x \equiv \Delta e_x$ **then**
    return (r_1, r_2)
else
    return GenerateDesired$x$-spacing(r_1, r_2, e_1, e_2, L)
end if

2.4.1 Controlling the mean position

We control mean position with a PD controller that uses the mean position and mean velocity. Our control input is the global force applied to each robot:

$$u_x = K_p(x_{goal} - \bar{x}) + K_d(0 - \bar{v}_x)$$

$$u_y = K_p(y_{goal} - \bar{y}) + K_d(0 - \bar{v}_y) \tag{2.19}$$

here K_p is the proportional gain, and K_d is the derivative gain. We performed a parameter sweep to identify the best values. Representative experiments are shown in Fig. 2.4. 100 robots were used and the maximum speed was 3 meters per second. As shown in Fig. 2.4,

Algorithm 4 GenerateDesired$y$-spacing(s_1, s_2, e_1, e_2, L)

Require: Knowledge of starting (s_1, s_2) and ending (e_1, e_2) positions of two robots. $(0, 0)$ is bottom corner, s_1 is rightmost robot, L is length of the walls. Current position of the robots are (r_1, r_2).

Ensure: $r_{1x} - r_{2x} \equiv s_{1x} - s_{2x}$

$\Delta s_y \leftarrow s_{1y} - s_{2y}$
$\Delta e_y \leftarrow e_{1y} - e_{2y}$
$r_1 \leftarrow s_1$, $r_2 \leftarrow s_2$
if $\Delta e_y < 0$ **then**
    $m \leftarrow (L - \max(r_{1y}, r_{2y}), 0)$ ▷ Move to top wall
else
    $m \leftarrow (-\min(r_{1y}, r_{2y}), 0)$ ▷ Move to bottom wall
end if
$m \leftarrow m + (0, -\min(r_{1x}, r_{2x}))$ ▷ Move to left
$r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
if $\Delta e_y - (r_{1y} - r_{2y}) > 0$ **then**
    $m \leftarrow (\min(|\Delta e_y - \Delta s_y|, L - r_{1y}), 0)$ ▷ Move top
else
    $m \leftarrow (-\min(|\Delta e_y - \Delta s_y|, r_{1y}), 0)$ ▷ Move bottom
end if
$m \leftarrow m + (0, \epsilon)$ ▷ Move right
$r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
$\Delta r_y = r_{1y} - r_{2y}$
if $\Delta r_y \equiv \Delta e_y$ **then**
    $m \leftarrow (e_{1x} - r_{1x}, e_{1y} - r_{1y})$
    $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
    return (r_1, r_2)
else
    return GenerateDesired$y$-spacing(r_1, r_2, e_1, e_2, L)
end if

we can achieve an overshoot of 1% and a rise time of 1.52 s with $K_p = 4$, and $K_d = 1$. Fig. 2.5 shows an example of controlling mean position by making it trace the word "SWARM".

2.4.2 Controlling the variance

For variance control we use the control law discussed in Section 2.2.4. Moving away from the wall and waiting is sufficient to increase variance because Brownian noise naturally disperses the swarm in such a way that the variance increases linearly [35].

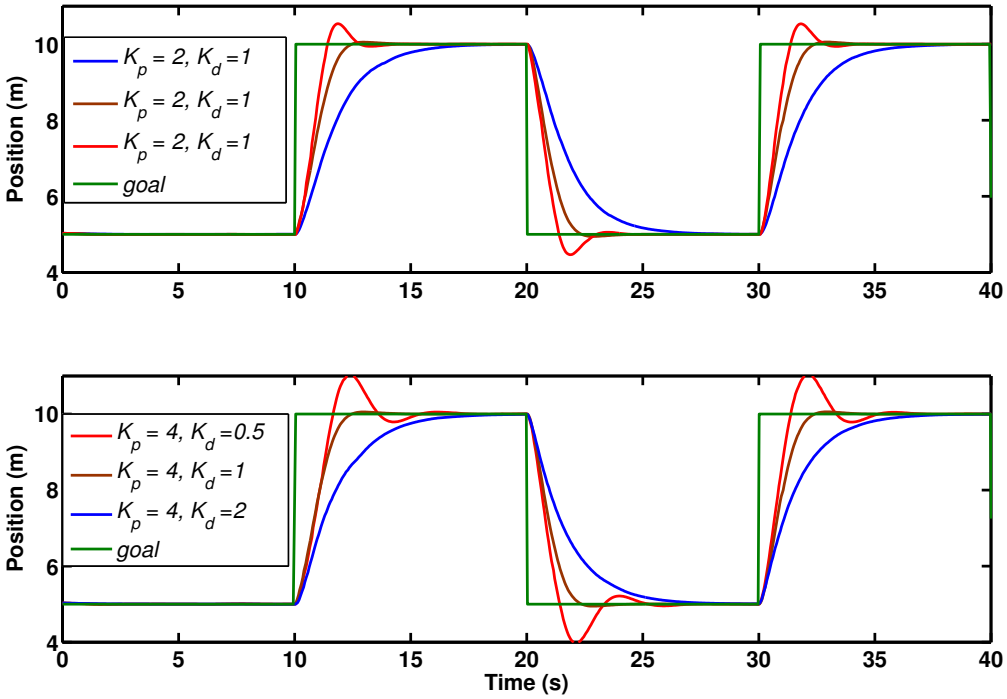

Figure 2.4: Tuning proportional (K_p, top) and derivative (K_d, bottom) gain values in (2.19) improves performance with $n = 100$ robots.

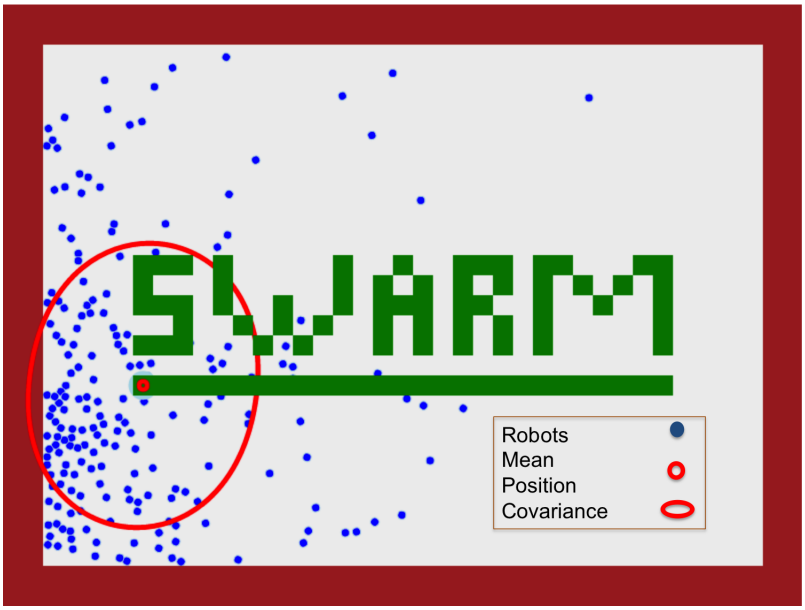

Figure 2.5: A frame from video attachment showing mean position control on a swarm of 200 robots. The mean position of the swarm traces "SWARM".

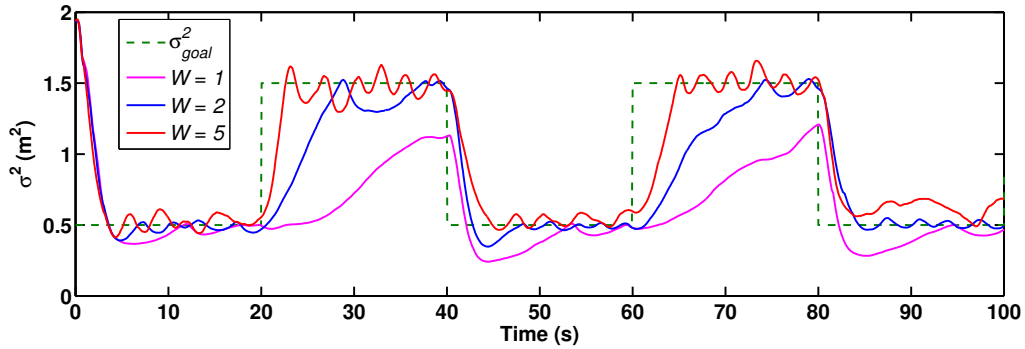

Figure 2.6: Increased noise results in more responsive variance control because stronger Brownian noise causes a faster increase of variance.

If faster dispersion is needed, the swarm can be pushed through obstacles such as a diffraction grating or Pachinko board [6].

The variance control law to regulate the variance to σ^2_{ref} has three gains:

$$u_x = K_p(x_{goal}(\sigma^2_{ref}) - \bar{x}) - K_d\bar{v}_x + K_i(\sigma^2_{ref} - \sigma^2_x)$$
$$u_y = K_p(y_{goal}(\sigma^2_{ref}) - \bar{y}) - K_d\bar{v}_y + K_i(\sigma^2_{ref} - \sigma^2_y). \tag{2.20}$$

In a slight abuse of notation we call the gain scaling the variance error K_i because the variance, if unregulated, integrates over time. Eq. (2.20) assumes the nearest wall is to the left of the robot at $x = 0$, and chooses a reference goal position that in steady-state would have the correct variance according to (2.13):

$$x_{goal}(\sigma^2_{ref}) = r + \sqrt{3\sigma^2_{ref}}. \tag{2.21}$$

If another wall is closer, the signs of $[K_p, K_i]$ are inverted, and the location x_{goal} is translated. Results are shown in Fig. 2.6, with $K_{p,i,d} = [4, 1, 1]$.

2.4.3 Hybrid control of mean and variance

Fig. 2.7 shows a simulation run of the hybrid controller in Alg. 1 with 100 robots in a square workspace containing no internal obstacles. Fig. 2.8 shows the experiment of running the hybrid controller.

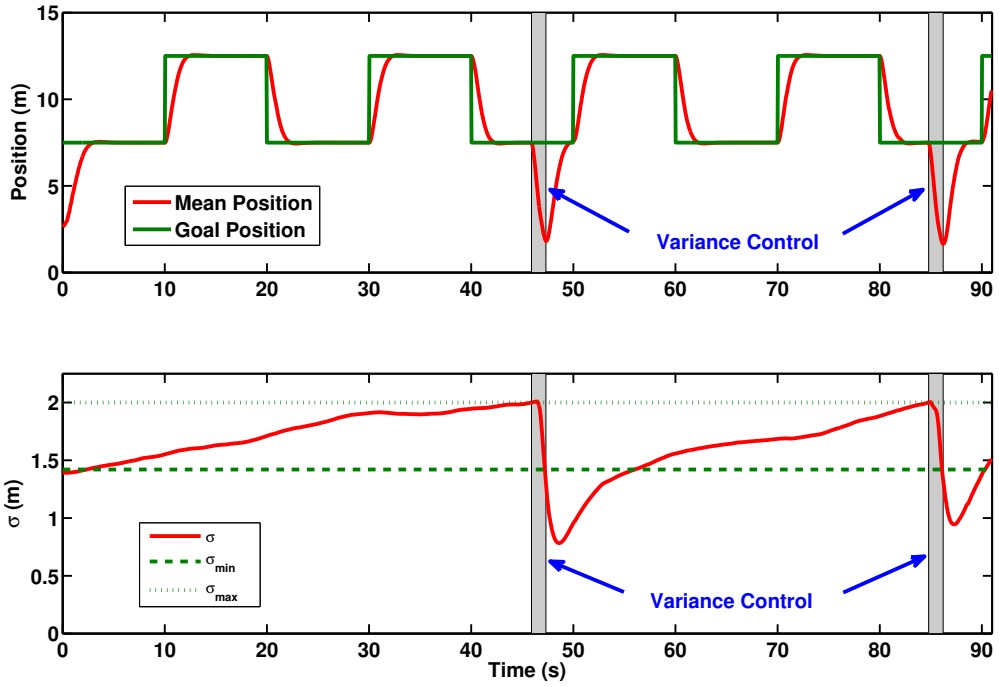

Figure 2.7: Simulation result with 100 robots under hybrid control Alg. 1, which controls both the mean position (top) and variance (bottom). For ease of analysis, only x position and variance are shown.

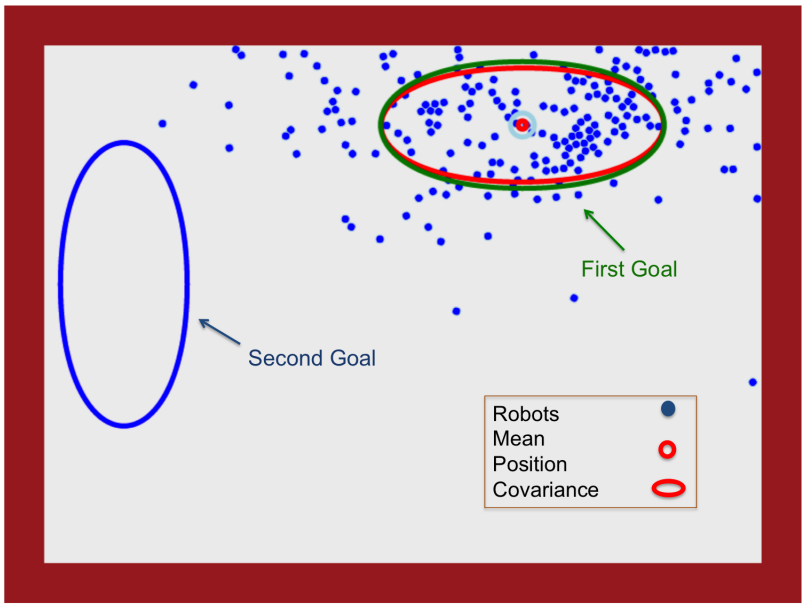

Figure 2.8: A frame from video, using Alg. 1 to control variance and mean position of a swarm of 200 robots.

Algorithms 2, 3, 4, were implemented in Mathematica using point robots (radius = 0). Figs 2.9 and 2.10 show the examples of the implementation of our algorithm. In both of these figures we have denoted the starting points and the destinations by small circles. However, destination points are surrounded by larger circles so as to be distinct from starting points.

In each of these figures we have five snapshots of the running of our algorithm taken every quarter second. For the sake of brevity we have replaced straight moves (e.g. upward, downward, etc) with oblique moves that shows a combination of two moves simultaneously (e.g., left and down together).

As we can see, in Fig. 2.9 Δr_x is adjusted to Δe_x in the second snapshot, i.e., at $t = t_1$ where $t_1 < 0.25$. The rest of the steps in this figure is dedicated to adjusting the Δr_y to Δe_y. As it is clear from Fig. 2.9, Δr_y is also adjusted at $t = t_2$ where $0.75 < t_2 < 1$. Finally, once Δr_x and Δr_y are adjusted, the algorithm gives a global input both of the robots so as to move them toward their corresponding destinations. This is happening in the time interval of $(t_2, 1]$.

Similarly, in Fig. 2.10 we can see that the Δr_x is adjusted in the third snapshot, i.e., at $t = t_3$ where $0.25 < t_3 < 0.5$ and Δr_y is adjusted in the last snapshot at $t = t_4$ where $0.75 < t_4 < 1$. The final positioning steps are happening in the time interval of $(t_4, 1]$.

As we pointed out earlier, adjusting each of Δr_x and Δr_y needs two iterations in the worst case. In other words, both of the Alg. 3 and Alg. 4 are executed two times in the worst case in positioning process of the robots. It is easy to see that we need two iterations of Alg. 3 only if $|\Delta e_x - \Delta s_x| > L$. Similarly we need two iterations of Alg. 4 only if $|\Delta e_y - \Delta s_y| > L$.

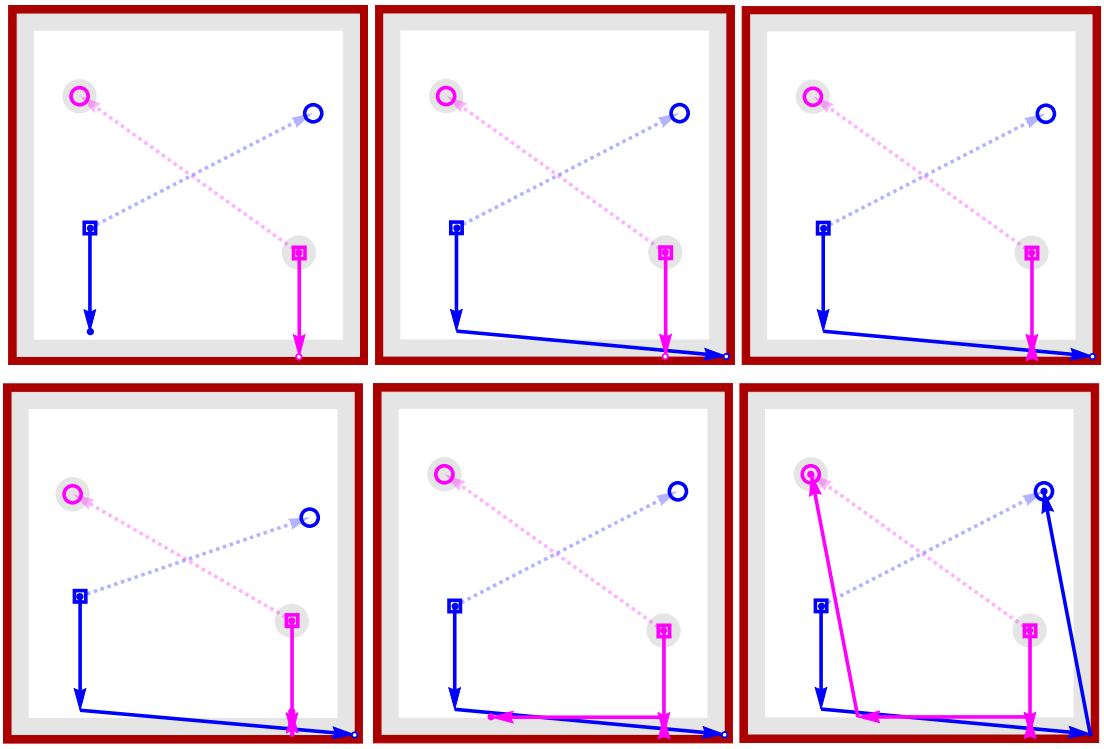

Figure 2.9: Frames from an implementation of Alg. 2: two robot positioning using walls with infinite friction. Robot initial positions are shown by a crosshair, and final positions by a circled square. Dashed lines show the shortest route if robots could be controlled independently. The path given by Alg. 2 is shown with solid arrows.

2.5 Block-pushing results

This section analyzes a *block-pushing* task attempted by both our hybrid, hysteresis-based controller and by human users.

2.5.1 Human-controlled block-pushing

In previous work over 1000 human users completed an online version of this task using varying levels of feedback. The original experiment explored manipulation with varying amounts of sensing information: **full-state** sensing showed the position of all robots; **convex-hull** drew a convex hull around the outermost robots; **mean** displayed the average position of the population; and **mean + variance** added a confidence ellipse.

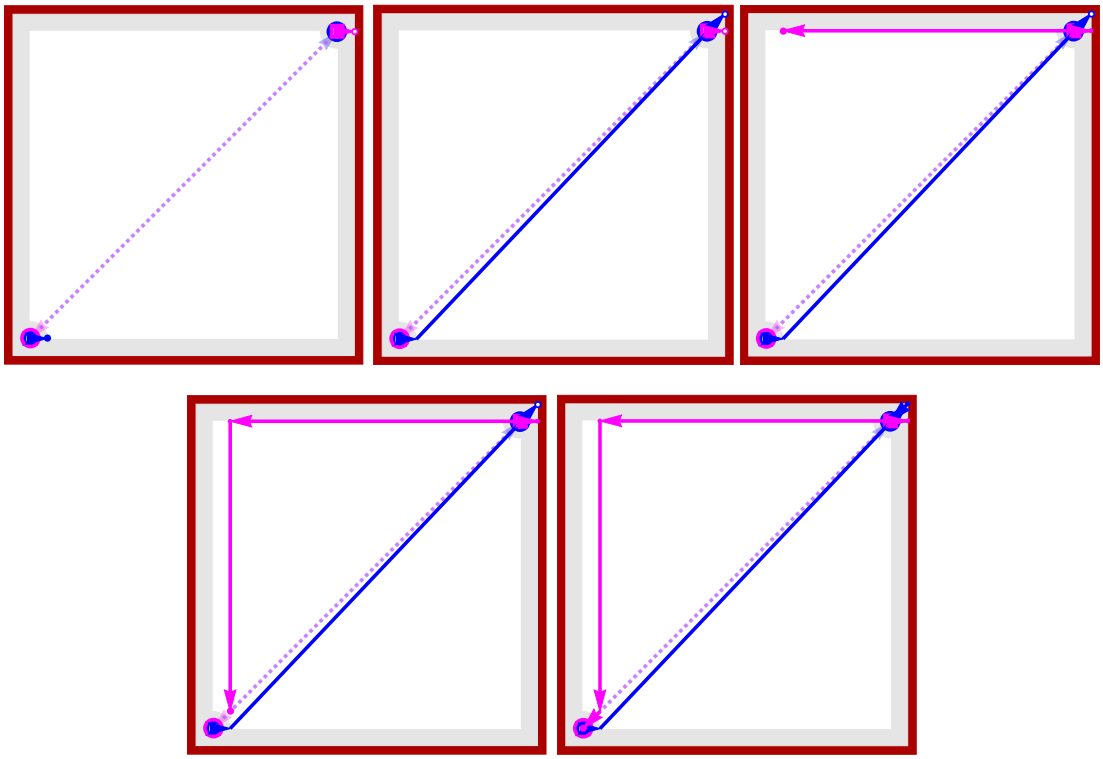

Figure 2.10: Two robot positioning: switching positions using walls with infinite friction.

Fig. 2.11 shows screenshots of the same robot swarm with each type of visual feedback. Full-state requires $2n$ data points for n robots. Convex-hull requires at worst $2n$, but usually a smaller number. Mean requires two, and variance three, data points. Mean and mean + variance are convenient even with millions of robots. We hypothesized a steady decay in performance as the amount of visual feedback decreased.

To our surprise, the results indicated the opposite: players with just the mean

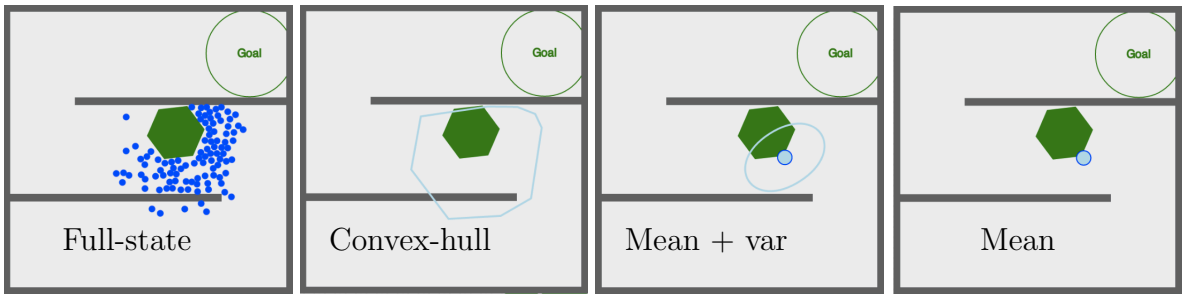

Figure 2.11: Screenshots from a block-pushing task with human users. This experiment challenged players to quickly steer 100 robots (blue discs) to push an object (green hexagon) into a goal region.

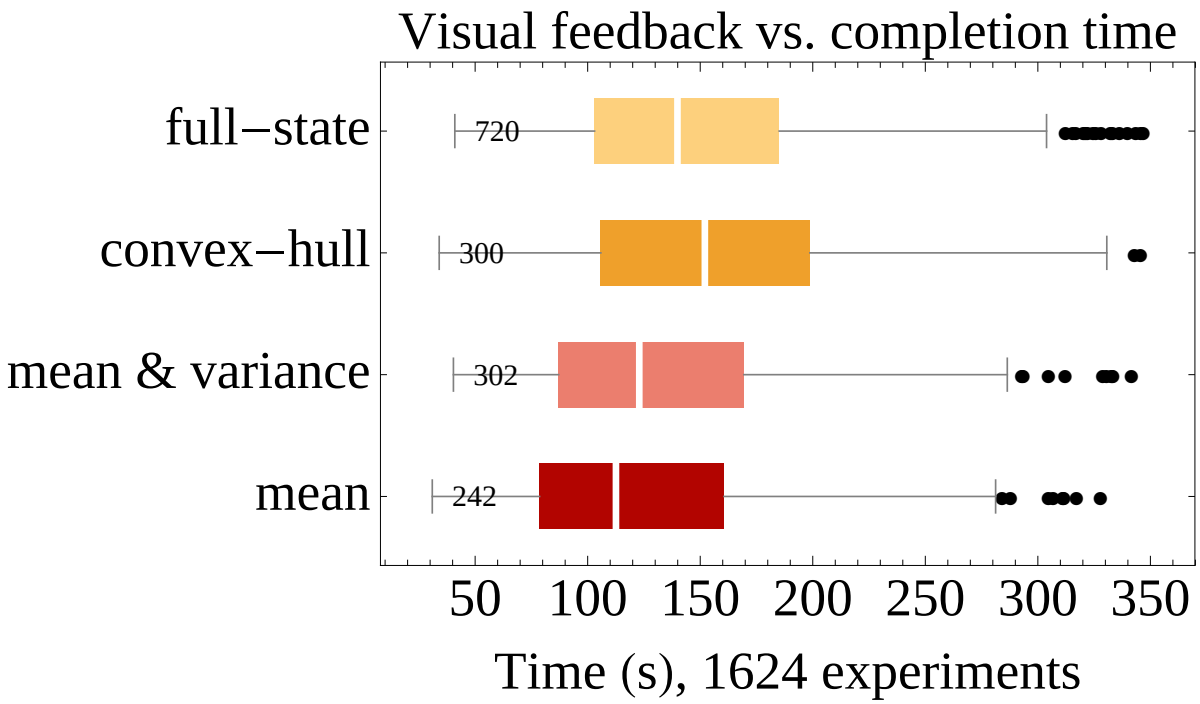

Figure 2.12: Completion-time results for the four levels of visual feedback shown in Fig. 2.11.

completed the task faster than those with full-state feedback. As Fig. 2.12 shows, the levels of feedback arranged by increasing completion time are [mean, mean + variance, full-state, convex-hull]. Interviews with beta-testers suggests that tracking 100 robots was overwhelming—similar to schooling phenomenons that confuse predators—while working with just the mean + variance was like using a "spongy" manipulator. Convex-hull feedback was confusing and irritating because a single robot left behind an obstacle would distort the entire hull, obscuring the information about the majority of the swarm.

2.5.2 Automated block-pushing

Fig. 2.13 shows snapshots during an execution of this algorithm. To solve this block-pushing task, we discretized the environment. On this discretized grid we used breadth-first search to determine $\mathbf{M}$, the shortest distance from any grid cell to the goal, and generated a gradient map $\nabla\mathbf{M}$ toward the goal as shown in Fig. 2.14. The block's center of mass is at $\mathbf{b}$ and has radius r_b. Three constants are needed, where $k_1 > k_2 > 1$

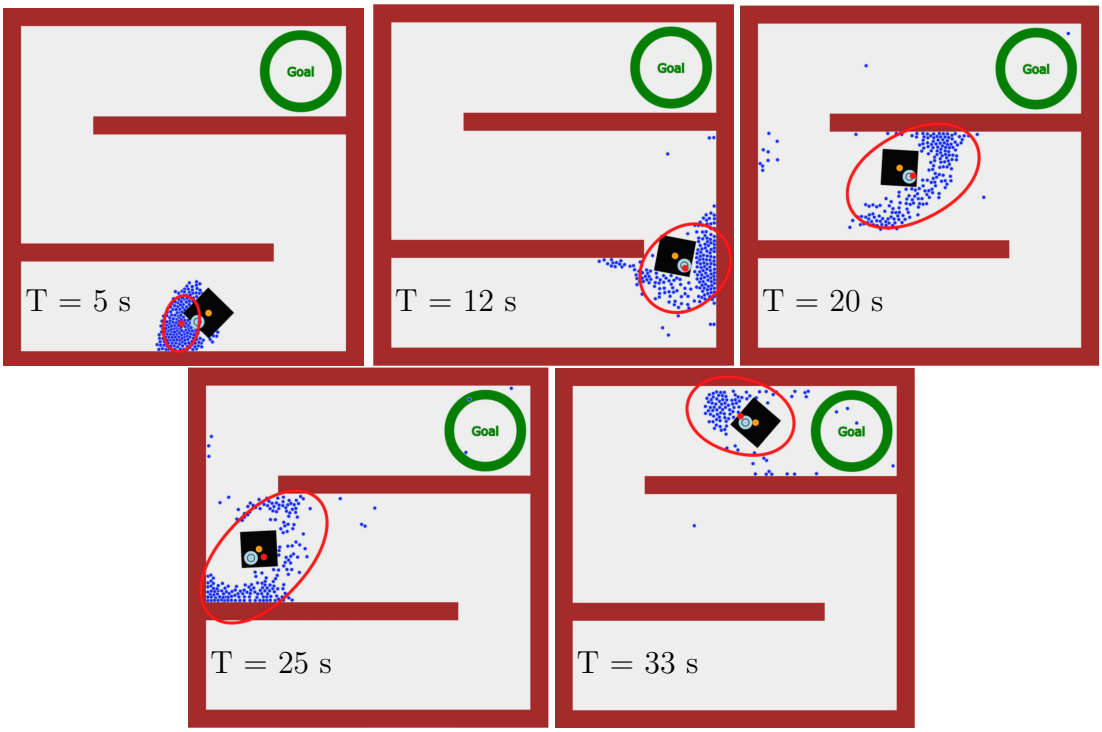

Figure 2.13: Snapshots showing the block-pushing experiment with 200 robots under automatic control.

and $1 > k_2 > 0$. All experiments used $[k_1, k_2, k_3] = [2.5, 1.5, 0.1]$. The robots were directed to assemble behind the block at $\mathbf{b} - k_2 r_b \nabla \mathbf{M}(\mathbf{b})$, then move to $\mathbf{b} - k_3 r_b \nabla \mathbf{M}(\mathbf{b})$ to push the block toward the goal location. We use the hybrid hysteresis-based controller in Alg. 1 to track the desired position, while maintaining sufficient robot density to move a block by switching to minimize variance whenever variance exceeds a set limit. The minimize variance control law (2.20) is slightly modified to choose the nearest corner further from the goal than $\mathbf{b}$ with an obstacle-free straight-line path to $\mathbf{b}$. The control algorithm for block-pushing is listed in Alg. 5. Experimental results are summarized in Fig. 2.15. Although larger populations of robots can apply more force, minimizing the variance requires more time with larger populations and dominates task completion time.

Algorithm 5 is an imperfect solution and has a failure mode if the robot swarm becomes multi-modal with modes separated by an obstacle, as shown in Fig. 2.16. In this case, moving toward a corner will never reduce the variance below σ^2_{min}.

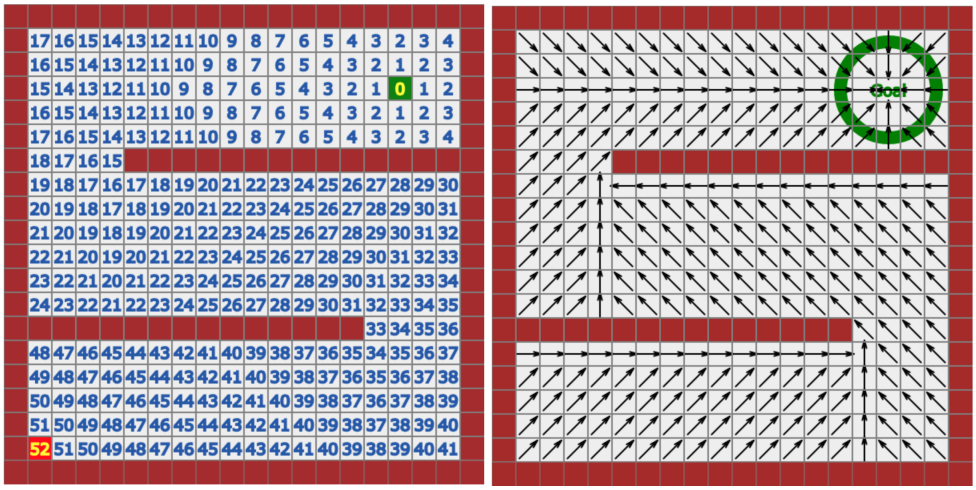

Figure 2.14: The BFS algorithm finds the shortest path for the moveable block (left), which is used to compute gradient vectors (right).

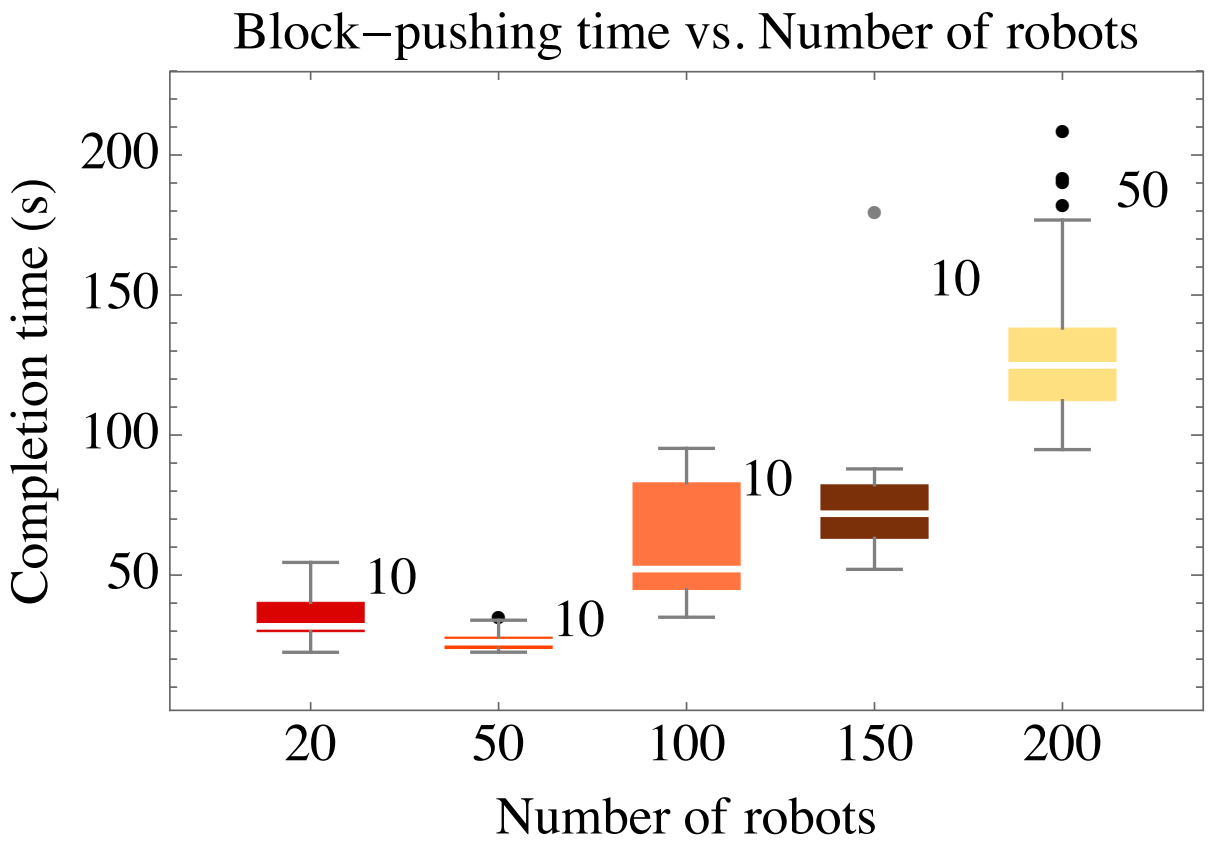

Figure 2.15: Completion-time results using the automatic controller from Alg. 5 for different numbers of robots. Each bar is labelled with the number of trials.

Algorithm 5 Block-pushing controller for a robotic swarm.

Require: Knowledge of swarm mean $[\bar{x}, \bar{y}]$, variance $[\sigma_x^2, \sigma_y^2]$, moveable block's center of mass $\mathbf{b}$, map of the environment, and the locations of all convex corners $\mathbf{C}$
Require: Robot distribution is unimodal
Require: Obstacle-free, straight-line path from swarm to moveable block

Compute $\mathbf{M}$, the distance to goal, with breadth-first search
Compute the gradient, $\nabla\mathbf{M}$
$\mathbf{C} \leftarrow \text{sort}(\mathbf{C})$ according to $-\mathbf{M}$
while $\mathbf{b}$ is not in goal region **do**
    $\sigma^2 \leftarrow \max(\sigma_x, \sigma_y)$
    if $\sigma^2 > \sigma_{max}^2$ **then**
        while $\sigma^2 > \sigma_{min}^2$ **do**
            $\mathbf{c}_i \leftarrow$ the nearest corner in $\mathbf{C}$ to $[\bar{x}, \bar{y}]$
            $[x_{goal}, y_{goal}] \leftarrow \mathbf{c}_i$
            if $\mathbf{M}(\mathbf{b}) > \mathbf{M}(\mathbf{c}_i)$ **then**
                $[x_{goal}, y_{goal}] \leftarrow \mathbf{c}_{i-1}$
                Apply (2.19) to move toward $[x_{goal}, y_{goal}]$
            end if
        end while
    else
        if distance$(\mathbf{b}, [x_{goal}, y_{goal}]) > k_1 r_b$ **then**
            $r_p \leftarrow k_2 r_b$ ▷ guarded move
        else
            $r_p \leftarrow k_3 r_b$ ▷ pushing move
        end if
        $[x_{goal}, y_{goal}] \leftarrow \mathbf{b} - r_p \nabla\mathbf{M}(\mathbf{b})$
    end if
    Apply (2.19) to move toward $[x_{goal}, y_{goal}]$
end while

The first challenge is to identify when the distribution has become multi-modal. Measuring just the mean and variance is insufficient to determine if a distribution is no longer unimodal, but if the swarm is being directed to a corner, and the variance does not reduce below σ_{min}^2, the swarm has become separated. In this case, we must either manipulate with a partial swarm, or run a gathering algorithm. For the 'S'-shaped workspace in this study, an open-loop input that commands the swarm to move in succession {LEFT, DOWN, RIGHT, DOWN} will move the swarm to the bottom right corner. This is not true for all obstacle fields. In a 'T'-shaped workspace, it is not possible to find an open-loop input that will move the entire swarm to the bottom of the 'T'.

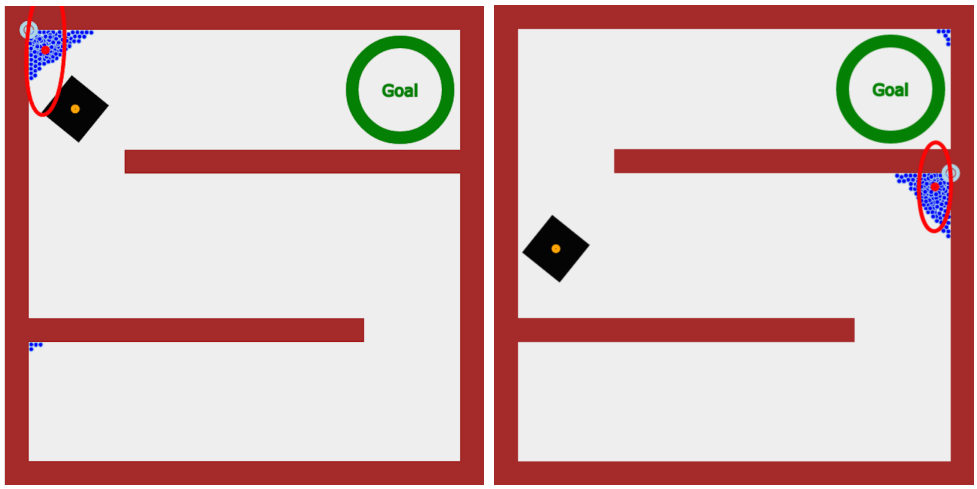

Figure 2.16: Algorithm 5 fails when some robots are separated by the maze and the swarm can not achieve $\sigma^2 < \sigma^2_{min}$. These failures occured during 14% of trials.

Using only the mean and variance may be overly restrictive. Many heuristics using high-order moments have been developed to test if a distribution is multimodal [36]. Often the sensor data itself, though it may not resolve individual robots, will indicate multi-modality. For instance CCD images reveal clusters of bacteria, and MRI scans show agglomerations of particles [37]. This data can be fitted with k-means or expectation maximization algorithms, and manipulation could be performed with the nearest swarm of sufficient size.

2.5.3 Hardware system

Our experiments are on centimeter-scale hardware systems called *kilobots*. These allows us to emulate a variety of dynamics, while enabling a high degree of control over robot function, the environment, and data collection. The kilobot [38, 39] is a low-cost robot designed for testing collective algorithms with large numbers of robots. It is available commercially or as an open source platform [40]. Each robot is approximately 3 cm in diameter, 3 cm tall, and uses two vibration motors to move on a flat surface at speeds up to 1 cm/s. Each robot has one ambient light sensor that is used to implement

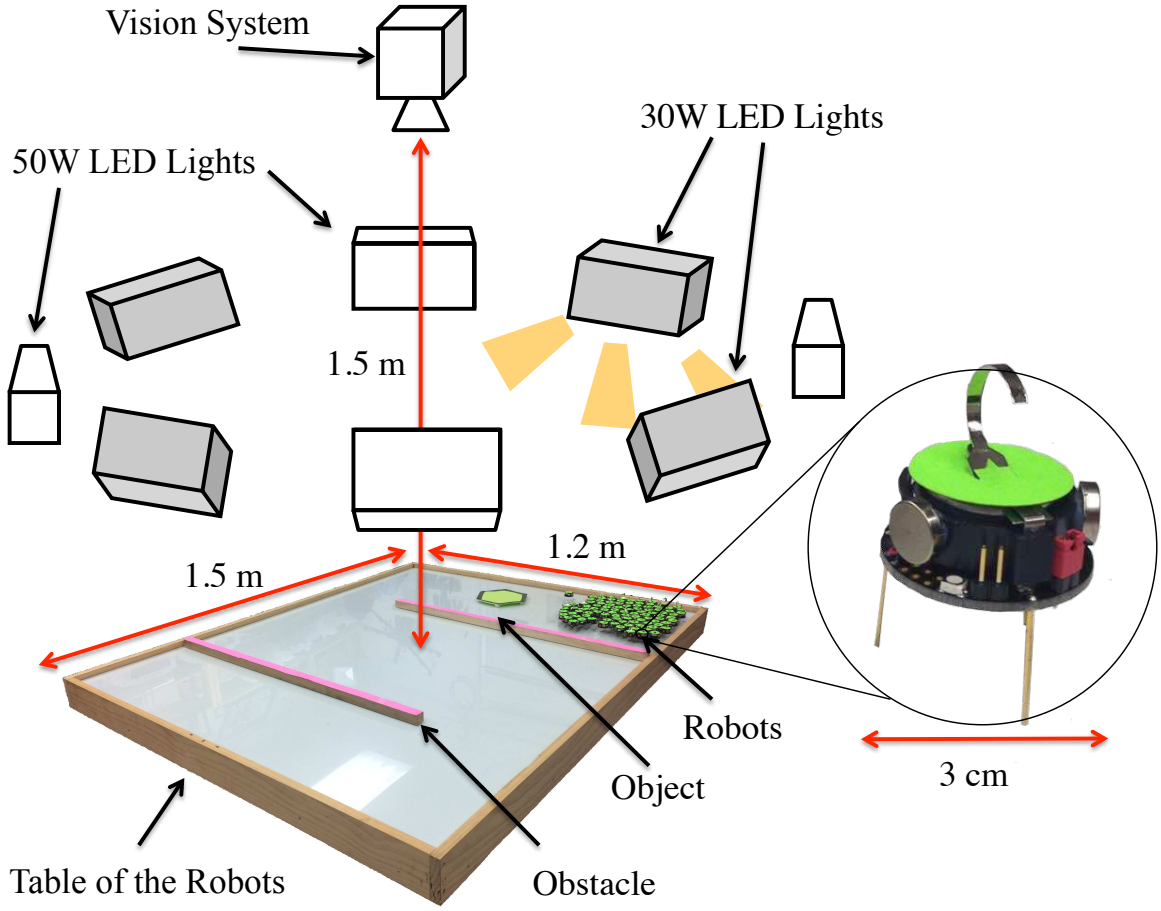

Figure 2.17: Hardware platform: table with 1.5×1.2 m workspace, surrounded by eight remotely triggered 30W LED floodlights, with an overhead machine vision system.

phototaxis, moving towards a light source. In these experiments as shown in Fig. 2.17, we used n=64 kilobots, a 1.5 m×1.2 m whiteboard as the workspace, and four 30W LED floodlights arranged 1.5 m above the plane of the table at the $\{N, E, S, W\}$ vertices of a 6 m square centered on the workspace. The lights were controlled using an Arduino Uno board connected to an 8 relay shield board. At top of the table, an overhead machine vision system was added to track the position of the swarm. Laser-cut patterns for our neon green fiducial markers and our MATLAB tracking code are available at our github repository [41].

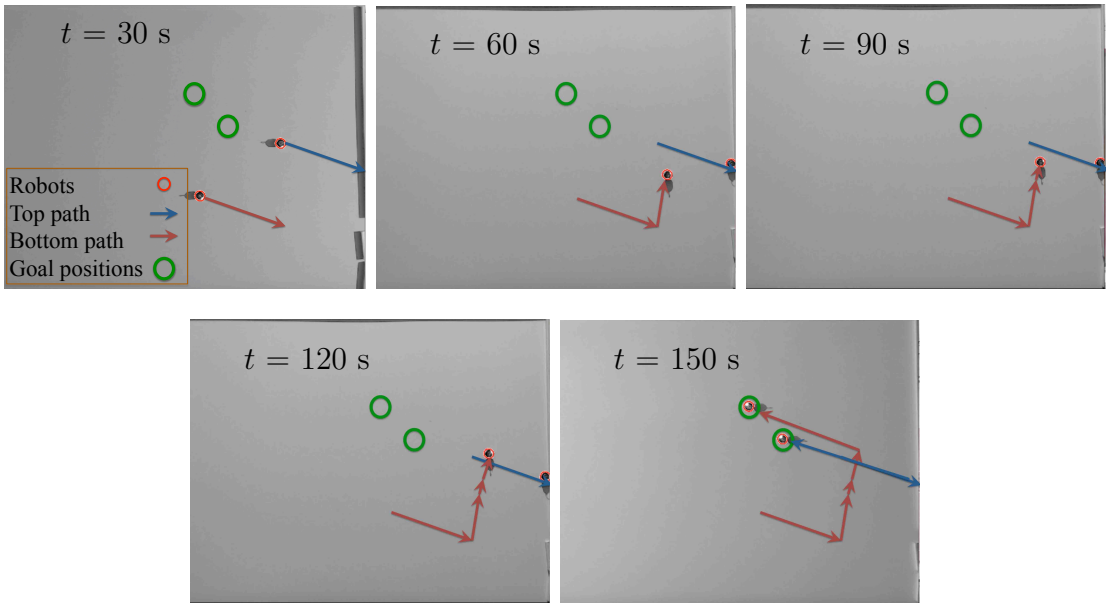

Figure 2.18: Two robot positioning using the hardware setup and two kilobot robots. The walls have nearly infinite friction, as illustrated by this fig, the robot with the blue path that is stopped by the wall until the light changes orientation, while the orange robot in free-space is unhindered.

The walls of the hardware platform have almost infinite friction, due to the three legged design of the kilobots. When a kilobot is steered into the wall, they pin themselves to the wall until the light changes direction and they begin turning in the other direction. This wall friction is sufficient to enable independent position control of two kilobots, as shown in Fig. 2.18.

To demonstrate covariance control $n = 64$ robots were placed on the workspace and manually steered with a single light source, using friction with the boundary walls to vary the covariance from -5000 to 10,000. The resulting covariance is plotted in Fig. 2.19, along with snapshots of the swarm.

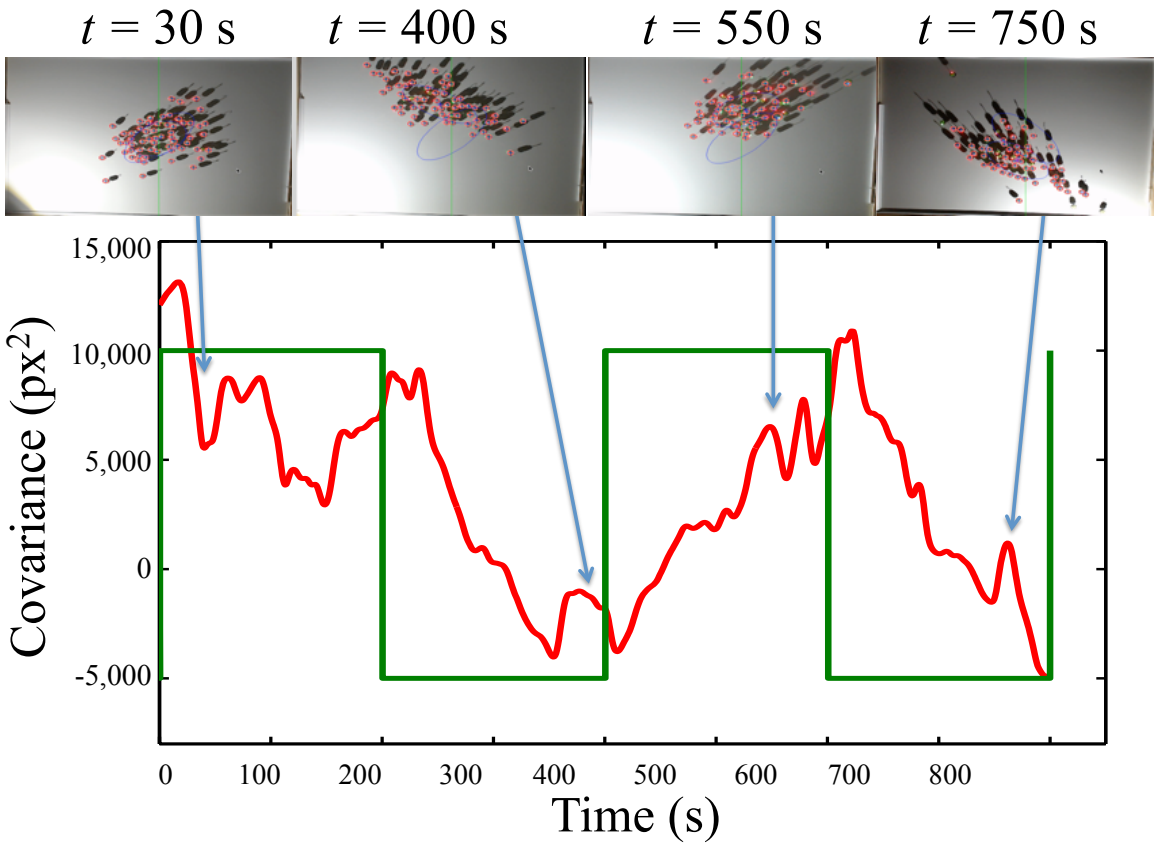

Figure 2.19: Hardware demonstration steering 64 kilobot robots to desired covariance. Frames above the plot show output from machine vision system and an overlaid covariance ellipse.

Chapter 3

Conclusion and Future Work

Inspired by large-scale human experiments with swarms of robots under global control, this thesis investigated controllers that use only the mean and variance of a robot swarm. We proved that the mean position is controllable, and provided conditions under which variance is controllable. We derived automatic controllers for each and a hysteresis-based switching control that controls the mean and variance of a robot swarm. We employed these controllers as primitives for a block-pushing task.

3.1 Future Work: Gathering Problem

Large populations of micro- and nanorobots are being produced in laboratories around the world, with diverse potential applications in drug delivery and construction. These activities require robots that behave intelligently. Limited computation and communication rules out autonomous operation or direct control over individual units; instead we must rely on global control signals broadcast to the entire robot population. It is not always practical to gather pose information on individual robots for feedback control; the robots might be difficult or impossible to sense individually due to their size and location. However, it is often possible to sense global properties of the group, such as mean position and variance. In our experiments, we showed that controlling mean is possible, and controlling variance with some constraints is possible. One of the constraints was that we had assumed that the swarm is clustered together and the swarm was trying to keep the variance small. If we did not assume that and we had a world like Fig. 3.1 or if during the experiment, a fraction of the swarm makes a new branch and put the swarm to two or more clusters by hitting the obstacles in their path, we could

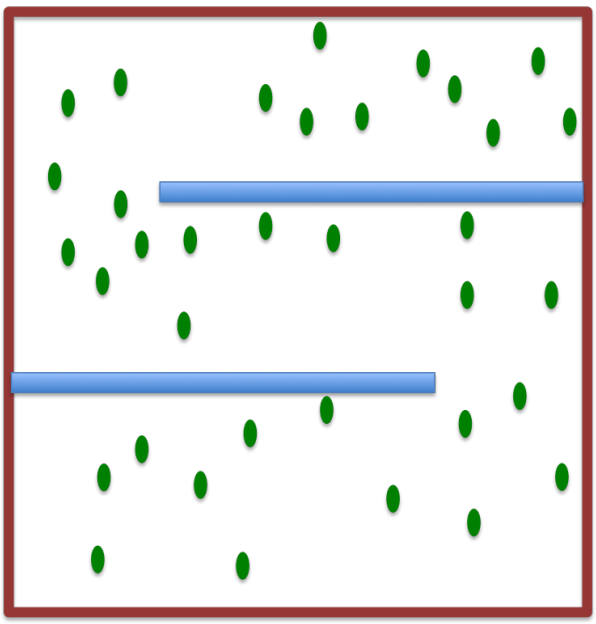

Figure 3.1: If the robots were not grouped initially, how can we gather them all together and what are the features of the map which is gather-able?

not gather them together again, and our algorithm for controlling variance was failed. So the next problem is to design an algorithm that will gather all the robots to an specific region with global input so that we can have the minimum variance.

For a S shaped maze it is easy to implement. Alg. 6 shows how to gather all the particles of the swarm when we have a S shaped maze. With the same algorithm, we have a solution for a mirrored S shaped maze. It is not an optimal way for a T shaped map also, but works for it. The problem is: Is there a general way to gather all the particles in *any* map?

Algorithm 6 Gather Robots in One Corner of an S-shaped or inverse S-shape Maze

Ensure: The maze is S-shaped
1: **while** $\sigma_x^2 > \sigma_{min}^2 \vee \sigma_y^2 > \sigma_{min}^2$ **do**
2: Go Maximal Left
3: Go Maximal Down
4: Go Maximal Right
5: Go Maximal Down
6: **end while**

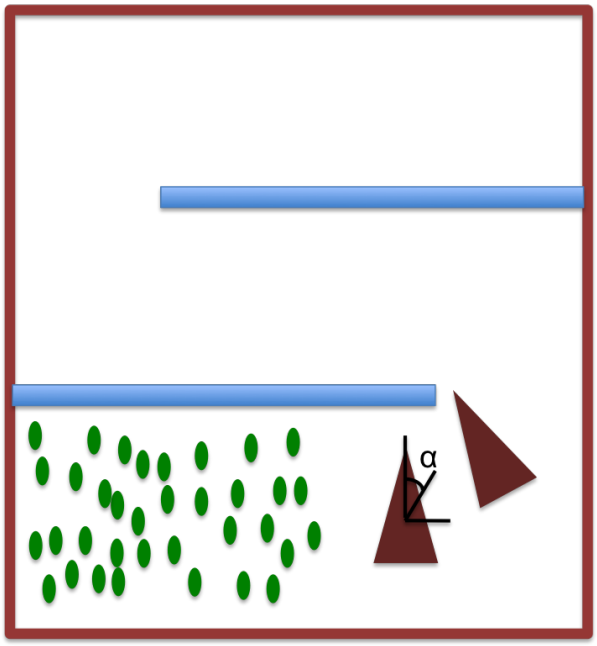

Figure 3.2: How can we control torque of the object with a swarm of robots and a shared input?

3.2 Future Work: Torque Control

Another remaining problem is that sometimes we do not want the block to rotate during the experiment, or we want to rotate it to some angle. The problem statement is how to control torque of the object, and discussing what features of the object make torque controllable. Fig. 3.2 shows an example of torque control.

References

[1] S. Martel, S. Taherkhani, M. Tabrizian, M. Mohammadi, D. de Lanauze, and O. Felfoul, "Computer 3d controlled bacterial transports and aggregations of microbial adhered nano-components," *Journal of Micro-Bio Robotics*, vol. 9, no. 1-2, pp. 23–28, 2014.

[2] M. Egerstedt, "Formation constrained multi-agent control," *IEEE Trans. Robotics Automat.*, vol. 17, pp. 947–951, 2001.

[3] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized controllers for shape generation with robotic swarms," *Robotica*, vol. 26, no. 05, pp. 691–701, 2008.

[4] S. Martel, "Magnetotactic bacteria for the manipulation and transport of micro-and nanometer-sized objects," *Micro-and Nanomanipulation Tools*, 2015.

[5] X. Yan, Q. Zhou, J. Yu, T. Xu, Y. Deng, T. Tang, Q. Feng, L. Bian, Y. Zhang, A. Ferreira, and L. Zhang, "Magnetite nanostructured porous hollow helical microswimmers for targeted delivery," *Advanced Functional Materials*, vol. 25, no. 33, pp. 5333–5342, 2015. [Online]. Available: http://dx.doi.org/10.1002/adfm.201502248

[6] A. Becker, G. Habibi, J. Werfel, M. Rubenstein, and J. McLurkin, "Massive uniform manipulation: Controlling large populations of simple robots with a common input signal," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2013, pp. 520–527.

[7] B. Donald, C. Levey, C. McGray, I. Paprotny, and D. Rus, "An untethered, electrostatic, globally controllable MEMS micro-robot," *J. of MEMS*, vol. 15, no. 1, pp. 1–15, Feb. 2006.

[8] B. Donald, C. Levey, and I. Paprotny, "Planar microassembly by parallel actuation of MEMS microrobots," *J. of MEMS*, vol. 17, no. 4, pp. 789–808, Aug. 2008.

[9] P.-T. Chiang, J. Mielke, J. Godoy, J. M. Guerrero, L. B. Alemany, C. J. Villagómez, A. Saywell, L. Grill, and J. M. Tour, "Toward a light-driven motorized nanocar: Synthesis and initial imaging of single molecules," *ACS Nano*, vol. 6, no. 1, pp. 592–597, Feb. 2011.

[10] D. Frutiger, B. Kratochvil, K. Vollmers, and B. J. Nelson, "Magmites - wireless resonant magnetic microrobots," in *IEEE Int. Conf. Rob. Aut.*, Pasadena, CA, May 2008.

[11] S. Tottori, L. Zhang, F. Qiu, K. Krawczyk, A. Franco-Obregón, and B. J. Nelson, "Magnetic helical micromachines: Fabrication, controlled swimming, and cargo transport," *Advanced Materials*, vol. 24, no. 811, 2012.

[12] K. E. Peyer, L. Zhang, and B. J. Nelson, "Bio-inspired magnetic swimming microrobots for biomedical applications," *Nanoscale*, 2013.

[13] S. Nunnally, P. Walker, A. Kolling, N. Chakraborty, M. Lewis, K. Sycara, and M. Goodrich, "Human influence of robotic swarms with bandwidth and localization issues," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, Oct 2012, pp. 333–338.

[14] A. Becker, C. Ertel, and J. McLurkin, "Crowdsourcing swarm manipulation experiments: A massive online user study with large swarms of simple robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. [Online]. Available: https://arxiv.org/submit/913241/view

[15] C. Ertel and A. Becker. (2013, Sep.) Swarmcontrol git repository. [Online]. Available: https://github.com/crertel/swarmmanipulate.git

[16] A. Sudsang, F. Rothganger, and J. Ponce, "Motion planning for disc-shaped robots pushing a polygonal object in the plane," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 550–562, Aug. 2002.

[17] J. Fink, N. Michael, and V. Kumar, "Composition of vector fields for multi-robot manipulation via caging," in *Robotics Science and Systems*, Atlanta, GA, Jun. 2007.

[18] K. Lynch, "Locally controllable manipulation by stable pushing," *IEEE Trans. Robot. Autom.*, vol. 15, no. 2, pp. 318–327, Apr. 1999.

[19] M. D. Armani, S. V. Chaudhary, R. Probst, and B. Shapiro, "Using feedback control of microflows to independently steer multiple particles," *Journal of Microelectromechanical systems*, vol. 15, no. 4, Aug. 2006.

[20] A. Becker, R. Sandheinrich, and T. Bretl, "Automated manipulation of spherical objects in three dimensions using a gimbaled air jet," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, oct. 2009, pp. 781 –786. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5354427

[21] L. U. Odhner, L. P. Jentoft, M. R. Claffee, N. Corson, Y. Tenzer, R. R. Ma, M. Buehler, R. Kohout, R. D. Howe, and A. M. Dollar, "A compliant, underactuated hand for robust manipulation," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 736–752, 2014.

[22] R. Deimel and O. Brock, "A novel type of compliant, underactuated robotic hand for dexterous grasping," *Robotics: Science and Systems, Berkeley, CA*, pp. 1687–1692, 2014.

[23] B. R. Munson, A. P. Rothmayer, T. H. Okiishi, and W. W. Huebsch, *Fundamentals of Fluid Mechanics*, 7th ed. Wiley, 2012.

[24] D. Spears, W. Kerr, and W. Spears, "Physics-based robot swarms for coverage problems," *The international journal of intelligent control and systems*, vol. 11, no. 3, 2006.

[25] A. Sudsang and L. E. Kavraki, "A geometric approach to designing a programmable force field with a unique stable equilibrium for parts in the plane," in *Proceedings of The 2001 IEEE International Conference on Robotics and Automation (ICRA 2001)*, vol. 2, IEEE Press. Seoul, Korea: IEEE Press, May 2001, inproceedings, pp. 1079–1085, this paper was a finalist for best conference paper award.

[26] T. Vose, P. Umbanhowar, and K. Lynch, "Friction-induced velocity fields for point parts sliding on a rigid oscillated plate," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 1020–1039, 2009. [Online]. Available: http://ijr.sagepub.com/content/28/8/1020.abstract

[27] T. H. Vose, P. Umbanhowar, and K. M. Lynch, "Sliding manipulation of rigid bodies on a controlled 6-dof plate," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 819–838, 2012.

[28] A. Becker, C. Onyuksel, T. Bretl, and J. McLurkin, "Controlling many differential-drive robots with uniform control inputs," *The international journal of Robotics Research*, vol. 33, no. 13, pp. 1626–1644, 2014.

[29] R. L. Graham and N. J. Sloane, "Penny-packing and two-dimensional codes," *Discrete & Computational Geometry*, vol. 5, no. 1, pp. 1–11, 1990.

[30] A. M. Lyapunov, translated and edited by A.T. Fuller, *The General Problem of the Stability of Motion*. London: Tayor & Francis, 1992.

[31] S. Sadraddini and C. Belta, "Swarm manipulation with a uniform magnetic field," in *unpublished*, 2015.

[32] M. Kloetzer and C. Belta, "Temporal logic planning and control of robotic swarms by hierarchical abstractions," *Robotics, IEEE Transactions on*, vol. 23, no. 2, pp. 320–330, 2007.

[33] E. Catto, "User manual, Box2D: A 2D physics engine for games, http://www.box2d.org," 2010. [Online]. Available: http://www.box2d.org

[34] S. Shahrokhi and A. T. Becker, "BlockPushingIROS2015, https://github.com/aabecker/swarmcontrolsandbox/blob/master/ examplecontrollers/blockpushingiros2015.html," Jul. 2015.

[35] A. Einstein, *Investigations on the Theory of the Brownian Movement*. Courier Corporation, 1956.

[36] J. Haldane, "Simple tests for bimodality and bitangentiality," *Annals of eugenics*, vol. 16, no. 1, pp. 359–364, 1951.

[37] M. Stuber, W. D. Gilson, M. Schär, D. A. Kedziorek, L. V. Hofmann, S. Shah, E.-J. Vonken, J. W. Bulte, and D. L. Kraitchman, "Positive contrast visualization of iron oxide-labeled stem cells using inversion-recovery with on-resonant water suppression (iron)," *Magnetic Resonance in Medicine*, vol. 58, no. 5, pp. 1072–1077, 2007.

[38] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *IEEE Int. Conf. Rob. Aut.*, May 2012, pp. 3293–3298.

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

[40] K-Team, "Kilobot, www.k-team.com/mobile-robotics-products/kilobot," 2015.

[41] S. Shahrokhi and A. T. Becker, ""Wall-Friction Swarm Shape Control", https://github.com/aabecker/swarmcontrolsandbox/ tree/master/papers/icra2016," Aug. 2015.