Steering a Particle Swarm Using Global Inputs and Swarm Statistics

Shiva Shahrokhi, Lillian Lin, Chris Ertel, Mable Wan, Aaron T. Becker

***Abstract*—This paper examines object manipulation by a swarm of particles, each actuated by the same shared global input. Microrobotics has the potential to revolutionize many applications including targeted material delivery, assembly, and surgery. The same properties that promise breakthrough solutions—small size and large populations—present unique challenges for controlling motion.**

Robotic manipulation usually assumes intelligent agents, not particle systems manipulated by a global signal. To identify the key parameters for particle manipulation, we used a collection of online games where players steer swarms of up to 500 particles to complete manipulation challenges. We recorded statistics from over ten thousand players. Inspired by techniques where human operators performed well, this paper next investigates controllers that use only the mean and variance of the swarm. We prove that the mean position is controllable and then provide conditions under which variance is controllable. We next derive automatic controllers for these and a hysteresis-based switching control to regulate the first two moments of the particle distribution. Finally, we employ these controllers as primitives for an object manipulation task and implement all the automatic controllers on 100 kilobots controlled by the direction of a global light source. This automatic controller uses policy iteration for path planning, handles outliers by partitioning the workspace, and minimizes pushing the object backwards using potential field navigation.

I. Introduction

Large populations of micro- and nanorobots are being produced in laboratories around the world, with diverse potential applications in drug delivery and construction, see [1]–[3]. 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 particle population. This paper examines object manipulation by a swarm of particles, each actuated by the same shared global input, as illustrated in Fig. 1.

Many promising applications for particle swarms require direct human control, but user interfaces to thousands—or millions—of particles is a daunting human-swarm interaction (HSI) challenge. Our early work with over a hundred hardware robots and thousands of simulated particles demonstrated that direct human control of large swarms is possible, [4]. Unfortunately, the logistical challenges of repeated experiments with over one hundred robots prevented large-scale tests. There is currently no comprehensive understanding of user interfaces for controlling multi-robot systems with massive populations.

This work was supported by the National Science Foundation under Grant No. [IIS-1553063].

Authors are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204 USA {sshahrokhi2, atbecker}@uh.edu

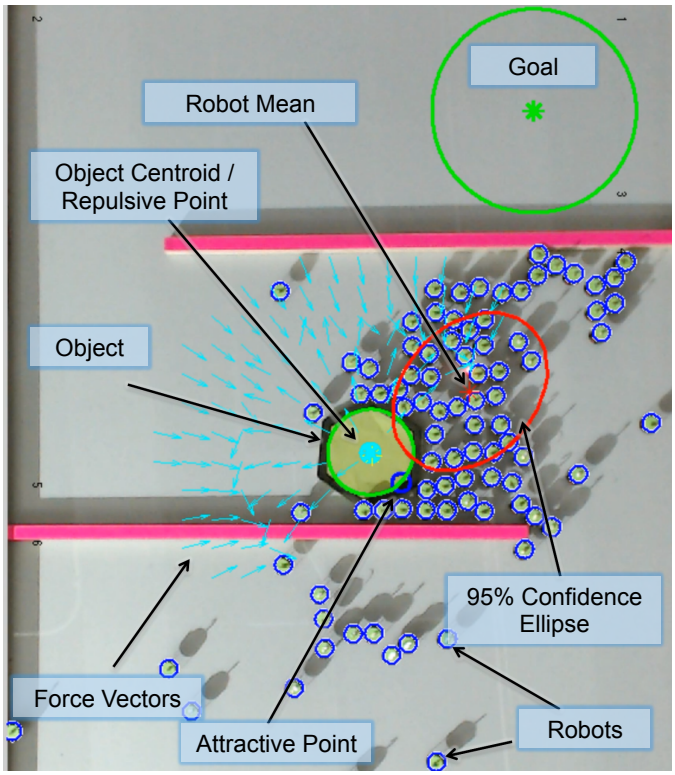

Fig. 1. A swarm of particles, all actuated by a uniform control input where each particle gets the same control input, can be effectively manipulated by a control law that uses only the mean and variance of the robot distribution. Here a swarm of particles (kilobot robots) pushes a green hexagon toward the goal (see video attachment).

One contribution of this paper is a tool for investigating HSI methods through statistically significant numbers of experiments.

Often particles are difficult or impossible to sense individually due to their size and location. For example, microrobots are smaller than the minimum resolution of a clinical MRI-scanner, see [5], 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, this paper presents swarm manipulation controllers inspired by our online experiments that require only mean and variance measurements of the particle's positions. These controllers are used as primitives to perform the object manipulation task illustrated in Fig. 1.

Our paper is organized as follows. After a discussion of related work in §II, we describe our experimental methods

for an online human-user experiment and their results in §III. Next we prove that the mean and variance of a particle swarm are controllable in §IV, and present automatic controllers in §V. We use these controllers as primitives and present a framework for manipulating an object through a maze in §VI. We conclude with implementations of these controllers in our hardware robots and use them to complete an object manipulation task with 100 kilobots in §VII.

II. Related Work

This section describes global control challenges and reviews highlights of human-swarm interaction, block pushing, and compliant manipulation.

A. Global control of microrobots

Small robots have been constructed with physical heterogeneity so that they respond differently to a global broadcast control signal. Examples include *scratch-drive microrobots*, actuated and controlled by a DC voltage signal from a substrate by [6], [7]; magnetic structures with different cross-sections that can be independently steered by [8], [9]; *MagMite* microrobots with different resonant frequencies and a global magnetic field by [10]; and magnetically controlled nanoscale helical screws constructed to stop movement at different cutoff frequencies of a global magnetic field by [11] and [1].

Similarly, our previous work, [12], [13], focused on exploiting inhomogeneity between robots. These control algorithms theoretically apply to any number of robots—even robotic continuums—but in practice process noise cancels the differentiating effects of inhomogeneity for more than tens of robots. We desire control algorithms that extend to many thousands of robots.

While it is now possible to create many microrobots, there remain challenges in control, sensing, and computation:

a) Control—global inputs: Many micro- and nanorobotic systems, see [1]–[3], [6]–[11], [14] rely on global inputs, where each robot receives an exact copy of the control signal. Our experiments follow this global model.

b) Sensing—large populations: n differential-drive robots in a 2D workspace require $3n$ state variables. Even holonomic robots require $2n$ state variables. Numerous methods exist for measuring this state in microrobotics [1], [3], [5]. These solutions use computer vision systems to sense position and heading angle, with corresponding challenges of handling missed detections and image registration between detections and robots. These challenges increase at small scales where sensing competes with control for communication bandwidth. We examine control when the operator has access to partial feedback, including only the first and/or second moments of a population's position, or only the convex-hull containing the robots.

c) Computation—calculating the control law: In our previous work the controllers required at best a summation over all the robot states, see [13] and at worst a matrix inversion, see [12]. These operations become intractable for large populations of robots. By focusing on *human* control of large robot populations, we accentuate computational difficulties because the controllers are implemented by the unaided human operator.

B. Human-swarm interaction

Most humans are able to, with practice, steer a swarm of robots controlled by a global input. Prior to our paper, no algorithm existed. Using human input to learn how to control a dynamic system is a line of research with a rich history [15], [16]. This paper exploits insights gained from "swarmcontrol.net", particularly the fact that having a swarm's mean and variance is sufficient for object manipulation through an obstacle field.

A user interface enabling an operator to maneuver a swarm of robots through a cluttered workspace by specifying the bounding prism for the swarm and then translating or scaling this prism is designed in [17]. Our paper shares the concept of a global control input, but our robots have no onboard computation and cannot track a virtual boundary.

Human *fanout*, the number of robots a single human user could directly control is studied in [18]. They postulated that the optimal number of robots was approximately the autonomous time divided by the interaction time required by each robot. Their sample problem involved a multi-robot search task, where users could assign goals to robots. Their user interaction studies with simulated planar robots indicated a *fanout plateau* of about 8 robots, with diminishing returns for more robots. They hypothesized the location of this plateau is highly dependent on the underlying task. Indeed, our paper indicates there are tasks without plateaus. Their research investigated robots with 3 levels of autonomy. We use robots without autonomy, corresponding with their first-level robots.

Several user studies compare methods for controlling large swarms of simulated robots, for example [19]–[21]. These studies provide insights but are limited by cost to small user studies; have a closed-source code base; and focus on controlling intelligent, programmable agents. For instance, the studies [19], [20], and [21] were limited to a pool of 5, 18, and 32 participants. Using an online testing environment, we conduct similar studies but with sample sizes three orders of magnitude larger.

C. Block pushing and compliant manipulation

Unlike *caging* manipulation, where robots form a rigid arrangement around an object, as in [22], [23], 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. [24], 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 as in [25] and [26].

Our n-robot system with 2 control inputs and $4n$ states is inherently under-actuated, and superficially bears resemblance to compliant, under-actuated manipulators. Our swarms conform to the object to be manipulated, but lack the restoring

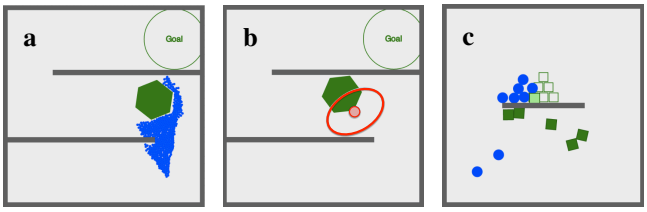

Fig. 2. Screenshots from our online experiments controlling multi-particle systems with limited, global control. **(a)** Varying the number of particles from 1-500 **(b)** Comparing 4 levels of visual feedback **(c)** Varying noise from 0 to 200% of control authority

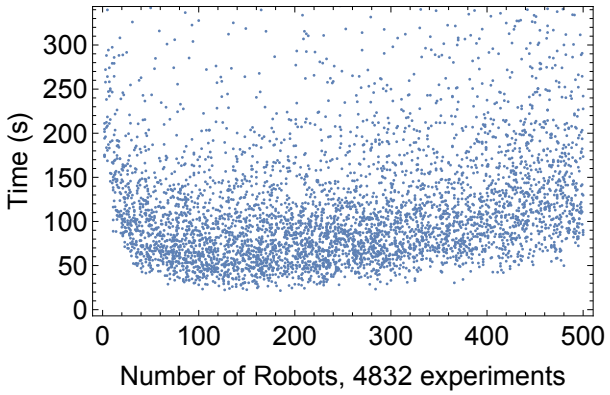

Fig. 3. Data from *Varying Number* using particles to push an object through a maze to a goal location.

force provided by flexures in [27] or silicone in [28]. Our swarms tend to disperse and so to regroup them we require artificial forces like the variance control primitives in §IV-C.

D. Relationship to authors' prior work

This paper combines the content of two preliminary conference papers, extending their substance and providing full details in a single journal paper. One paper covered the first three months of SwarmControl.net experiments [29], and the second presented simulations of object manipulation [30]. This paper presents three years of results from new games at SwarmControl.net. For object manipulation, this paper presents robust new algorithms for manipulation, path planning, and obstacle avoidance, and a rich set of parameter sweeps over key variables. All hardware validation experiments are new.

III. Online Experiment

The goal of these online experiments is to test several scenarios involving large-scale human-swarm interaction (HSI), and to do so with a statistically-significant sample size. Towards this end, we have created SwarmControl.net: an open-source, online testing platform suitable for inexpensive deployment and data collection on a scale not yet seen in swarm robotics research. Screenshots from this platform are shown in Fig. 2. All code and experimental results are online at [31].

We developed a flexible testing framework for online human-swarm interaction studies. Over 5,000 humans performed over 20,000 swarm-robotics experiments with this framework, logging almost 700 hours of experiments. These experiments indicated three lessons used for designing automatic controllers for object manipulation with particle swarms:

1) When the number of particles is large (> 50) varying the number of particles does not significantly affect the performance.

2) Swarm control is robust to IID noise.

3) Controllers that only use the mean and variance of the swarm can perform better than controllers with full feedback.

A. Implementation

Our web server generates a unique identifier for each participant and sends it along with the landing page to the participant. A script on the participant's browser runs the experiment and posts the experiment data to the server.

Anonymized human subject data was collected under IRB #14357-01.

We designed six experiments to investigate human control of large swarms for manipulation tasks. Screenshots of representative experiment are shown in Fig. 2. Each experiment examined the effects of varying a single parameter: population of particles for manipulation, four levels of visual feedback, different levels of Brownian noise.The users could choose which experiment to try, and our architecture randomly assigned a parameter value for each trial. We recorded the completion time and the participant ID for each successful trial.

B. Varying number

This experiment varied from 1 to 500 the population of particles used to transport an object. The total area, maximum particle speed, and total net force the swarm could produce were constant. The particles pushed a large hexagonal object through an S-shaped maze. We hypothesized participants would complete the task faster with more particles. The results, shown in Fig. 3, do not support our hypothesis, indicating a minimum around 130 particles, but only a gradual increase in completion time from 50 to 500.

C. Varying visualization

This experiment explores manipulation with varying amounts of sensing information: **full-state** sensing provides

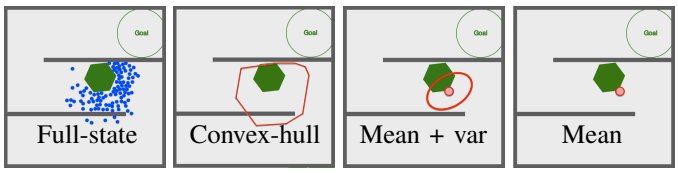

Fig. 4. Screenshots from task *Vary Visualization*. This experiment challenges players to quickly steer 100 particles (blue discs) to push an object (green hexagon) into a goal region. We record the completion time and other statistics.

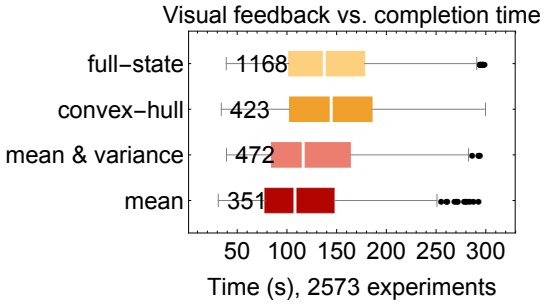

Fig. 5. Completion-time results for the four levels of visual feedback shown in Fig. 4. Players performed better with limited feedback.

the most information by showing the position of all particles; **convex-hull** draws a convex hull around the outermost particles; **mean** provides the average position of the population; and **mean + variance** adds a confidence ellipse. Fig. 4 shows screenshots of the same particle swarm with each type of visual feedback. Full-state requires $2n$ data points for n particles. Convex-hull requires at worst $2n$, but according to [32], the expected number is $O(2n^{1/3})$. Mean requires two, and variance three, data points. Mean and mean + variance are convenient even with millions of particles.

Our hypothesis predicted a steady decay in performance as the amount of visual feedback decreased. Our experiment indicated the opposite: players with just the mean completed the task faster than those with full-state feedback. As Fig. 5.b shows, the levels of feedback arranged by increasing completion time are [mean, mean + variance, full-state, convex-hull]. All experiments lasting over 300s were removed, under the assumption that the user stopped playing. Using ANOVA analysis, we rejected the null hypothesis that all visualization methods are equivalent, with p-value 2.69×10^{-19}. Anecdotal evidence from beta-testers who played the game suggests that tracking 100 particles is overwhelming—similar to schooling phenomenons that confuse predators—while working with just the mean + variance is like using a "spongy" manipulator. Our beta-testers described convex-hull feedback as confusing and irritating. A single particle left behind an obstacle will stretch the entire hull, obscuring the majority of the swarm. Similarly, our algorithm must be robust to outliers.

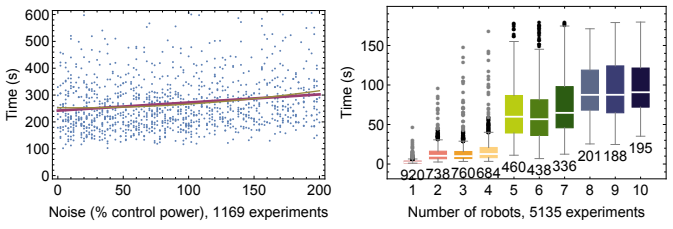

Fig. 6. Left: Varying the noise from 0 to 200% of the maximum control input resulted in only a small increase in completion time. Right: For position control, increasing the number of particles resulted in longer completion times. For more than 4 particles the goal pattern contained a void, which may have caused the jump in completion times.

D. Varying noise

This experiment varied the strength of disturbances to study how noise affects human control of large swarms. Noise was applied at every timestep:

$$\dot{x}_i = u_x + m_i \cos(\psi_i)$$
$$\dot{y}_i = u_y + m_i \sin(\psi_i).$$

m_i, ψ_i were uniformly IID, with $m_i \in [0, M]$ and $\psi_i \in [0, 2\pi]$. M was a constant for each trial ranging from 0 to 200% of the maximum control power ($u_{\max}$).

We hypothesized 200% noise was the largest a human could be expected to control—at 200% noise, the particles move erratically. Disproving our hypothesis, the results in Fig. 6.a show only a 40% increase in completion time for the maximum noise. This indicates swarm control is robust to IID noise.

IV. Global Control Laws for a Holonomic Swarm

Emboldened by the three lessons from our online experiments, this section presents automatic controllers for large numbers of particles that only rely on the first two moments of the swarm position distribution.

We represent particles as holonomic robots that move in the 2D plane. We want to control position and velocity of the particles. 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:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t). \tag{1}$$

and our state space representation is:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix}. \tag{2}$$

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

$$\mathcal{C} = [B, AB, A^2B, ..., A^{n-1}B]. \tag{3}$$

$$\text{Here } \mathcal{C} = \left[\begin{array}{cc|c|cc|cc|cc} 0 & 0 & \frac{1}{m} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{m} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right], \tag{4}$$

$$\text{rank}(\mathcal{C}) = 4, \tag{5}$$

and thus all four states are controllable. This section starts by proving independent position control of many robots is not possible, but the mean position can be controlled. We then provide conditions under which the variance of many robots is also controllable.

A. Independent control of many particles is impossible

In this model, a single particle is fully controllable. 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 particles to be controlled in the x axis, there are $2n$ states: n positions and n velocities. Without loss of generality, assume $m = 1$. Our state-space representation is:

$$\begin{bmatrix} \dot{x}_1 \\ \ddot{x}_1 \\ \vdots \\ \dot{x}_n \\ \ddot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ \vdots \\ x_n \\ \dot{x}_n \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u_x. \tag{6}$$

However, just as with one particle, we can only control two states because the controllability matrix $\mathcal{C}_n$ has rank two:

$$\mathcal{C}_n = \left[\begin{array}{c|c|c|c|c} 0 & 1 & 0 & 0 & \\ 1 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \dots \\ 0 & 1 & 0 & 0 & \\ 1 & 0 & 0 & 0 & \end{array}\right], \quad \text{rank}(\mathcal{C}_n) = 2. \tag{7}$$

B. Controlling the mean position

This means *any* number of particles controlled by a global command have just two controllable states in each axis. We cannot arbitrarily control the position and velocity of two or more robots, but have options on which states to control. We create the following reduced order system that represents the mean x position and velocity of the n particles:

$$\begin{bmatrix} \dot{\bar{x}} \\ \ddot{\bar{x}} \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 0 & 1 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ \vdots \\ x_n \\ \dot{x}_n \end{bmatrix} + \frac{1}{n} \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u_x. \tag{8}$$

Thus:

$$\begin{bmatrix} \dot{\bar{x}} \\ \ddot{\bar{x}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \dot{\bar{x}} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_x. \tag{9}$$

We again analyze the controllability matrix $\mathcal{C}_\mu$:

$$\mathcal{C}_\mu = \left[\begin{array}{c|c} 0 & 1 \\ 1 & 0 \end{array}\right], \quad \text{rank}(\mathcal{C}_\mu) = 2. \tag{10}$$

Thus the mean position and mean velocity are controllable.

There are several techniques for breaking the symmetry of the control control input to allow controlling more states, for example by using obstacles as in [4], or by allowing independent noise sources as in [33].

We control mean position with a PD controller that uses the mean position and mean velocity. $[u_x, u_y]^\top$ is the global force applied to each robot:

$$\begin{aligned} u_x &= K_p(x_{\text{goal}} - \bar{x}) + K_d(0 - \dot{\bar{x}}), \\ u_y &= K_p(y_{\text{goal}} - \bar{y}) + K_d(0 - \dot{\bar{y}}). \end{aligned} \tag{11}$$

K_p is the proportional gain, and K_d is the derivative gain.

C. Controlling the variance

The variance, σ_x^2, σ_y^2, of n robots' position is computed as:

$$\begin{aligned} \overline{x}(\mathbf{x}) &= \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \sigma_x^2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2, \\ \overline{y}(\mathbf{x}) &= \frac{1}{n}\sum_{i=1}^{n} y_i, \qquad \sigma_y^2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2. \end{aligned} \tag{12}$$

Controlling the variance requires being able to increase and decrease the variance. We will list a sufficient condition for each. Microscale systems are affected by unmodelled dynamics. These unmodelled dynamics are dominated by Brownian noise, described by [34]. To model this (1) must be modified as follows:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) + W\boldsymbol{\varepsilon}(t), \tag{13}$$

where $W\boldsymbol{\varepsilon}(t)$ is a random perturbation produced by Brownian noise with magnitude W. Given a large obstacle-free workspace with $\mathbf{u}(t) = 0$, a *Brownian noise* process increases the variance linearly with time.

$$\dot{\sigma}_x^2(\mathbf{x}(t), \mathbf{u}(t)) = W\boldsymbol{\varepsilon}, \quad \sigma_x^2(t) = \sigma_x^2(0) + W\boldsymbol{\varepsilon}t. \tag{14}$$

If faster dispersion is needed, the swarm can be pushed through obstacles such as a diffraction grating or Pachinko board as in [4].

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,

$$[\sigma_x^2, \sigma_y^2] = \frac{1}{12}[(\ell_x - 2r)^2, (\ell_y - 2r)^2]. \tag{15}$$

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, [35] showed that the minimum-variance packing for n circles with radius r is

$$\sigma^2_{\text{optimal}}(n, r) \approx \frac{\sqrt{3}}{\pi} nr^2 \approx 0.55nr^2. \tag{16}$$

We will prove the goal is globally asymptotically stabilizable by using a control-Lyapunov function, as in [36]. A suitable Lyapunov function is the squared variance error:

$$\begin{aligned} V(t, \mathbf{x}) &= \frac{1}{2}(\sigma^2(\mathbf{x}) - \sigma^2_{\text{goal}})^2, \\ \dot{V}(t, \mathbf{x}) &= (\sigma^2(\mathbf{x}) - \sigma^2_{\text{goal}})\dot{\sigma}^2(\mathbf{x}). \end{aligned} \tag{17}$$

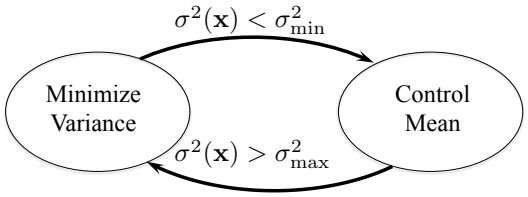

Fig. 7. Hysteresis to control swarm mean and variance.

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

$$u(t) = \begin{cases} \text{move to wall} & \text{if } \sigma^2(\mathbf{x}) > \sigma^2_{\text{goal}} \\ \text{move from wall} & \text{if } \sigma^2(\mathbf{x}) \leq \sigma^2_{\text{goal}}. \end{cases} \quad (18)$$

For such a $u(t)$,

$$\dot{\sigma}^2(\mathbf{x}) = \begin{cases} \text{negative} & \text{if } \sigma^2(\mathbf{x}) > \max(\sigma^2_{\text{goal}}, \sigma^2_{\text{optimal}}(n, r)) \\ W\epsilon & \text{if } \sigma^2(\mathbf{x}) \leq \sigma^2_{\text{goal}}, \end{cases} \quad (19)$$

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

A PID controller to regulate the variance to σ^2_{ref} is:

$$\begin{aligned} u_x &= K_p(x_{\text{goal}}(\sigma^2_{\text{ref}}) - \bar{x}) - K_d \bar{v}_x + K_i(\sigma^2_{\text{ref}} - \sigma^2_x), \\ u_y &= K_p(y_{\text{goal}}(\sigma^2_{\text{ref}}) - \bar{y}) - K_d \bar{v}_y + K_i(\sigma^2_{\text{ref}} - \sigma^2_y). \end{aligned} \quad (20)$$

We call the gain scaling the variance error K_i because the variance, if unregulated, integrates over time. Eq. (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 (15):

$$x_{\text{goal}}(\sigma^2_{\text{ref}}) = \ell_x/2 = r + \sqrt{3\sigma^2_{\text{ref}}}. \quad (21)$$

If a wall to the right is closer, the signs of $[K_p, K_i]$ are inverted, and the location x_{goal} is translated.

D. Controlling both mean and variance

The mean and variance of the swarm cannot be controlled simultaneously, however if the dispersion due to Brownian motion is less than the maximum controlled speed, we can adopt the hybrid, hysteresis-based controller shown in Alg. 1 to regulate the mean and variance. Such a controller normally controls the mean position, but switches to minimizing variance if the variance exceeds some $\sigma^2_{\max}$. Variance is reduced until less than $\sigma^2_{\min}$, then control again regulates the mean position. This technique satisfies control objectives that evolve at different rates as in [37], and the hysteresis avoids rapid switching between control modes. The process is illustrated in Fig. 7.

A key challenge is to select proper values for $\sigma^2_{\min}$ and $\sigma^2_{\max}$. The optimal packing variance was given in (16). The random packings generated by pushing our robots into corners are suboptimal, so we choose the conservative values:

$$\begin{aligned} \sigma^2_{\min} &= 2.5r + \sigma^2_{\text{optimal}}(n, r), \\ \sigma^2_{\max} &= 15r + \sigma^2_{\text{optimal}}(n, r). \end{aligned} \quad (22)$$

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 $[x_{\text{target}}, y_{\text{target}}]$.

$x_{goal} \leftarrow x_{\text{target}}$, $y_{\text{goal}} \leftarrow y_{\text{target}}$
loop
  if $\sigma^2_x > \sigma^2_{\max}$ **then**
    $x_{\text{goal}} \leftarrow x_{\min}$
  else if $\sigma^2_x < \sigma^2_{\min}$ **then**
    $x_{\text{goal}} \leftarrow x_{\text{target}}$
  end if
  if $\sigma^2_y > \sigma^2_{\max}$ **then**
    $y_{\text{goal}} \leftarrow y_{\min}$
  else if $\sigma^2_y < \sigma^2_{\min}$ **then**
    $y_{\text{goal}} \leftarrow y_{\text{target}}$
  end if
  Apply (11) to move toward $[x_{\text{goal}}, y_{\text{goal}}]$
end loop

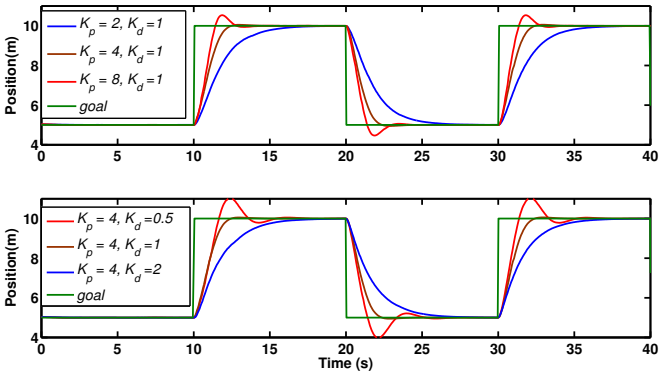

Fig. 8. In simulation, tuning proportional (K_p, top) and derivative (K_d, bottom) gain values in (11) improves performance with $n = 100$ particles.

V. Simulation of Control Laws

Our simulations use a Javascript port of Box2D, a popular 2D physics engine with support for rigid-body dynamics and fixed-time step simulation, presented in [38]. All experiments in this section ran on a Chrome web browser on a 2.6 GHz Macbook. All code is available at [39].

A. Controlling the mean position

We performed a parameter sweep using the PD controller (11) to identify the best control gains. Representative experiments are shown in Fig. 8. 100 particles were used and the maximum speed was 3 meters per second. As shown in Fig. 8, we can achieve an overshoot of 1% and a rise time of 1.52 s with $K_p = 4$, and $K_d = 1$.

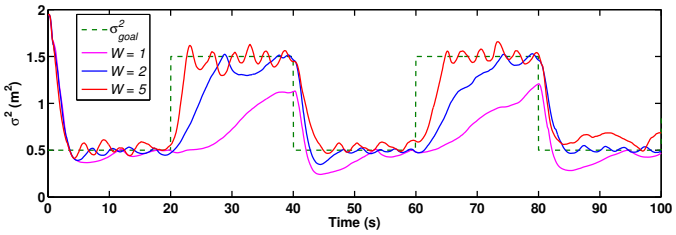

Fig. 9. In simulation, increased noise results in more responsive variance control because stronger Brownian noise causes a faster increase of variance.

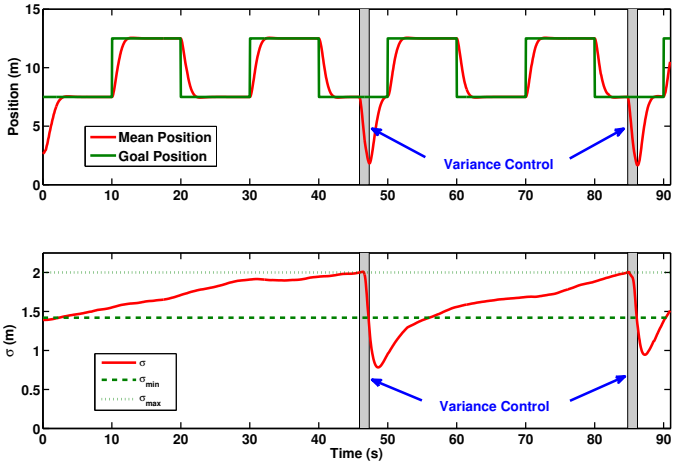

Fig. 10. Simulation result with 100 particles under hybrid control Alg. 1, which controls both the mean position (top) and variance (bottom). For ease of analysis, only x position and variance are shown.

B. Controlling the variance

For variance control we use the control law (20). Results are shown in Fig. 9, with $K_{p,i,d} = [4, 1, 1]$.

C. Hybrid control of mean and variance

Fig. 10 shows a simulation run of the hybrid controller in Alg. 1 with 100 particles in a square workspace containing no internal obstacles.

VI. Particle Swarm Object Manipulation

This section analyzes an *object manipulation* task attempted by our hybrid, hysteresis-based controllers. The swarm must deliver the object to the goal region. To solve this object manipulation task we divide the task into three components: 1) designing a policy for the object, 2) pushing the object with a compliant swarm, and 3) managing outliers.

The table below summarizes the simulation results for each 10 successful trials:

Method	Result, mean±std (s)
Value Iteration (VI)	367 ± 253
VI + Potential Fields (PF)	271 ± 267
VI + Outlier Rejection (OR)	245 ± 135
BFS + PF + OR	183 ± 179
VI + PF + OR	90 ± 35

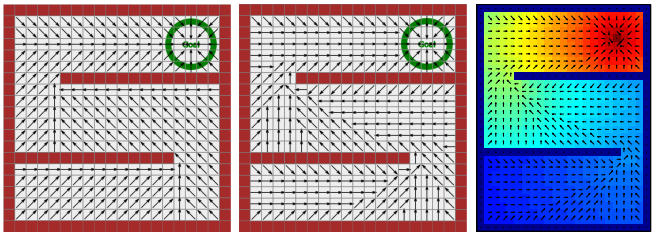

Fig. 11. BFS finds the shortest path for the moveable object to compute gradient vectors (left). Modeling as an MDP enables encoding penalties for being near obstacles. (Middle) The control policy from value iteration. (Right) The vision algorithm detects obstacles in the hardware setup. This map is used to produce the value function and control policy shown.

A. Learning a policy for the object

To design the policy we first discretize the environment. In [30], we used breadth-first search (BFS) on this discretized grid, but using workspace BFS fails to account for the hull of the object and will suggest moves that can cause collisions with the workspace. A configuration-space BFS approach avoids that problem but still fails to model uncertain actuation of the object by the swarm.

To solve both these problems, this paper models object movement as a Markov Decision Process (MDP) with non-deterministic movement. Value iteration, as described in [40], is used to learn an *optimal policy*. At each state the object can be commanded to move in one of eight directions with a small probability of moving in a wrong direction.

The reward function $r(x, \mathbf{u})$ is defined as

$$r(x,\mathbf{u}) = \begin{cases} +100, & \text{if } \mathbf{u} \text{ leads to goal state} \\ -100, & \text{if } \mathbf{u} \text{ leads to an obstacle state} \\ -1, & \text{otherwise} \end{cases} \quad (23)$$

where x is the current state and $\mathbf{u}$ is the action. Value iteration computes $\hat{V}(x)$, the expected discounted sum reward if the optimal policy is implemented, for the object starting in each state x. The optimal policy is

$$\mathbf{D}(x) = \arg\max_{\mathbf{u}} [r(x,\mathbf{u}) + \sum_{j=1}^{N} \hat{V}(x_j) p(x_j|x,\mathbf{u})]. \quad (24)$$

The value function $\hat{V}(x_j)$ is calculated by computing the value $\hat{V}$ for all N states and iterating until convergence:

for $i = 1$ to N do

$$\hat{V}(x_j) = \gamma \max_{\mathbf{u}} [r(x_j,\mathbf{u}) + \sum_{i=1}^{N} \hat{V}(x_i) p(x_i|x_j,\mathbf{u})]$$

end (25)

In our experiments $\gamma = 0.97$, and (25) was iterated 200 times. A MATLAB implementation of this algorithm is available at [41].

$\mathbf{M}_{\text{BFS}}$ and the value function are shown in Fig. 11. In 10 simulations with 100 particles, pushing the object to goal using BFS required 183±179 s while Policy Iteration required 90±35 s (mean±std).

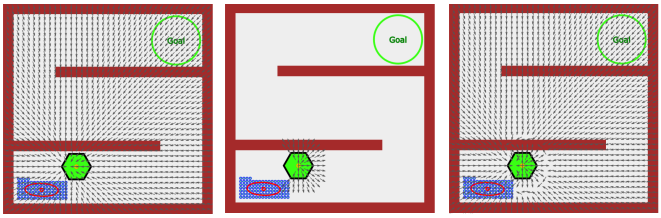

Fig. 12. (Left) The attractive field is centered behind the object's COM. (Middle) The repulsive field is centered at the object's COM. (Right) Combining these forces prevents the swarm from pushing the object backwards.

B. Potential fields for swarm management with a compliant manipulator

When the swarm is in front of the object, control law (11) pushes the object backwards. To fix this, we implement a potential field approach inspired by [42] that attracts the swarm to the intermediate goal, but repulses the swarm from in front of the object. The repulsive potential field is centered at the object's COM and is active for a radius ρ_0, but is implemented only when the swarm mean is within θ of the desired direction of motion $\mathbf{D}(\mathbf{b})$ as shown in Fig. 12b.

$$F_{\text{att}} = -\zeta \Delta\rho/\rho, \tag{26}$$

$$\phi = \cos^{-1}\left(\frac{\mathbf{D}(\mathbf{b}) \cdot ([\bar{x},\bar{y}] - \mathbf{b})}{\|\mathbf{D}(\mathbf{b})\| \, \|[\bar{x},\bar{y}] - \mathbf{b}\|}\right), \tag{27}$$

$$F_{\text{rep}} = \begin{cases} \eta(1/\rho - 1/\rho_0)\frac{1}{\rho^2}\Delta\rho, & \rho \le \rho_0 \ \wedge \ \phi < \theta \\ 0, & \text{otherwise} \end{cases}, \tag{28}$$

$$F_{\text{pot}} = F_{\text{att}} + F_{\text{rep}}. \tag{29}$$

In simulations, $\theta = \pi/2$, $\eta = 75$, $\zeta = 2$ and $\rho_0 = 3$. Because the kilobots have a slower time constant, they use $\theta = \pi/2$, $\eta = 50$, $\zeta = 1$ and $\rho_0 = 7.5$.

In 10 simulations with 100 particles, pushing the object to goal without a repulsive potential field failed in two of twelve runs. No failures occurred with the repulsive potential field. Of successful trials, completion time without repulsive potential fields required 245±135 s while using repulsive potential fields required 90±35 s (mean±std).

C. Outlier rejection

The variance controller in Alg. 1 is a greedy algorithm that is susceptible to outliers. The controller in [30] failed in 14% trials, some particles were unable to reach the object because workspace obstacles were blocking them. This failure rate increases if object weight increases or ground-robot friction increases. The mean and covariance calculations (12) included all particles in the workspace. Particles that cannot reach the object due to obstacles skew these calculations. The state machine in Fig. 13a solves this problem by creating two states for the maze: either main or transfer. Each state has a set of regions representing a discretized visibility polygon. Whenever the object crosses a region boundary the state toggles. The *main* regions are generated by extending obstacles until they meet another obstacle. The *transfer* regions are perpendicular to obstacle boundaries, and act as a buffer between two main regions.

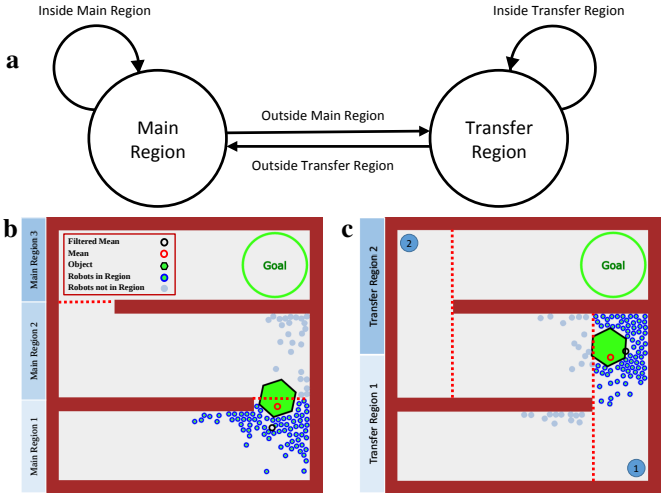

Fig. 13. Outlier rejection state machine and regions.

Fig. 13b shows the regions for the main state. The object is in region 1. An indicator function is applied to (12) so only robots inside region 1 are counted. This filtering increases experimental success because the mean calculation only includes nearby robots that can directly interact with the object. When the object leaves main region 1 the state switches to transfer. The transfer regions are shown in Fig. 13c. The object is in transfer region 1, so only robots in transfer region 1 are included in the mean and covariance calculations. The robots should push the object to the left. Without filtering using regions, the red circle is the mean and the algorithm would instruct the robots to push the object up. The black circle shows the filtered mean and the algorithm instructs the robots to push the object directly left.

In 10 simulations with 100 robots, completion time without outlier rejection required 271±267 s while using outlier rejection required 90±35 s (mean±std).

D. Simulation results

We use the hybrid hysteresis-based controller in Alg. 1 to track the desired position, while maintaining sufficient robot density to move the object by switching to minimize variance whenever variance exceeds a set limit: $0.003W$ and $0.006W$ were added to the min and max variance limits from (22). The minimize variance control law (20) is slightly modified to choose the nearest corner further from the goal than the object with an obstacle-free straight-line path to the object. The control algorithm for object manipulation is listed in Alg. 2.

In rare cases during simulations the swarm may become trapped in a local minimum of (29). If the swarm mean position does not change for five seconds, the swarm is assumed to be in a local minimum and is commanded to

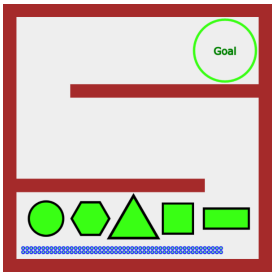

Fig. 14. The six equal-area objects tested in simulation.

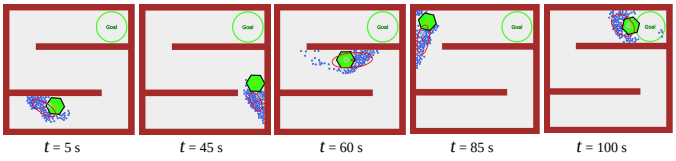

Fig. 15. Snapshots showing an object manipulation simulation with 100 robots under automatic control (see also Extension 1).

move toward the previous corner. As soon as the mean position changes, normal control resumes.

Algorithm 2 Object-manipulation controller for a robotic swarm.

Require: Knowledge of moveable object's center of mass $\mathbf{b}$; swarm mean $[\bar{x}, \bar{y}]$ and variance $[\sigma_x^2, \sigma_y^2]$, each calculated using the regions function from §VI-C; map of the environment

Compute optimal policy for object, according to §VI-A
while is not in goal region **do**
  $\sigma^2 \leftarrow \max(\sigma_x, \sigma_y)$
  if $\sigma^2 > \sigma^2_{\max}$ **then**
    while $\sigma^2 > \sigma^2_{\min}$ **do**
      $[x_{\text{goal}}, y_{\text{goal}}] \leftarrow$ nearest corner in region
      Apply (11) to move toward $[x_{\text{goal}}, y_{\text{goal}}]$
    end while
  else
    Calculate $\mathbf{D}(\mathbf{b})$ ▷ direction for object at $\mathbf{b}$
    Apply (29) ▷ potential field for swarm
  end if
end while

Fig. 15 shows snapshots during an execution of this algorithm in simulation. Experimental results of parameters sweeps are summarized in Fig. 16. Each trial measured the time to deliver the object to the goal location. The default parameter settings used 100 robots, a normalized weight of 1, a hexagon shape, and Brownian noise (applied once each simulation step) with $W = 5$.

The interaction between the robots and object is impulsive so, like the study of impulsive pulling in [43], adding additional robots decreases completion time, but with diminishing returns. After 75 robots, additional robots no longer can interact with the object and do not contribute to the task success. Brownian noise adds stochasticity. This randomness can break the object free if it is stuck, but diminishes the effect of the control input. Increasing noise increases completion time. The robots have limited force, so increasing the object weight increases completion time. Each shape was designed to have the same mass and area. Rectangles and squares tend to get stuck in the 90° workspace corners, and cause longer completion times than circles, triangles, and hexagons.

VII. Object Manipulation with Hardware Robots

Our experiments use centimeter-scale hardware systems called *kilobots*. While those are far larger than the micro scale devices we model, using kilobots 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, reported in [44], [45], is a low-cost robot designed for testing collective algorithms with large numbers of robots. It is available as an open-source platform or commercially from [46]. 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 *phototaxis*, moving towards a light source.

A. Environmental setup

In these experiments as shown in Fig. 17, we used n=100 kilobots, a 1.5 m×1.2 m whiteboard as the workspace, and lights: four 50W LED floodlights at the corners and four 30W LED floodlights on the sides of a 6 m square centered on the workspace and 1.5 m above the table. An Arduino Uno connected to an 8-relay shield controlled the lights.

Above the table, an overhead machine vision system tracks the swarm. The vision system identifies obstacles by color segmentation, determines the corners (used to decrease variance), the object by color segmentation, and identifies robots using color segmentation and circle detection with a circular Hough transform.

The objects were 3D printed from ABS plastic with a paper overlay. Shapes included a 325 cm^2 equilateral triangle, 324 cm^2 square, 281 cm^2 hexagon, 254 cm^2 circle, and a 486 cm^2 (18 cm×27 cm) rectangle, all shown in Fig. 17. The laser-cut patterns for the neon green fiducial markers on the robots and 3D files for objects are available at our github repository, [39].

a) Swarm mean control (hardware experiment): Unlike the PD controller (11), we cannot command a force input to the kilobots. Instead, control is given by turning on one of eight lights. The kilobots run a phototaxis routine where they search for an orientation that aligns them with the light source, and then move with an approximately constant velocity toward this light. The kilobots oscillate along this orientation because they only have one light detector.

We use the sign of (11), and choose the closest orientation to $\mathbf{D}(\mathbf{b})$ among the eight light sources. Fig. 18 shows that this limited, discretized control still enables regulating the mean position of a swarm of 100 robots.

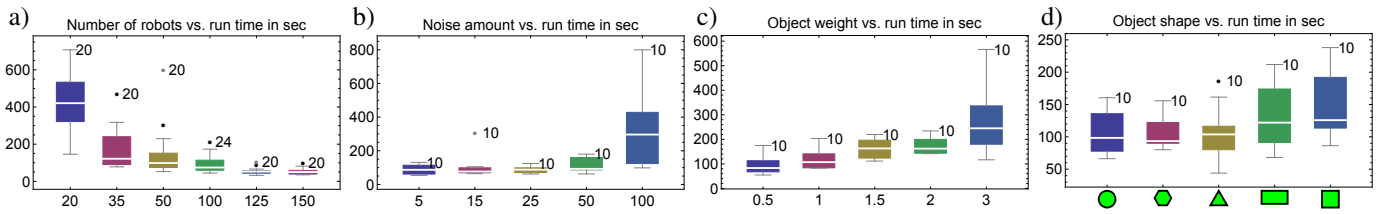

Fig. 16. Parameter sweep simulation studies for a) number of robots, b) different noise values, c) object weight, and d) object shape. Each bar is labelled with the number of trials. Completion time is in seconds.

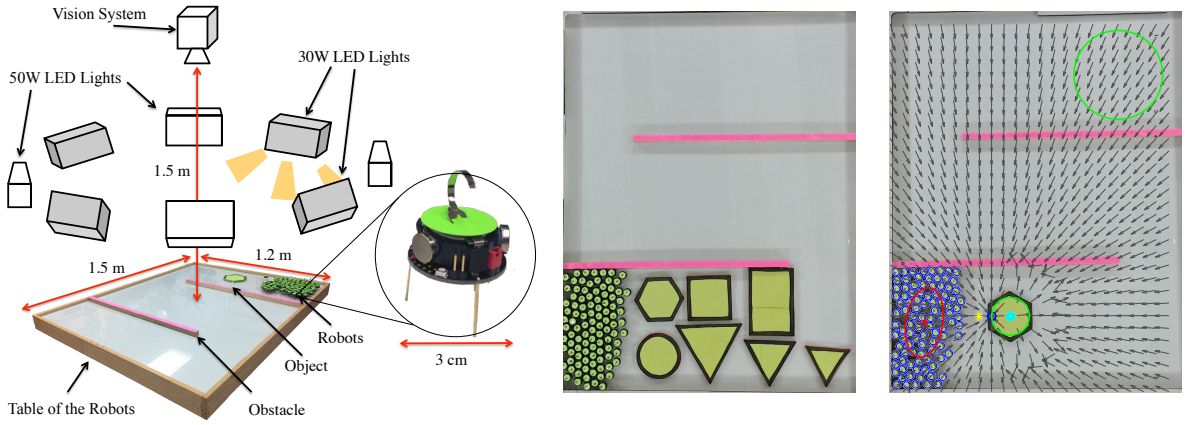

Fig. 17. Hardware platform. At right are the shapes used for hardware experiments and a visualization of the potential field.

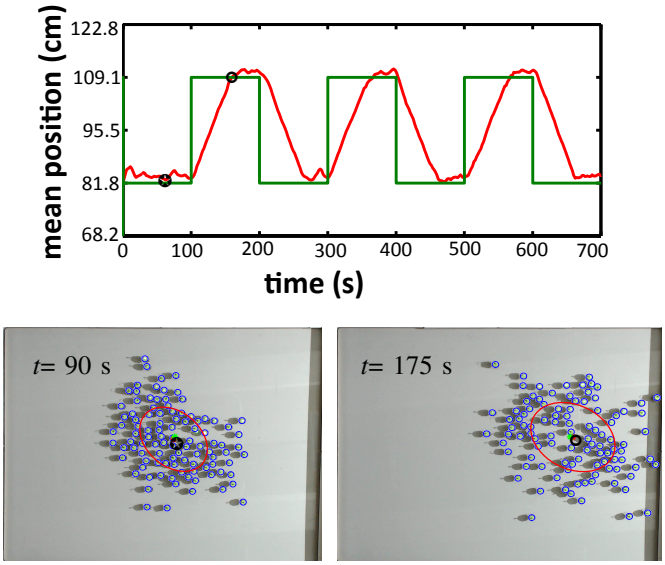

Fig. 18. Regulating mean x position of 100 kilobots using control law (11).

B. Automated object manipulation (hardware experiment)

The kilobots performed five successful runs manipulating a hexagonal object through an obstacle maze. Videos of these runs are in Extension 2. These hardware experiments represent the results of over 100 hours of trials. Each trial used 100 kilobots. Trials two through five were performed in a row with no failures in between. For each trial, fully charged kilobots were placed in the lower left-hand of the workspace, as shown in Fig. 19. The moveable object was placed in the lower center of the workspace. MATLAB code for vision processing, the policy iteration of §VI-A and the algorithm of §VI-D is available on MATLAB Central at [47]. Trials were run until the object COM entered the goal region. The trials ran for $\{1465, 3457, 3000, 2162, 2707\}$ s. This is 2558 ± 771 s (mean$\pm$std).

We also tested other object shapes. A circular object completed in 3155 s. A square object completed in 6871 s. A rectangle and three equilateral triangle objects of varying sizes failed in a total of nine runs. Manipulation failures occurred when the object was pushed into a corner, requiring torque to be unstuck. Swarm torque control is the subject of our ongoing research begun in [48].

VIII. CONCLUSION

The small size of micro and nano particles makes individual control and autonomy challenging, so currently these particles are steered by global control inputs such as magnetic fields or chemical gradients. To investigate this control challenge, this paper introduced SwarmControl.net, an open-source tool for large-scale user experiments where human users steer swarms of robots to accomplish tasks. Analysis of the game play results revealed benefits of measuring and controlling statistics of the swarm rather than full state feedback, robustness to IID noise, and small effects of varying population size of large swarms.

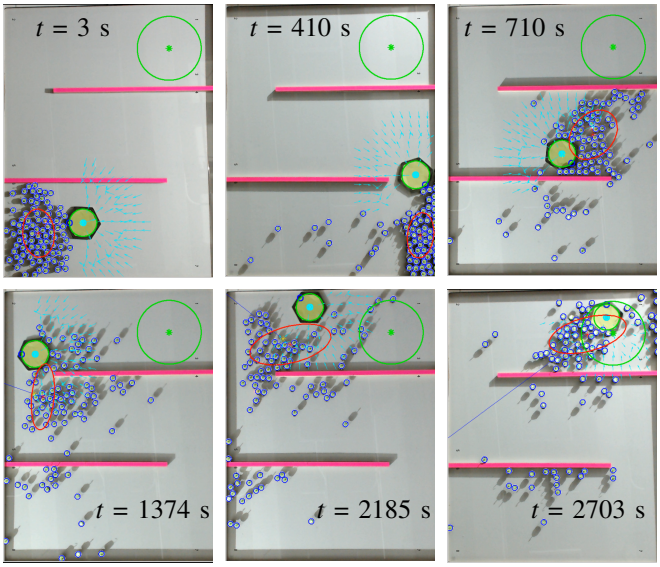

Fig. 19. Snapshots showing object manipulation experiment with 100 kilobots under automatic control. The automatic controller generates a policy to the goal, (see Extension 2).

Inspired by the three lessons from swarmcontrol.net, this paper designed controllers and controllability results using only the mean and variance of a particle swarm. We developed a hysteresis based controller to regulate the position and variance of a swarm. We designed a controller for object manipulation using policy iteration for path planning, regions for outlier rejection, and potential fields for minimizing moving the object backwards. All automatic controllers were implemented using 100 kilobots steered by the direction of a global light source. These experiments culminated in an object manipulation task in a workspace with obstacles.

Our future goal is to perform assembly using particle swarms to manipulate and attach components. This task requires controlling the position and orientation of components, manipulating them through obstacles and other components, and applying force and torque to components. This work provides foundational algorithms and techniques for steering swarms, object manipulation, and addressing obstacle fields, but there are many opportunities to extend the work.

Topics of interest include control with nonuniform flow such as fluid in an artery, gradient control fields like that of an MRI, competitive playing, multi-modal control, optimal-control, and targeted drug delivery in a vascular network.

References

[1] K. E. Peyer, L. Zhang, and B. J. Nelson, "Bio-inspired magnetic swimming microrobots for biomedical applications," *Nanoscale*, vol. 5, no. 4, pp. 1259–1272, 2013.

[2] Y. Shirai, A. J. Osgood, Y. Zhao, K. F. Kelly, and J. M. Tour, "Directional control in thermally driven single-molecule nanocars," *Nano Letters*, vol. 5, no. 11, pp. 2330–2334, Feb. 2005.

[3] 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.

[4] A. T. 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.

[5] 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.

[6] B. R. Donald, C. G. Levey, C. D. 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.

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

[8] S. Floyd, E. Diller, C. Pawashe, and M. Sitti, "Control methodologies for a heterogeneous group of untethered magnetic micro-robots," *Int. J. Robot. Res.*, vol. 30, no. 13, pp. 1553–1565, Nov. 2011.

[9] E. Diller, J. Giltinan, and M. Sitti, "Independent control of multiple magnetic microrobots in three dimensions," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 614–631, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/5/614.abstract

[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, pp. 1770–1771.

[11] S. Tottori, L. Zhang, F. Qiu, K. K. Krawczyk, A. Franco-Obregón, and B. J. Nelson, "Magnetic helical micromachines: Fabrication, controlled swimming, and cargo transport," *Advanced Materials*, vol. 24, no. 811, pp. 811–816, 2012.

[12] A. T. Becker and T. Bretl, "Approximate steering of a unicycle under bounded model perturbation using ensemble control," *IEEE Trans. Robot.*, vol. 28, no. 3, pp. 580–591, Jun. 2012.

[13] A. T. Becker, C. Onyuksel, and T. Bretl, "Feedback control of many differential-drive robots with uniform control inputs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 2256–2262.

[14] K. Takahashi, N. Ogawa, H. Oku, and K. Hashimoto, "Organized motion control of a lot of microorganisms using visual feedback," in *IEEE Int. Conf. Rob. Aut.*, May 2006, pp. 1408–1413.

[15] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[16] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[17] N. Ayanian, A. Spielberg, M. Arbesfeld, J. Strauss, and D. Rus, "Controlling a team of robots with a single input," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1755–1762.

[18] D. R. Olsen Jr. and S. B. Wood, "Fan-out: Measuring human control of multiple robots," in *SIGCHI Conference on Human Factors in Computing Systems*, Vienna, Austria, Apr. 2004, pp. 231–238.

[19] S. Bashyal and G. K. Venayagamoorthy, "Human swarm interaction for radiation source search and localization," in *Swarm Intelligence Symposium (SIS)*. IEEE, 2008, pp. 1–8.

[20] J.-P. de la Croix and M. Egerstedt, "Controllability characterizations of leader-based swarm interactions," in *AAAI Fall Symposium Series*, 2012, pp. 1–6.

[21] A. Kolling, S. Nunnally, and M. Lewis, "Towards human control of robot swarms," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. ACM, 2012, pp. 89–96.

[22] 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.

[23] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," in *Robotics and Automation, ICRA IEEE International Conference on*, May 2008, pp. 1471–1476.

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

[25] 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, pp. 945–956, Aug. 2006.

[26] A. T. Becker, R. Sandheinrich, and T. Bretl, "Automated manipulation of spherical objects in three dimensions using a gimbaled air jet," in *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, Oct. 2009, pp. 781–786.

[27] 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.

[28] 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.

[29] A. T. 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*, 2014, pp. 2825–2830.

[30] S. Shahrokhi and A. T. Becker, "Stochastic swarm control with global inputs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 421–427.

[31] C. Ertel, S. Shahrokhi, and A. T. Becker, "SwarmControl.net git repository," Aug. 2016. [Online]. Available: https://github.com/RoboticSwarmControl/SwarmControlRedux.git

[32] S. Har-Peled, "On the expected complexity of random convex hulls," *arXiv preprint arXiv:1111.5340*, 2011.

[33] A. T. 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.

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

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

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

[37] 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.

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

[39] S. Shahrokhi, M. Wan, L. Lin, and A. T. Becker, "Steering Swarm Simulation, http:goo.gl/qsVqTU," Aug. 2016.

[40] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, Sep. 2005.

[41] A. T. Becker, ""MDP robot grid-world example", https://www.mathworks.com/matlabcentral/fileexchange/49992-mdp-robot-grid-world-example," Mar. 2015.

[42] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*. John Wiley & Sons, 2008.

[43] D. L. Christensen, S. A. Suresh, K. Hahm, and M. R. Cutkosky, "Lets all pull together: Principles for sharing large loads in microrobot teams," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1089–1096, 2016.

[44] 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.

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

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

[47] S. Shahrokhi, L. Lin, M. Wan, and A. T. Becker, ""Kilobot Swarm Control using Matlab + Arduino", http://mathworks.com/matlabcentral/fileexchange/58690," Aug. 2016.

[48] S. Shahrokhi and A. T. Becker, "Object manipulation and position control using a swarm with global inputs," in *IEEE Conference on Automation Science and Engineering (CASE)*, Aug. 2016, pp. 1–6.