Quantifying Robotic Swarm Coverage

Brendon Anderson[1], Eva Loeser[2], Marissa Gee[3], Fei Ren[4], Swagata Biswas[5], Olga Turanova[6], Matt Haberland[7], and Andrea L. Bertozzi[8]

[1] UC Berkeley, Department of Mechanical Engineering, Berkeley, CA 94720
[2] UC San Diego, Department of Mathematics, San Diego, CA 92093
[3] Cornell University, Center for Applied Mathematics, Ithaca, NY 14853
[4] UC Berkeley, Industrial Engineering and Operations Research, Berkeley, CA 94720
[5] University of Edinburgh, School of Mathematics, Edinburgh, Scotland, EH9 3FD
[6] Institute for Advanced Study, School of Mathematics, Princeton, NJ 08540
[7] Cal Poly, BioResource and Agricultural Engineering, San Luis Obispo, CA 93407
mhaberla@calpoly.edu
[8] UCLA, Department of Mathematics, Los Angeles, CA 90095

Abstract. In the field of swarm robotics, the design and implementation of spatial density control laws has received much attention, with less emphasis being placed on performance evaluation. This work fills that gap by introducing an error metric that provides a quantitative measure of coverage for use with any control scheme. The proposed error metric is continuously sensitive to changes in the swarm distribution, unlike commonly used discretization methods. We analyze the theoretical and computational properties of the error metric and propose two benchmarks to which error metric values can be compared. The first uses the realizable extrema of the error metric to compute the relative error of an observed swarm distribution. We also show that the error metric extrema can be used to help choose the swarm size and effective radius of each robot required to achieve a desired level of coverage. The second benchmark compares the observed distribution of error metric values to the probability density function of the error metric when robot positions are randomly sampled from the target distribution. We demonstrate the utility of this benchmark in assessing the performance of stochastic control algorithms. We prove that the error metric obeys a central limit theorem, develop a streamlined method for performing computations, and place the standard statistical tests used here on a firm theoretical footing. We provide rigorous theoretical development, computational methodologies, numerical examples, and MATLAB code for both benchmarks.

Keywords: Swarm robotics, multi-agent systems, coverage, optimization, central limit theorem.

1 INTRODUCTION

Much of the research in swarm robotics has focused on determining control laws that elicit a desired group behavior from a swarm [6], and less attention has been placed on methods for evaluating the performance

of these controllers quantitatively. Both [6] and [9] point out the lack of developed performance metrics for assessing and comparing swarm behavior, and [6] notes that existing performance metrics are often too specific to the task being studied to be useful in comparing performance across controllers.

This extended version of the authors' paper [1] studies an error metric that evaluates one common desired swarm behavior: distributing the swarm according to a prescribed spatial density. Here, we delve deeper into the analysis of the properties of the error metric. Further, we include new examples, including a continuous target distribution to complement the original piecewise constant distribution, in order to better illustrate the utility of our methods.

In many applications of swarm robotics, the swarm must spread across a domain according to a target distribution in order to achieve its goal. Some examples are in surveillance and area coverage [8,20,23,31], achieving a heterogeneous target distribution [4, 10, 14, 16, 17, 34, 41], and aggregation and pattern formation [28–30, 36–38]. Despite the importance of assessing performance, some studies such as [28,29,34,38,41] rely only on qualitative methods such as visual comparison. Others present performance metrics that are too specific to be used outside of the specific application, such as measuring cluster size in [36], distance to a precomputed target location in [10, 31], and area coverage by tracking the path of each agent in [8]. Ergodicity has also been used as a notion of coverage error [2, 26], but it quantifies only the time-average statistics of the robot trajectories rather than the effective coverage of the swarm at an instant in time. In [30] an L^2 norm of the difference between the target and achieved swarm densities is considered, but the notion of achieved swarm density is particular to the controllers under study. These existing works do not provide methodology for carrying out computations with the notions they introduce. Moreover, they do not give guidance on how to interpret the values produced by the error metrics they define. Our work seeks to fill these gaps.

We develop and analyze an error metric that quantifies how well a swarm achieves a prescribed spatial distribution. Our method is independent of the controller used to generate the swarm distribution, and thus has the potential to be used in a diverse range of robotics applications. In [18, 25, 40], error metrics similar to the one presented here are used, but their properties are not discussed in sufficient detail for them to be widely adopted. In particular, although the error metric that we study always takes values somewhere between 0 and 2, these values are, in general, not achievable for an arbitrary desired distribution and a fixed number of robots. Therefore, one needs a better understanding of the finer properties of the error metric in order to judge whether its values (and hence the performance of the underlying controller) are "good" or not. We address this by studying two benchmarks,

1. the *extrema* of the error metric, and
2. the *probability density function* (PDF) of the error metric when robot positions are sampled from the target distribution,

which were first proposed in [25]. Using tools from nonlinear programming for (1) and rigorous probability results for (2), we put each of

these benchmarks on a firm foundation. In addition, we provide MATLAB code for performing all calculations at https://git.io/v5ytf. Thus, by using the methods developed here, one can assess the performance of a given controller for any target distribution by comparing the error metric value of robot configurations produced by the controller against benchmarks (1) and (2).

Our paper is organized as follows. Our main definition, its basic properties, and a comparison to common discretization methods is presented in Section 2. Then, Section 3 and Section 4 are devoted to studying (1) and (2), respectively. We suggest future work in Section 5 and conclude in Section 6.

2 QUANTIFYING COVERAGE

One difficulty in quantifying swarm coverage is that the target density within the domain is often prescribed as a continuous (or piecewise continuous) function, yet the swarm is actually composed of a finite number of robots at discrete locations. The approach we take here is to use the robot positions to construct a continuous function that represents coverage (e.g. [2, 18, 24, 29, 42]). It is also possible to use a combination of the two methods, as in [11].

The method we present and analyze is inspired by vortex blob numerical methods for the Euler equation and the aggregation equation (see [12] and the references therein). There are also similarities between our approach and the numerical methods known as smoothed particle hydrodynamics (SPH), which have also been applied in robotics [28, 29, 41]. One main idea behind vortex methods and SPH is to use particles to approximate a continuous function that represents the fluid. We apply this idea but with the opposite aim: namely, we represent discrete points (the robots' positions) with a continuous function. A similar strategy was alluded to in [18] and used in [25, 40] to measure the effectiveness of a certain robotic control law, but to our knowledge, our work here and in [1] is the first to develop any such method in a form sufficiently general for common use.

This section is devoted to our definition of the error metric and to its basic properties and computational considerations. We also provide a contrast between our method and another common way to measure error, namely, by discretizing the domain (e.g. [4,14]) in Subsection 2.3. Finally, in Subsection 2.4, we present the setup for the two main examples that we will use throughout the paper.

2.1 Preliminaries

We are given a bounded region[9] $\Omega \subset \mathbb{R}^d$, a desired robot distribution $\rho : \Omega \to (0, \infty)$ satisfying $\int_\Omega \rho(z)\, dz = 1$, and N robot positions

[9] We present our definitions for any number of dimensions $d \geq 1$ to demonstrate their generality. However, in the latter sections of the paper, we restrict ourselves to $d = 2$, a common setting in ground-based applications.

$x_1, \ldots, x_N \in \Omega$. To compare the discrete points $x_1, \ldots, x_N$ to the function ρ, we place a "blob" of some shape and size at each point x_i. The blob or *robot blob function* can be any function $K : \mathbb{R}^d \to \mathbb{R}$ that is non-negative on Ω and satisfies $\int_{\mathbb{R}^d} K(z)dz = 1$. In other contexts the blob function is referred to as a "kernel" [28, 39, 41]. Although more general blob shapes may be useful to represent properties of a robot's sensing and/or manipulation capabilities, it is often natural to use a K that is radially symmetric, decreases along radial directions, and enjoys certain integrability properties. We discuss these issues further in Subsection 4.1. One choice of K could be a scaled indicator function, for instance, a function of constant value within a disc of radius 1 and 0 elsewhere. This is an appropriate choice when a robot is considered to either perform its task at a point or not, and where there is no notion of the degree of its effectiveness. For the remainder of this paper, however, we usually take K to be the Gaussian

$$G(z) = \frac{1}{2\pi} \exp\left(-\frac{|z|^2}{2}\right),$$

which is useful when the robot is most effective at its task locally and to a lesser degree some distance away. A list of other common choices of K and different ways of constructing multivariate blobs from single variable functions can be found in [39, Chapter 2, Chapter 4].

To formulate our definition, we need one more parameter, a positive number δ, which we call *blob radius*. We then define K^δ as,

$$K^\delta(z) = \frac{1}{\delta^d} K\left(\frac{z}{\delta}\right). \tag{1}$$

We point out that this rescaling preserves the L^1 norm of K, so that $\int_{\mathbb{R}^d} K^\delta(z)\, dz = 1$ holds for all $\delta > 0$.

The shape of K and the blob radius δ have two physical interpretations as:

- the area in which an individual robot performs its task, or
- inherent uncertainty in the robot's position.

Either of these interpretations (or a combination of the two) can be invoked to make a meaningful choice of these parameters.

To define the *swarm blob function* ρ_N^δ, we place a blob G^δ at each robot position x_i, sum over i and renormalize, yielding,

$$\rho_N^\delta(z; x_1, ..., x_N) = \frac{\sum_{i=1}^N G^\delta(z - x_i)}{\sum_{i=1}^N \int_\Omega G^\delta(z - x_i)\, dz}. \tag{2}$$

For brevity, we usually write $\rho_N^\delta(z)$ to mean $\rho_N^\delta(z; x_1, ..., x_N)$. This swarm blob function gives a continuous representation of how the discrete robots are distributed. Note that each integral in the denominator of (2) approaches 1 if δ is small or all robots are far from the boundary, so that we have,

$$\rho_N^\delta(z) \approx \frac{1}{N} \sum_{i=1}^N G^\delta(z - x_i). \tag{3}$$

Moreover, $\rho_N^\delta(z; x_1, ..., x_N)$ is a smoothed version of $\sum_{i=1}^{N} \delta_{x_i}(z)$, which represents the physical robot positions (here, $\delta_{x_i}(z)$ denotes the Dirac mass at x_i). Indeed,

$$\lim_{\delta \to 0} \rho_N^\delta(z; x_1, ..., x_N) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}(z) \tag{4}$$

in the sense of distributions [7]. This point of view gives yet another interpretation of ρ_N^δ.
We now introduce our notion of error:

Definition 1. *The* error metric e_N^δ *is defined as:*

$$e_N^\delta(x_1, ..., x_N) = \int_\Omega \left| \rho_N^\delta(z) - \rho(z) \right| dz. \tag{5}$$

This definition, including the definitions of K^δ and ρ_N^δ, was introduced in this context by the authors in [1]. The relation (3) appeared there as well.

Remarks and Basic Properties Our error is defined as the L^1 norm of the difference between the swarm blob function ρ_N^δ and the desired robot distribution ρ. One could use another L^p norm; however, $p = 1$ is a standard choice in applications that involve particle transportation and coverage such as [18,40]. Moreover, the L^1 norm has a key property: for any two integrable functions f and g,

$$\int_\Omega |f - g| \, dz = 2 \sup_{B \subset \Omega} \left| \int_B f \, dz - \int_B g \, dz \right|.$$

The other L^p norms do not enjoy this property [15, Chapter 1]. Consequently, by measuring L^1 norm on Ω, we are also bounding the error we make on any particular subset, and, moreover, knowing the error on "many" subsets gives an estimate of the total error. This means that by using the L^1 norm we capture the idea that discretizing the domain provides a measure of error, but avoid the pitfalls of discretization methods described in Subsection 2.3.
Studies in optimal control of swarms often use the L^2 norm due to the favorable inner product structure [40]. We point out that the L^1 norm is bounded from above by the L^2 norm: indeed, according to the Cauchy-Schwarz inequality, for any function f we have,

$$\int_\Omega f \, dz \leq |\Omega| \left(\int_\Omega f^2 \, dz \right)^{1/2},$$

where $|\Omega|$ denotes the area of the bounded region Ω. Thus, if an optimal control strategy controls the L^2 norm, then it will also control the error metric we present here.
Last, we note that for any Ω, ρ, δ, N, and $(x_1, ..., x_N)$, we have $0 \leq e_N^\delta \leq 2$. This was established in Proposition 2.1 of [1]. The theoretical

minimum of e_N^δ can only be approached for a general target distribution when δ is small and N is large, or in the trivial special case when the target distribution is exactly the sum of N Gaussians of the given δ, motivating the need to develop benchmarks (1) and (2).

Variants of the Error Metric The notion of error in Definition 1 is suitable for tasks that require good instantaneous coverage. For tasks that involve tracking average coverage over some period of time (and in which the robot positions are functions of time t), an alternative "cumulative" version of the error metric is

$$\int_\Omega \left| \frac{1}{M} \sum_{j=1}^{M} \rho_N^\delta(z, t_j) - \rho(z) \right| dz \tag{6}$$

for time points $j = 1, \ldots, M$. This was defined in the authors' [1], and is similar to the metric used in [40]. It may also be likened to the approach of [2], which expresses the average over time using an integral rather than finite sum and measures the difference from the target distribution using the Kullback-Leibler divergence and Bhattacharyya distance instead of an L^1 norm. Although the cumulative error metric (6) is, in general, distinct from the instantaneous version of (5), note that the extrema and PDF of this cumulative version can be calculated in the same way as the extrema and PDF of the instantaneous error metric with MN robots. Therefore, in subsequent sections we restrict our attention to the extrema and PDF of the instantaneous formulation without loss of generality.

In addition, [40] considers a one-sided notion of error, in which a scarcity of robots is penalized but an excess is not, that is,

$$\hat{e}_N^\delta = \int_{\Omega^-} \left| \rho_N^\delta(z) - \rho(z) \right| dz,$$

where $\Omega^- := \{z | \rho_N^\delta(z) \leq \rho(z)\}$. The definition of $\hat{e}_N^\delta$, which appears in [1], is particularly useful in conjunction with the choice of K^δ as a scaled indicator function, as $\hat{e}_N^\delta$ becomes a direct measure of the deficiency in coverage of a robotic swarm. For instance, given a swarm of surveillance robots, each with observational radius δ, $\hat{e}_N^\delta$ is the percentage of the domain not observed by the swarm.[10]

Remarkably, $\hat{e}_N^\delta$ and e_N^δ are related by $e_N^\delta = 2\hat{e}_N^\delta$. This was established by the authors in Proposition 2.2 of [1]. This relationship implies that e_N^δ enjoys the interpretation of being a measure of deficiency, and also allows the techniques introduced here to be directly applied to $\hat{e}_N^\delta$.

2.2 Calculating e_N^δ

In practice, the integral in (5) can rarely be carried out analytically, primarily because the integral needs to be separated into regions for which

[10] The notion of "coverage" in [8] might be interpreted as $\hat{e}_N^\delta$ with δ as the width of the robot. There, only the time to complete coverage (t such that $\hat{e}_N^\delta(z; x_1(t), ..., x_n(t)) = 0$) was considered.

the quantity $\rho_N^\delta(z) - \rho(z)$ is positive and regions for which it is negative, the boundaries between which are usually difficult to express in closed form. Hence we must rely on numerical integration, or quadrature. Here we study the magnitude E_m of the difference between the true value of the error metric e_N^δ and an approximation $\tilde{e}_N^\delta$ as the number of grid points per axis m increases for three elementary quadrature rules: the rectangle rule, the trapezoidal rule, and Simpson's rule. As quadrature convergence rate proofs typically depend on smoothness of the integrand [13], we did not expect all of these rules to achieve their nominal rates of convergence. Indeed, in Figure 1 we see that the trapezoidal rule converges as $O(m^{-1.58})$ instead of $O(m^{-2})$ and Simpson's rule converges as $O(m^{-1})$ instead of $O(m^{-4})$. As our applications do not require extremely accurate estimates of the error metric, other error metric values reported throughout the paper have been calculated using the rectangle rule with a moderate number of grid points. For applications that demand increased accuracy, we recommend an adaptive scheme such as that employed by MATLAB's `integral2` function.

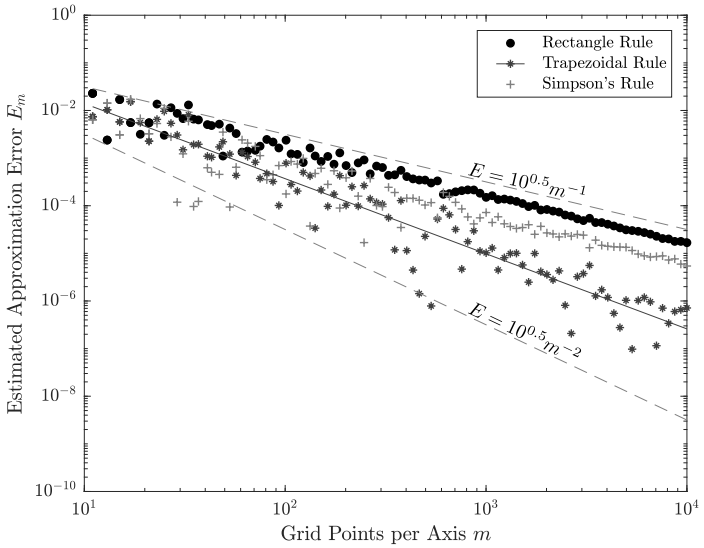

Fig. 1. Estimated error $E_m = \left|e_N^\delta - \tilde{e}_N^\delta\right|$ as the number of grid points per axis m increases. As the exact value of e_N^δ is unknown, we use the estimate provided by the MATLAB function `integral2` using the `'iterated'` method and a relative tolerance of 10^{-8}. The solid line is a best fit function of the form $10^a m^b$ for the trapezoidal rule, where $a = 0.2782$ and $b = 1.577$ were found using the MATLAB Curve Fit app. The particular configuration of robots is the optimal arrangement found (see Section 3.2) for the ring distribution from Definition 2 with $N = 200$ robots and $\delta = 2$in.

2.3 The Pitfalls of Discretization

Before concluding this section, we analyze a measure of error that involves discretizing the domain and show that the values produced by this method are strongly dependent on a choice of discretization. In particular, this error approaches its theoretical minimum when the discretization is too coarse and its theoretical maximum when the discretization is too fine, regardless of robot positions.

Discretizing the domain means dividing Ω into M disjoint regions $\Omega_i \subset \Omega$ such that $\bigcup_{i=1}^{M} \Omega_i = \Omega$. Within each region, the desired proportion of robots is the integral of the target density function within the region $\int_{\Omega_i} \rho(z)dz$. Using N_i to denote the observed number of robots in Ω_i, we can define an error metric as

$$\mu = \sum_{i=1}^{M} \left| \int_{\Omega_i} \rho(z)\, dz - \frac{N_i}{N} \right|. \tag{7}$$

This exact definition appeared in the authors' [1], and is based on discrete error metrics used in practice, for instance in [4, 14]. It is easy to check that $0 \leq \mu \leq 2$ always holds. One advantage of this approach is that μ is very easy to compute, but there are two major drawbacks.

First, the choice for domain discretization is not unique, and this choice can dramatically affect the value of μ. Indeed, if $M = 1$ then $\mu = 0$. This follows directly from the definition of μ and appears in [1] as Proposition 2.3. On the other hand, as the discretization becomes finer, the error approaches its maximum value 2, regardless of the robot positions. More precisely:

Proposition 1. *Suppose the robot positions are distinct*[11] *and the regions Ω_i are sufficiently small such that, for each i, Ω_i contains at most one robot and $\int_{\Omega_i} \rho(z)\, dz \leq 1/N$ holds. Then $\mu \to 2$ as $|\Omega_i| \to 0$.*

This proposition and its proof appear in [1] as Proposition 2.4.

Note that the shape of each region is also a choice that will affect the calculated value of μ. Although our approach also requires the choice of some size and shape (namely, δ and K), these parameters have much more immediate physical interpretations, making appropriate choices easier to make.

The second, and perhaps more significant, drawback, is that by discretizing the domain we also discretize the range of values that the error metric can assume. This means that we have simultaneously desensitized the error metric to changes in robot distribution within each region. That is, so long as the number of robots N_i within each region Ω_i does not change, the distribution of robots within any and all Ω_i may be changed arbitrarily without affecting the value of μ. On the other hand, the error metric e_N^δ is continuously sensitive to differences in distribution.

[11] This is reasonable in practice as two physical robots cannot occupy the same point in space. In addition, the proof can be modified to produce the same result even if the robot positions coincide.

2.4 Setup for Main Examples

We will perform computations with the following two desired distributions. We have purposefully chosen one that is piecewise continuous, ρ_{ring}, and one that is smooth, ρ_{ripple}.

Definition 2. *The* ring distribution ρ_{ring} *is defined on the Cartesian plane with coordinates* $z = (z_1, z_2)$ *as follows. Let inner radius* $r_1 = 11.4in$*, outer radius* $r_2 = 20.6in$*, width* $w = 48in$*, height* $h = 70in$*, domain* $\Omega = \{z : z_1 \in [0, w], z_2 \in [0, h]\}$*, and region* $\Gamma = \{z : r_1^2 < (z_1 - \frac{w}{2})^2 + (z_2 - \frac{h}{2})^2 < r_2^2\}$*. Define the non-normalized* $\rho'_{\text{ring}}(z)$ *to be* 36 *if* $z \in \Omega \cap \Gamma$ *and* 1 *if* $z \in \Omega \setminus \Gamma$*. Then let* $C = \int_\Omega \rho'_{\text{ring}}(z)\, dz$ *and* $\rho_{\text{ring}}(z) = C^{-1}\rho'_{\text{ring}}(z)$ *for* $z \in \Omega$*.*

Definition 3. *The* ripple distribution ρ_{ripple} *is defined on the Cartesian plane with coordinates* $z = (z_1, z_2)$ *as follows. Let width* $w = 48in$*, height* $h = 70in$*, domain* $\Omega = \{z : z_1 \in [0, w], z_2 \in [0, h]\}$*, non-normalized* $\rho'_{\text{ripple}}(z) = 2 + \sin[3\pi(z_1^2 + z_2^2)^{\frac{1}{2}}] + 2\frac{z_1^2}{w^2} + \frac{z_2^3}{h^3}$*, and normalization factor* $C = \int_\Omega \rho'_{\text{ripple}}(z)\, dz$*. Then* $\rho_{\text{ripple}}(z) = C^{-1}\rho'_{\text{ripple}}(z)$ *for* $z \in \Omega$*.*

3 ERROR METRIC EXTREMA

In the rest of the paper, we provide tools for determining whether or not the values of e_N^δ produced by a controller in a given situation are "good". As mentioned in Section 2.1, it is not possible to achieve $e_N^\delta = 0$ for every combination of target distribution ρ, number of robots N, and blob radius δ . Therefore, we would like to compare the achieved value of e_N^δ against its *realizable* extrema given ρ, N, and δ. Unfortunately, e_N^δ is a nonlinear and moreover non-convex function of the robot positions $(x_1, ..., x_N)$, and thus its extrema may elude analytical expression. Thus, we approach this problem with nonlinear programming.

3.1 Extrema Bounds via Nonlinear Programming

Let $x = (x_1, ..., x_N)$ represent a vector of N robot coordinates. The optimization problem, first introduced in the authors' [1], is

$$\begin{aligned} &\text{minimize } e_N^\delta(x_1, ..., x_N), \\ &\text{subject to } x_i \in \Omega \text{ for } i \in \{1, 2, \ldots, N\}. \end{aligned} \tag{8}$$

Note that the same problem structure can be used to find the maximum of the error metric by minimizing $-e_N^\delta$. Given ρ, N, and δ, we approach these problems using a standard nonlinear programming solver, MATLAB's `fmincon`.

A limitation of all general nonlinear programming algorithms is that successful termination produces only a local minimum, which is not guaranteed to be the global minimum. Since there is no obvious formulation of this problem for which a global solution is guaranteed, we use the local minimum as an upper bound for the global minimum of the error metric. Heuristics such as multi-start (running the optimization many times

from several initial guesses and taking the minimum of the local minima) can be used to make this bound tighter. We use e^- to denote the best local minimum found and e^+ to denote the best local maximum found. The bounds e^- and e^+ serve as benchmarks against which we can compare a value of the error metric achieved by a given control law. This is reasonable, because if a configuration of robots with a lower value of the error metric exists but eludes numerical optimization, it is probably not a fair standard against which to compare the performance of a general controller.
Examples of extremal swarm blob functions found using this approach are shown in Figures 2, 3, and 4.

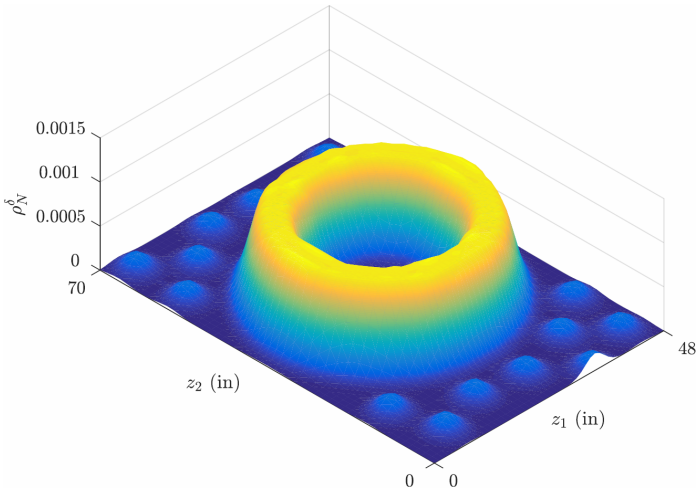

Fig. 2. Swarm blob function $\rho_{N=200}^{\delta=2\text{in}}$ corresponding with the robot distribution that yields a locally minimal value of the error metric for the ring distribution, 0.28205. This figure appears in [1] as Figure 1.

Relative Error We introduce the notion of *relative error*, a quantitative way of assessing the performance of a robot distribution controller. This notion first appeared in the authors' [1].

Definition 4. *Let* e_{observed} *denote the error value of a robot configuration. The* relative error *is defined as,*

$$e_{\text{rel}} = \frac{e_{\text{observed}} - e^-}{e^+ - e^-}. \tag{9}$$

In order to apply this definition, we need to specify how to find e_{observed}. If the robot positions $x_1, ..., x_N$ produced by a given controller are constant, then e_{observed} can simply be taken as $e_N^\delta(x_1, ..., x_N)$. In general,

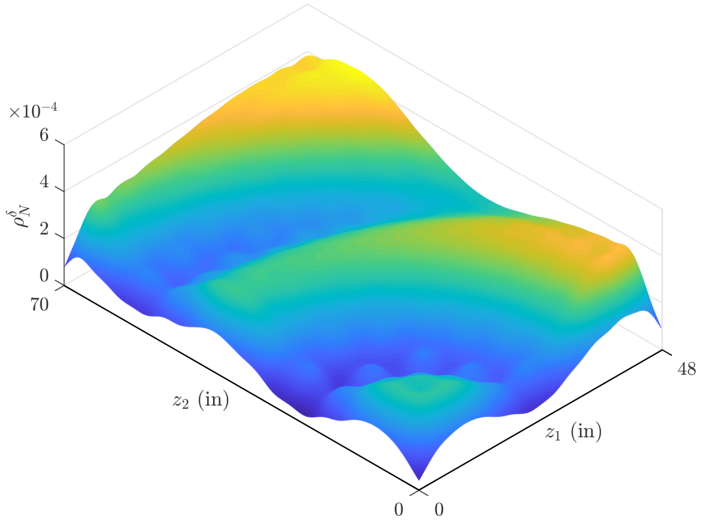

Fig. 3. Swarm blob function $\rho_{N=178}^{\delta=2.5\text{in}}$ corresponding with the robot distribution that yields a locally minimal value of the error metric for the ripple distribution, 0.06376.

however, the positions $x_1, ..., x_N$ may change over time, and it is natural to define e_{observed} based on the steady state value of e_N^δ. We suggest defining the *steady state settling time* t_s to be the time at which the error has settled to within 2% of its asymptotic value. Then, we propose taking e_{observed} to be the third-quartile value observed for $t > t_s$, which we denote e_{Q3}.

We suggest that if e_{rel} is less than 10%, the performance of the controller is quite close to the best possible, and if this ratio is 30% or higher, the performance of the controller is rather poor.

We summarize the procedure described in this section:

- Step 1: Given ρ, N and δ, compute e^- and e^+.
- Step 2: Compute e_{observed} for the desired swarm controller. That is,
  - Step 2.a: Calculate $e_N^\delta(t)$ from robot trajectories $x_1(t), ..., x_N(t)$ produced by the controller.
  - Step 2.b: Find the steady state settling time t_s.
  - Step 2.c: Measure the third quartile value of the error metric e_{Q_3} for $t > t_s$. This is e_{observed}.
- Step 3: Evaluate e_{rel} according to Definition 4.

We emphasize that e^- and e^+ are independent of the particular controller; therefore, Step 1 has to be completed only once when comparing different controllers or different outcomes of running one controller.

Example We apply this method to assess the performance of the controller in [25], which guides a swarm of $N = 200$ robots with $\delta = 2$in (the

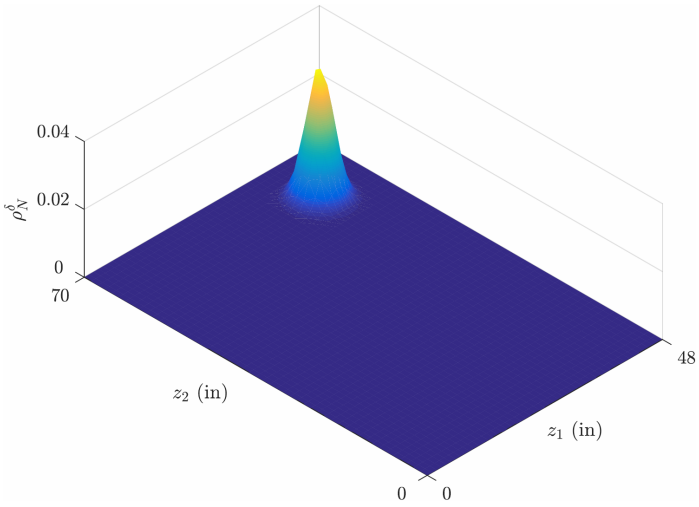

Fig. 4. Swarm blob function $\rho_{N=200}^{\delta=2\text{in}}$ corresponding with the robot distribution that yields a locally maximal value of the error metric for the ring distribution, 1.9867. This occurs when all robots coincide outside the ring. This figure appears in [1] as Figure 2.

physical radius of the robots) to achieve the ring distribution from Definition 2. These calculations were originally performed in the authors' [1].
Step 1: Compute e^- and e^+. To determine an upper bound on the global minimum of the error metric, we computed 50 local minima of the error metric starting with random initial guesses, then took the lowest of these to be $e^- = 0.28205$. An equivalent procedure bounds the global maximum as $e^+ = 1.9867$, produced when all robot positions coincide near a corner of the domain. The corresponding swarm blob functions are depicted in Figures 2 and 4. Note that the minimum of the error metric is significantly higher than zero for this finite number of robots of nonzero radius, emphasizing the importance of performing this benchmark calculation rather than using zero as a reference.
Step 2: Find e_{observed}. Under the stochastic control law of [25], the behavior of the error metric over time appears to be a noisy decaying exponential. Therefore, we fit to the data shown in Figure 7 of [25] a function of the form $f(t) = \alpha + \beta \exp(-\frac{t}{\tau})$ by finding error asymptote α, error range β, and time constant τ that minimize the sum of squares of residuals between $f(t)$ and the data. By convention, the steady state settling time is taken to be $t_s = 4\tau$, which can be interpreted as the time at which the error has settled to within 2% of its asymptotic value [3]. The third quartile value of the error metric for $t > t_s$ is $e_{Q3} = 0.5157$.
Step 3: Find e_{rel}. Using these values for e_{Q3}, e^-, and e^+, we calculate e_{rel} according to Equation 9 as 13.71%.

Remark 1. Although the sentiment of the e_{rel} benchmark is found in [25], we have formalized the calculation and made three important improve-

ments to make it suitable for general use. First, we have placed the calculation of e^- and e^+ into a framework appropriate for nonlinear programming, so that the calculation is objective and repeatable. On the other hand, in [25], values analogous to e^- and e^+ were found by "manual placement" of robots. Second, we suggest a general way of evaluating error in time-dependent controllers. In particular, although [25] refers to steady state, a definition is not suggested there. Adopting the 2% settling time convention allows for an unambiguous calculation of e_{Q_3} and other steady state error metric statistics. It also provides a metric for assessing the speed with which the control law effects the desired distribution. Finally, we suggest using the third quartile value e_{Q_3} in the calculation, in contrast to the minimum observed value of the error metric used in [25]. Since e_{Q_3} better represents the distribution of error metric values achieved by a controller, our notion of relative error is more representative of the controller's overall performance.
These changes account for the difference between our calculated value of $e_{\text{rel}} = 13.71\%$ and the report in [25] that the error is "7.2% of the range between the minimum error value... and maximum error value". Our substantially higher value of e_{rel} indicates that the performance of this controller is not very close the best possible. We emphasize this to motivate the need for our second benchmark in Section 4, which may be more appropriate for a stochastic controller like the one presented in [25].

3.2 Error Metric for Optimal Swarm Design

So far we have taken N and δ to be fixed; we have assumed that the robotic agents and size of the swarm have already been chosen. Now, we briefly consider the use of the error metric as an objective function for the *design* of a swarm. Adding $\delta > 0$ as a decision variable to (8) yields the minimization problem,

$$\begin{aligned} &\text{minimize } e_N^{\delta}(x_1, ..., x_N), \\ &\text{subject to } x_i \in \Omega \text{ for } i \in \{1, 2, \ldots, N\},\ \delta > 0. \end{aligned} \tag{10}$$

Solving (10) for several fixed values of N provides insight into the number of robots and what effective working radius are needed to achieve a given level of coverage for a particular target distribution. Visualizations of such calculations are provided in Figure 5 and 6, and Figure 7 shows the optimal value of δ as a function of swarm size.
Note that "breakthroughs", or relatively rapid decreases in the error metric, can occur once a critical number of robots is available; these correspond with a qualitative change in the distribution of robots. For example, at $N = 22$ in Figure 5 the robots are arranged in a single ring; beginning with $N = 25$ we see the robots start to be arranged in two separate concentric rings of different radii and the error metric begins to drop sharply. On a related note, there are also "lulls" in which increasing the number of robots has little effect on the minimum value of the error metric, such as between $N = 44$ and $N = 79$. We leave for future work the task of finding a theoretical explanation for these breakthroughs and lulls. In the meantime, computations like these can help a swarm designer

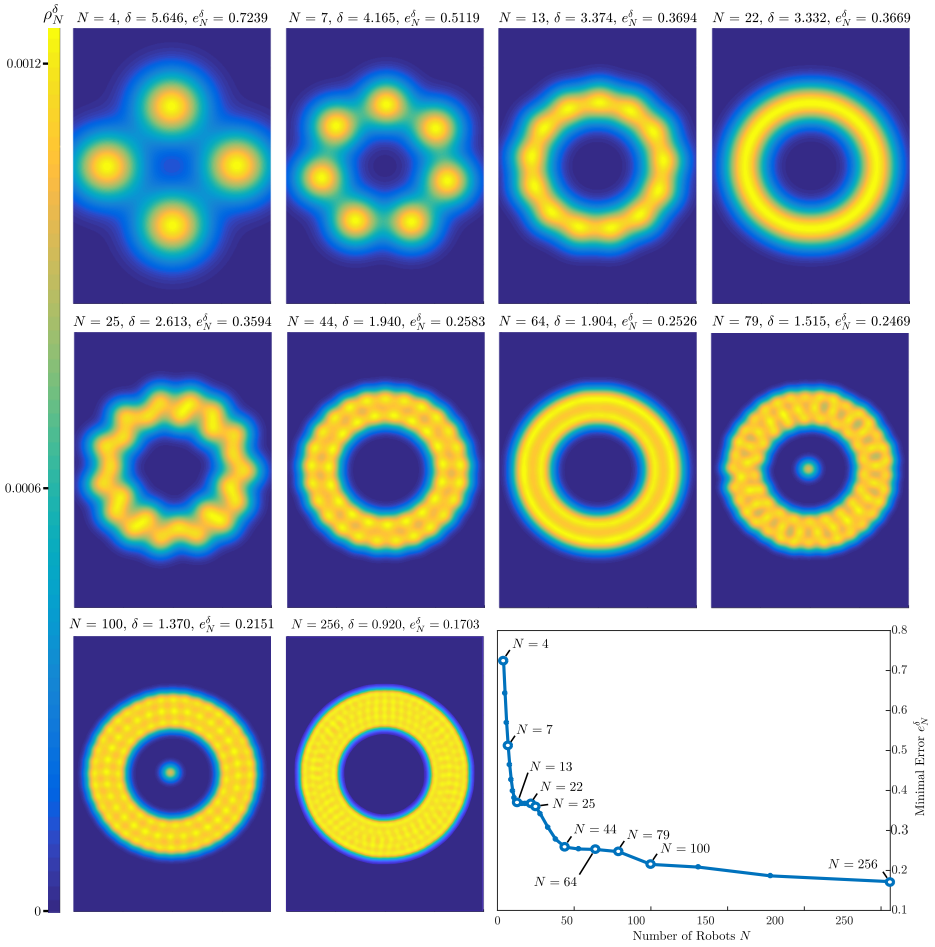

Fig. 5. Swarm blob functions ρ_N^δ corresponding with the robot distributions and values of δ that yield the minimum value of the error metric for the ring distribution target. These plots appear in Figure 3 of [1]. Inset graph shows the relationship between N and the minimum value of the error metric observed from repeated numerical optimization. For $N < 256$, the initial guess provided to the optimizer had robots uniformly randomly distributed within the domain. For $N = 256$ in [1], such a guess resulted in local minima with error values *greater* than those for smaller swarms, inconsistent with the intuitive notion that a larger swarm should be able to more accurately achieve the target distribution. To remedy this here, the initial guesses for $N = 256$ and higher (not shown) were taken with all robots between the inner and outer radius of the ring (i.e. within the region Γ from Definition 2). With these the plot decreases monotonically for all values of N tested (including 300, 350, 400, 450, and 500), but progress appears slow beyond $N = 256$; doubling the number of robots decreases the error metric by only 10%.

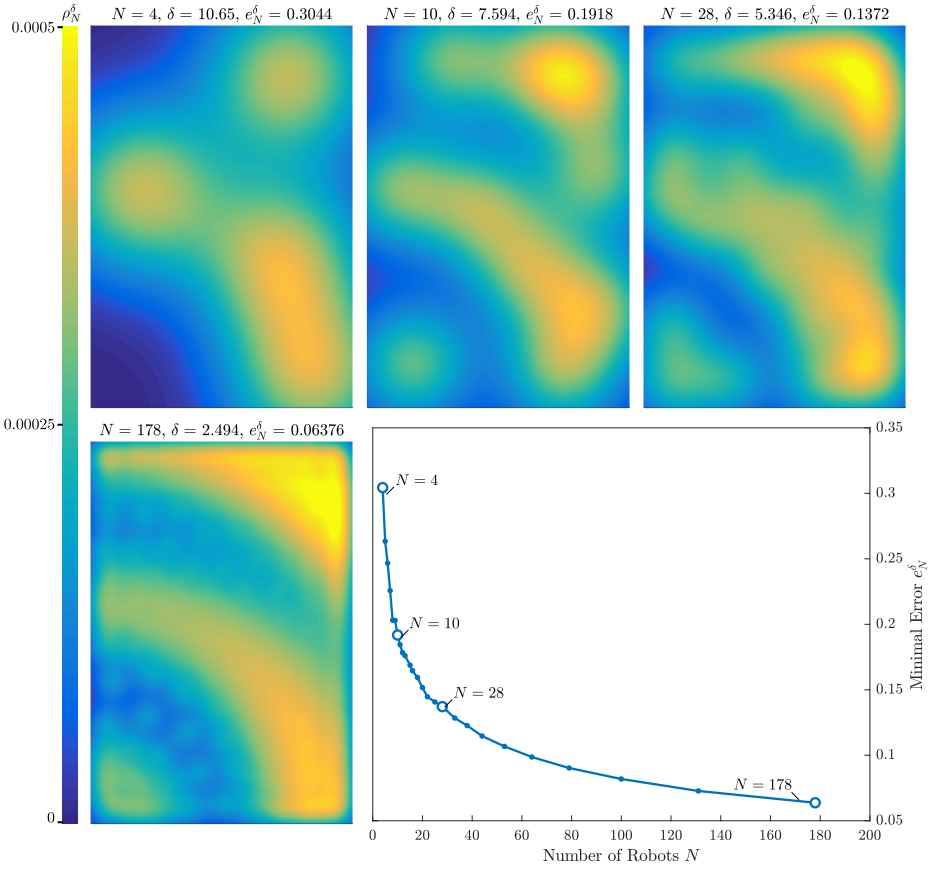

Fig. 6. Swarm blob functions ρ_N^δ corresponding with the robot distributions and values of δ that yield the minimum value of the error metric for the ripple distribution target. Inset graph shows the relationship between N and the minimum value of the error metric observed from repeated numerical optimization.

determine the best number of robots N and effective radius of each δ to achieve the required coverage.

4 ERROR METRIC PROBABILITY DENSITY FUNCTION

In the previous section, we described how to find bounds on the minimum and maximum values for error and use these as benchmarks against which to measure the performance of a given control law. However, stochastic control laws may tend to produce robot positions with observed error e_{observed} well above the minimum e^-. For these controllers, e_{rel} may be too stringent of a measure for assessing whether the controller's performance is "good" or "bad", and so we propose an alternative.

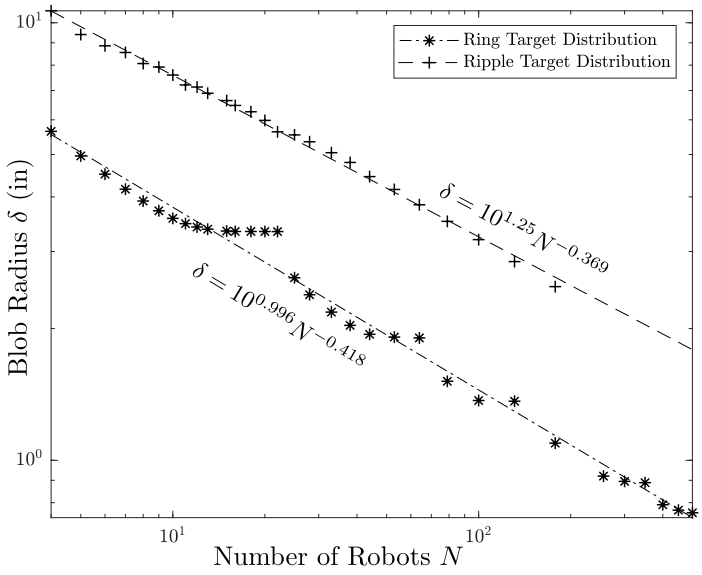

Fig. 7. Optimal blob radius δ to minimize error e_N^δ along with best fit lines (log-log scale). Note that for both target distributions the optimal blob radius seems to scale with N^p where $p \approx -0.4$. This relationship may be useful for approximating the effective radius of robot sensing/manipulation capabilities in order to maximize effectiveness for a given swarm size.

According to the setup of our problem, the goal of any control law is for the robots to achieve the desired distribution ρ. Thus, it is natural to compare the outcome of a given control law to simply picking the robot positions at random from the target distribution ρ. We take the robots' positions $X_1, \ldots, X_N$, to be independent, identically distributed bivariate random vectors in $\Omega \subset \mathbb{R}^2$ with probability density function ρ. We place a blob of shape K at each of the X_i (previously we took K to be the Gaussian G), so that the swarm blob function is,

$$\rho_N^\delta(z) = \frac{1}{N} \sum_{i=1}^{N} K^\delta \left(z - X_i\right), \tag{11}$$

where K^δ is defined by (1). Equation (11) appears in the authors' [1]. We point out that the right-hand sides of (11) and (3) agree upon taking K to be the Gaussian G in (11) and the robot locations x_i to be the randomly selected X_i in (3). Moreover, we have the relationship,

$$\rho_N^\delta = K^\delta * \left(\frac{1}{N} \sum_{i=1}^{N} \delta_{X_i} \right), \tag{12}$$

where $*$ denotes convolution. This equality is the reason (4) holds.

In this context, the error e_N^δ is a random variable, the value of which depends on the particular realization of the robot positions $X_1, \ldots, X_N$, and thus has a well-defined probability density function (PDF) and cumulative distribution function (CDF). We denote the PDF and CDF by $f_{e_N^\delta}$ and $F_{e_N^\delta}$, respectively. The performance of a stochastic robot distribution controller can be quantitatively assessed by calculating the error values it produces in steady state and comparing their distribution to $f_{e_N^\delta}$. We introduce this benchmark in Subsection 4.4.

4.1 Kernel Density Estimation

The approach we take in this section is closely linked to the statistical theory of kernel density estimation (KDE). For an overview, see, for example, the book [32]. Broadly, the aim of this area of study is to find the underlying density ρ from the values of identically distributed random variables sampled from this density. The expression (11) is the so-called *kernel density estimator* of ρ, and ρ_N^δ is considered as an approximation to ρ. In a sense, this is the reverse point of view from the one we take, since we are given ρ and seek to learn about the distribution of the X_is. Nevertheless, we are able to apply ideas from this well-developed field of study to our context.

There are many notions of error that are used in the context of KDE. We refer the reader to [39, Chapter 2] for a summary. Our notion e_N^δ corresponds to *integrated absolute error* (IAE). The use of IAE instead of other notions, particularly ones involving the L^2 norm, was advocated for in [15].

We conclude this subsection by applying two results from KDE theory to ideas we've presented so far. Then, in Subsection 4.2 we present rigorous results that show that the error metric has an approximately normal distribution when the X_i are sampled from ρ. As a corollary we obtain that the limit of this error is zero as N approaches infinity and δ approaches 0. Subsections 4.3 and 4.4 include a numerical demonstration of these results.

The theoretical results presented in the next subsection not only support our numerical findings, but they also allow for faster computation. Indeed, if one did not already know that the error when robots are sampled randomly from ρ has a normal distribution for large N, tremendous computation may be needed to get an accurate estimate of this probability density function. On the other hand, since the results we present prove that the error metric has a normal distribution for large N, we need only fit a Gaussian function to the results of relatively little computation. In Subsection 4.4 we present an example calculation demonstrating the utility of this approach.

Optimal $\delta(N)$ In Subsection 3.2, we included δ as a parameter in the optimization problem (8) and numerically obtained an estimate for the optimal δ given a swarm size N. This problem has also been studied in the context of KDE. In [33] it was found that the optimal choice is,

$$\delta^*(N) = CN^{-1/6},$$

where C is a constant that does not depend on N. This is equation (2.3) of [33] in the 2-dimensional case. Here, "optimal" means that this choice of δ minimizes the expectation of the L^1 error between ρ and ρ_N^δ. This result holds under the hypotheses that K satisfies (13) and ρ has bounded second derivatives.
We remark that the optimal $\delta(N)$ minimizing (10) that was found in Subsection 3.2 and shown in Figure 7 scales with N^p where $p \approx -0.4 \neq -1/6$. Despite the difference in exponent, these computations do not contradict the results of [33] as the latter considers the radius δ that minimizes the *expected* error value, and hence takes into account robot positions that are not optimal. On the other hand, the values of $\delta(N)$ computed in Subsection 3.2 are optimal only in concert with the optimal robot positions. The contrast between the two situations suggests that for the design of a swarm as described in Subsection 3.2, the sensing/manipulation radius required to maximize coverage may depend on whether the controller is stochastic or deterministic.

Choice of K In the context of KDE, there is a notion of how efficient a given kernel K is. This notion is formulated in terms of L^2 error between ρ and ρ_N^δ, and therefore the precise details do not directly apply to our situation. However, we point out that it is known [39, Section 2.7] that, so long as the kernel K is non-negative, has integral 1, and satisfies

$$\int zK(z)\,dz = 0, \quad \int z^2K(z)\,dz < \infty, \tag{13}$$

then the particular choice of K has only a marginal effect on efficiency. We leave to future work the extension of this concept to our context.

4.2 Theoretical Central Limit Theorem

It turns out that, under appropriate hypotheses, the L^1 error between ρ and ρ_N^δ has a normal distribution with mean and variance that approach zero as N approaches infinity. In other words, a central limit theorem holds for the error. The first such result was obtained in [15] in the one-dimensional case. Previously, similar results, such as those in [5], were available only for L^2 notions of error, which are easier to analyze.
For any result of central limit theorem type to hold, δ and N have to be compatible. Thus, for the remainder of this subsection δ will depend on N, and we display this as $\delta(N)$. We have,

Theorem 1. *Suppose ρ is twice continuously differentiable, and K is zero outside of some bounded region and radially symmetric. Then, for $\delta(N)$ satisfying*

$$\delta(N) = O(N^{-1/6}) \text{ and } \lim_{N\to\infty} \delta(N)N^{1/4} = \infty, \tag{14}$$

we have

$$e_N^{\delta(N)} \approx \mathcal{N}\left(\frac{e(N)}{N^{1/2}}, \frac{\sigma(N)^2\delta(N)^2}{N}\right),$$

where $\sigma^2(N)$ and $e(N)$ are deterministic quantities that are bounded uniformly in N.[12]

From Theorem 1 it is easy to deduce:

Corollary 1. *Under the hypotheses of Theorem 1, the error $e_N^{\delta(N)}$ converges in distribution to zero as $N \to \infty$.*

Corollary 1, Theorem 1, and the proof of Theorem 1 (which follows from [22, Theorem, page 1935]) appear in the authors' [1].

Remark 2. There are a few ways in which practical situations may not align perfectly with the assumptions of Theorem 1. However, we posit that in all of these cases, the differences are numerically insignificant. Moreover, we believe that a stronger version of Theorem 1 may hold, one which applies in our situation. We now briefly summarize these three discrepancies and indicate how to resolve them.
First, we defined our density ρ_N^δ by (2), but in this section we use a version with denominator N. However, as explained above, the two expressions approach each other for small δ, and this is the situation we are interested in here. We believe that the result of Theorem 1 should hold with (2) instead of (11).
Second, in one of the two examples we consider here, the desired density ρ is only piecewise continuous but not twice differentiable. We point out that an arbitrary density ρ may be approximated to arbitrary precision by a smoothed out version, for example by convolution with a mollifier (a standard reference is [7, Section 4.4]). Moreover, the main results of [19] give a version of Theorem 1 that require that ρ is only Lebesgue measurable (piecewise continuous functions satisfy this hypothesis). However, the results of [19] only apply in 1 dimension. Generalizing the results of [19] to several spatial dimensions is an interesting open problem in KDE, but is beyond the scope of our work here.
Third, in our computations we use the kernel G, which is not compactly supported, for the sake of simplicity. Similarly, this kernel can be approximated, with arbitrary accuracy, by a compactly supported version. Making these changes to the kernel or target density would not affect the conclusions of numerical results.

4.3 Numerical Approximation of the Error Metric PDF

In this subsection we describe how to numerically find $f_{e_N^\delta}$ and $F_{e_N^\delta}$. According to Theorem 1, for sufficiently large N, one could simply use random sampling to estimate the mean and standard deviation, then take these as the parameters of the normal PDF and CDF. However, for moderate N, we choose to begin by estimating the entire CDF and confirming that it is approximately normal. To this end, we apply:

[12] Here $\mathcal{N}(\mu, \sigma^2)$ denotes the normal random variable of mean μ and variance σ^2, and we use the notation $\approx$ to mean that the difference of the quantity on the left-hand side and on the right-hand side converges to zero in the sense of distributions as $N \to \infty$.

Proposition 2. *We have,*

$$F_{e_N^\delta}(z) = \int_{\Omega^N} \mathbf{1}_{\{x | e_N^\delta(x) \leq z\}} \prod_{i=1}^{N} \rho(x_i) dx. \tag{15}$$

Here $\mathbf{1}$ denotes the indicator function. Proposition 2 and its proof appear in the authors' [1].
Notice that, since each of the x_i is itself a 2-dimensional vector as the X_i are random points in the plane, the integral defining the cumulative distribution function of the error metric is of dimension $2N$. Finding analytical representations for the CDF quickly becomes infeasible for large swarms. Therefore, we approximate (15) using Monte Carlo integration, which is well-suited for high-dimensional integration [35][13]. Next, we fit a Gauss error function ($\mathrm{erf}(\cdot)$, the integral of a Gaussian G) to the data. If the fitted curve matches the data well, we differentiate to obtain the PDF. We remark that we express the integral in (15) in terms of an indicator function in order to express the quantity of interest in a way that is easily approximated with Monte Carlo integration.

Computation We apply Proposition 2 to numerically find the PDF $f_{e_N^\delta}$ and the CDF $F_{e_N^\delta}$ for $\rho = \rho_{\mathrm{ring}}$ (see Definition 2), $N = 200$, and $\delta = 2\mathrm{in}$.
We approximate $F_{e_N^\delta}$ using $M = 1000$ Monte Carlo evaluation points; this is shown by a solid gray line in Figure 8. The numerical approximation appears to closely match a Gauss error function as theory predicts. Therefore an analytical $\mathrm{erf}(\cdot)$ curve, represented by the dashed line, is fit to the data using MATLAB's least squares curve fitting routine `lsqcurvefit`. To obtain $f_{e_N^\delta}$, the analytical curve fit for $F_{e_N^\delta}$ is differentiated, and the result is also shown in Figure 8.
In addition, we calculate the mean and standard deviation of e_N^δ to be, respectively, 0.4933 and 0.02484.

4.4 Benchmark for Stochastic Controllers

In this subsection we describe how to use the tools developed so far to assess the performance of a given stochastic controller for a given desired distribution ρ, number of robots N, and blob radius δ. We then demonstrate the method via an example.
We propose using two standard statistical tests, the two-sample t- and F- tests [27], to assess whether the performance of the control law is comparable to sampling robot positions from the target distribution. More precisely, these tests provide criteria for the rejection of the null hypothesis that the mean and variance of the steady state values of the

[13] Quasi-Monte Carlo techniques, which use a low-discrepancy sequence rather than truly random evaluation points, promise somewhat faster convergence but require considerably greater effort to implement. The difficulty is in generating a low-discrepancy sequence from the desired distribution, which is possible using the Hlawka-Mück method, but computationally expensive [21].

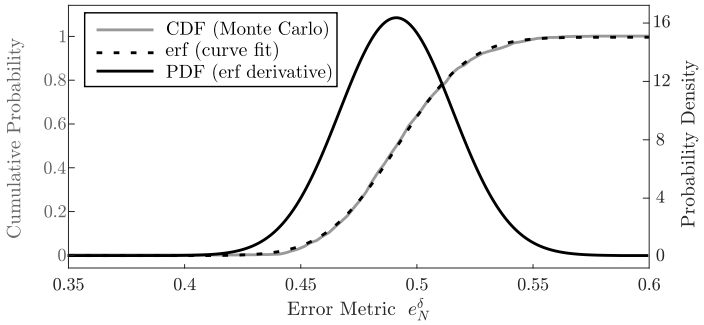

Fig. 8. The CDF of the error metric when robot positions are sampled from ρ is approximated by Monte Carlo integration, an erf$(\cdot)$ curve fit matches closely, and the PDF is taken as the derivative of the fitted erf$(\cdot)$. This figure appears as Figure 4 in the authors' [1].

error metric produced by a given control law are indistinguishable from mean and variance of $f_{e_N^\delta}$. This is summarized in the following procedure, which assumes that $f_{e_N^\delta}$ is approximately normal. This assumption is reasonable due to Theorem 1.

- Step 1: Numerically find the mean and variance of $f_{e_N^\delta}$ using either of the methods described in Subsection 4.3.
- Step 2: Find the mean and variance of the steady state error metric values produced by the controller.
- Step 3: Perform two-sample t- and F-tests to compare the means and variances, respectively.
- Step 4: Assess the results.
  - Step 4.a: If the tests fail to refute the null hypotheses, the performance of the controller is consistent with sampling robot positions from the target distribution.
  - Step 4.b: If the two-sample t-test suggests that the means are significantly different, find the 95% confidence interval of their difference to assess the magnitude of the controller's performance surplus or deficiency.

Just as for the benchmark e_{rel} defined in Section 3, the first step is independent of choice of controller. Thus, if the goal is to compare several controllers, the computations of Step 1 need to be performed only once.

Example We apply this method to assess the performance of the controller in [25], where the desired distribution is $\rho = \rho_{\text{ring}}$ (see Definition 2), there are $N = 200$ robots, and the radius is $\delta = 2$in. These calculations were originally performed in the authors' [1].
Step 1, the computation of the mean and variance of e_N^δ when robot positions are sampled at random from ρ_{ring}, was completed in Subsection 4.3. For Step 2, we use the data presented as Figure 7 of [25] to calculate

that the distribution of steady state error metric values produced by their controller has a mean of 0.5026 with a standard deviation of 0.02586. These values are summarized in Table 1.

Table 1. Summary of mean and standard deviation of error metric values for the example in Subsection 4.4.

	Mean	Standard deviation
positions sampled from ρ_{ring}	0.4933	0.02484
steady state error values	0.5026	0.02586

Next we perform the two-sample F-test and two-sample t-test. We take the null hypothesis to be that the distribution of these error metric values is the same as $f_{e_N^\delta}$. The results are summarized in Table 2.

Table 2. Summary of statistical tests for the example in Subsection 4.4.

F-statistic	1.0831
t-statistic	8.5888

The F-statistic of 1.0831 fails to refute the null hypothesis, indicating that there is no significant difference in the standard deviations. On the other hand, a two-sample t-test rejects the null hypothesis with a t-statistic of 8.5888, indicating that the steady state error is not distributed with the same population mean as $f_{e_N^\delta}$. Nonetheless, the 95% confidence interval for the true difference between population means is computed to be merely $(0.00717, 0.01141)$. This shows that the mean steady state error achieved by this controller is unlikely to exceed that of $f_{e_N^\delta}$ by more than 2.31%. Therefore, we find the performance of the controller in [25] to be acceptable given its stochastic nature, as the error metric values it realizes are only slightly different from those produced by sampling robot positions from the target distribution.

Remark 3. As with e_{rel} of Section 3, the sentiment of this benchmark is preceded by [25]. However, here we have presented a concrete, repeatable, and objective approach that makes this benchmark suitable for general use. In particular, we have introduced into this context the use of appropriate statistical tests for comparing two approximately normal distributions. On the other hand, [25] relies on visual inspection, noting, "the error values [from simulation] mostly lie between ... the 25th and

75th percentile error values when robot configurations are randomly sampled from the target distribution". The fact that $f_{e_N^\delta}$ is approximately Gaussian (according to Theorem 1) not only allows for the use of these statistical tests, but also provides a way to greatly speed up computation. Indeed, the calculations in [25][14] took two orders of magnitude more computation than we perform in Subsection 4.3.

5 FUTURE WORK

Several open problems remain regarding the computational and theoretical aspects of our proposed performance assessment methods. For instance, the benchmark calculations scale quickly with the size of the swarm, and therefore they become unfeasible for a standard personal computer with swarms on the order of 1000 robots. To extend our proposed methods to larger swarms, computational techniques should be improved. Possibilities include the derivation of an analytical representation of the error metric PDF in terms of the target distribution, more sophisticated optimization algorithms to compute the extrema, and alternate formulations of the optimization problem that reduce the number of variables or introduce guarantees on the global optimality of a resulting solution. As for theoretical extensions, analysis of the shape and size of the robot blob function, as well as a rigorous understanding of the optimal radius, are intriguing topics for further study. Specifically, if a certain control law were characterized as "good" with one choice of blob function, would the same conclusion be reached with a different blob function? These questions were raised by the authors' [1], and here we provide possible routes to answering them, as well as connections to known results in the theory (see Subsection 4.1).

6 CONCLUSION

This work deals with the performance assessment of spatial density control of robotic swarms. We introduce a sensitive error metric and two corresponding benchmarks, one based on the realizable extrema of the error metric and one based on its probability density function, which provide methods for evaluating the performance of control algorithms. The parameters of the error metric correspond to physical properties such as robot shape and size, and can therefore be tailored to the particular system under study. The benchmark calculations then provide reference points to which the performance of different control schemes can be compared. The error metric and these benchmarks were first carefully defined in the authors' [1]; here, we have made these definitions more transparent, and, in Subsections 3.1 and 4.4, we have added step-by-step methods for performing the relevant computations. This will allow swarm designers and practitioners to more easily use these benchmarks

[14] According to the caption of Figure 2 of [25], the figure was generated as a histogram from 100,000 Monte Carlo samples.

to quantitatively decide which control scheme best fits their application. As demonstrated by the examples, including the new example featured in Subsections 3.1 and 3.2, these evaluation methods work well for a variety of target distributions and can be applied to judge the performance of both deterministic and stochastic control algorithms.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of NSF grant CMMI-1435709, NSF grant DMS-1502253, the Dept. of Mathematics at UCLA, and the Dept. of Mathematics at Harvey Mudd College. OT acknowledges support from the Charles Simonyi Endowment at the Institute for Advanced Study. The authors also thank Hao Li (UCLA) for his insightful contributions regarding the connection between this error metric and kernel density estimation.

References

1. Anderson, B., Loeser, E., Gee, M., Ren, F., Biswas, S., Turanova, O., Haberland, M., Bertozzi, A.: Quantitative assessment of robotic swarm coverage. Proc. 15th Int. Conf. on Informatics in Control, Automation, and Robotics (ICINCO 2018) **2**, 91–101 (2018)
2. Ayvali, E., Salman, H., Choset, H.: Ergodic coverage in constrained environments using stochastic trajectory optimization. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) pp. 5204–5210 (2017)
3. Barbosa, R.S., Machado, J.T., Ferreira, I.M.: Tuning of pid controllers based on bodes ideal transfer function. Nonlinear dynamics **38**(1-4), 305–321 (2004)
4. Berman, S., Kumar, V., Nagpal, R.: Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 378–385. IEEE (2011)
5. Bickel, P.J., Rosenblatt, M.: On some global measures of the deviations of density function estimates. Ann. Statist. **1**(6), 1071–1095 (1973). DOI 10.1214/aos/1176342558. URL https://doi.org/10.1214/aos/1176342558
6. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence **7**(1), 1–41 (2013)
7. Brezis, H.: Functional analysis, Sobolev spaces and partial differential equations. Springer Science & Business Media (2010)
8. Bruemmer, D.J., Dudenhoeffer, D.D., McKay, M.D., Anderson, M.O.: A robotic swarm for spill finding and perimeter formation. Tech. rep., Idaho National Engineering and Environmental Lab, Idaho Falls (2002)
9. Cao, Y.U., Fukunaga, A.S., Kahng, A.: Cooperative mobile robotics: Antecedents and directions. Autonomous robots **4**(1), 7–27 (1997)

10. Chaimowicz, L., Michael, N., Kumar, V.: Controlling swarms of robots using interpolated implicit functions. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 2487–2492 (2005). DOI 10.1109/ROBOT.2005.1570486
11. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Transactions on Robotics and Automation (2004)
12. Craig, K., Bertozzi, A.: A blob method for the aggregation equation. Mathematics of computation **85**(300), 1681–1717 (2016)
13. Cruz-Uribe, D., Neugebauer, C.: Sharp error bounds for the trapezoidal rule and simpson's rule. J. Inequal. Pure Appl. Math **3**(4), 1–22 (2002)
14. Demir, N., Eren, U., Açıkmeşe, B.: Decentralized probabilistic density control of autonomous swarms with safety constraints. Autonomous Robots **39**(4), 537–554 (2015)
15. Devroye, L., Győrfi, L.: Nonparametric density estimation: the L^1 view. Wiley (1985)
16. Elamvazhuthi, K., Adams, C., Berman, S.: Coverage and field estimation on bounded domains by diffusive swarms. In: Decision and Control (CDC), 2016 IEEE 55th Conference on, pp. 2867–2874. IEEE (2016)
17. Elamvazhuthi, K., Berman, S.: Optimal control of stochastic coverage strategies for robotic swarms. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 1822–1829. IEEE (2015)
18. Eren, U., Akmee, B.: Velocity field generation for density control of swarms using heat equation and smoothing kernels. IFAC-PapersOnLine **50**(1), 9405 – 9411 (2017). DOI https://doi.org/10.1016/j.ifacol.2017.08.1454. 20th IFAC World Congress
19. Giné, E., Mason, D.M., Zaitsev, A.Y.: The l¡sub¿1¡/sub¿-norm density estimator process. The Annals of Probability **31**(2), 719–768 (2003)
20. Hamann, H., Wörn, H.: An analytical and spatial model of foraging in a swarm of robots. In: International Workshop on Swarm Robotics, pp. 43–55. Springer (2006)
21. Hartinger, J., Kainhofer, R.: Non-uniform low-discrepancy sequence generation and integration of singular integrands. In: Monte Carlo and Quasi-Monte Carlo Methods 2004, pp. 163–179. Springer (2006)
22. Horváth, L.: On L_p-norms of multivariate density estimators. The Annals of Statistics pp. 1933–1949 (1991)
23. Howard, A., Mataric, M.J., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. Distributed autonomous robotic systems **5**, 299–308 (2002)
24. Hussein, I.I., Stipanovic, D.M.: Effective coverage control for mobile sensor networks with guaranteed collision avoidance. IEEE Transactions on Control Systems Technology **15**(4), 642–657 (2007)
25. Li, H., Feng, C., Ehrhard, H., Shen, Y., Cobos, B., Zhang, F., Elamvazhuthi, K., Berman, S., Haberland, M., Bertozzi, A.L.: Decentralized stochastic control of robotic swarm density: Theory, simulation,

and experiment. In: Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE (2017)
26. Mathew, G., Mezi, I.: Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. Physica D: Nonlinear Phenomena **240**(4), 432 – 442 (2011). DOI https://doi.org/10.1016/j.physd.2010.10.010. URL http://www.sciencedirect.com/science/article/pii/S016727891000285X
27. Moore, D.S., McCabe, G.P., Craig, B.A.: Introduction to the Practice of Statistics (2009)
28. Pimenta, L.C.A., Michael, N., Mesquita, R.C., Pereira, G.A.S., Kumar, V.: Control of swarms based on hydrodynamic models. In: 2008 IEEE International Conference on Robotics and Automation, pp. 1948–1953 (2008). DOI 10.1109/ROBOT.2008.4543492
29. Pimenta, L.C.A., Pereira, G.A.S., Michael, N., Mesquita, R.C., Bosque, M.M., Chaimowicz, L., Kumar, V.: Swarm coordination based on smoothed particle hydrodynamics technique. IEEE Transactions on Robotics **29**(2), 383–399 (2013). DOI 10.1109/TRO.2012.2234294
30. Reif, J.H., Wang, H.: Social potential fields: A distributed behavioral control for autonomous robots. Robotics and Autonomous Systems **27**(3), 171–194 (1999)
31. Schwager, M., McLurkin, J., Rus, D.: Distributed coverage control with sensory feedback for networked robots. In: robotics: science and systems (2006)
32. Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization, 2nd Edition (2015)
33. Scott, D.W., Wand, M.: Feasibility of multivariate density estimates. Biometrika **78**(1), 197–205 (1991). DOI 10.1093/biomet/78.1.197
34. Shen, W.M., Will, P., Galstyan, A., Chuong, C.M.: Hormone-inspired self-organization and distributed control of robotic swarms. Autonomous Robots **17**(1), 93–105 (2004)
35. Sloan, I.: Integration and approximation in high dimensions–a tutorial. Uncertainty Quantification, Edinburgh (2010)
36. Soysal, O., Şahin, E.: A macroscopic model for self-organized aggregation in swarm robotic systems. In: International Workshop on Swarm Robotics, pp. 27–42. Springer (2006)
37. Spears, W.M., Spears, D.F., Hamann, J.C., Heil, R.: Distributed, physics-based control of swarms of vehicles. Autonomous Robots **17**(2), 137–162 (2004)
38. Sugihara, K., Suzuki, I.: Distributed algorithms for formation of geometric patterns with many mobile robots. Journal of Field Robotics **13**(3), 127–139 (1996)
39. Wand, M., Jones, M.: Kernel Smoothing. Chapman and Hall/CRC (1994)
40. Zhang, F., Bertozzi, A.L., Elamvazhuthi, K., Berman, S.: Performance bounds on spatial coverage tasks by stochastic robotic swarms. IEEE Transactions on Automatic Control (2017)
41. Zhao, S., Ramakrishnan, S., Kumar, M.: Density-based control of multiple robots. In: Proceedings of the 2011 American Control Conference, pp. 481–486 (2011). DOI 10.1109/ACC.2011.5990615

42. Zhong, M., Cassandras, C.G.: Distributed coverage control and data collection with mobile sensor networks. IEEE Transactions on Automatic Control **56**(10), 2445–2455 (2011)