Unifying Control Architecture for Reactive Particle Swarms

Shae T. Hart, *Graduate Student Member, IEEE*, and Christopher A. Kitts, *Senior Member, IEEE*

***Abstract*—Swarm control strategies allow for decentralized control of many simple robots to perform collective behaviors based on local interactions. Despite the broad applications of swarm robotics, there lacks a generalized and unified set of control equations for swarms. First, this work presents a novel mathematical control law for decentralized velocity control of planar, homogeneous swarms called reactive particle swarms (RPS). The architecture unifies the development of new RPS behaviors. The weighted summation of simple base behaviors and external command inputs form complex composite behaviors. Second, a series of simulated and on-hardware case studies demonstrate the utility and flexibility of the architecture. Third, this work establishes a library of verified RPS base behaviors.**

***Index Terms*—Control design, control system synthesis, decentralized control, hardware-in-the-loop simulation, multirobot systems, swarm robotics, velocity control.**

Nomenclature

ACR	Artificial communication range.
$C_i(d_l)$	ACR matrix for robot i.
CM	Center of mass.
D_i	Normalization matrix for robot i
d_{ij}	Distance between robot i and j.
d_l	Specified ACR for trial.
f_b	Weighting function.
N	Number of robots in the swarm.
$\mathbb{N}$	Number of robots within ACR.
k_b	Base behavior scaling constant.
k_u	External control input scaling constant.
$\vec{r}_i$	Aggregate relative position vector.
$\vec{u}_{ui}$	External control input for robot i.
$\vec{v}_{bi}$	Base behavior command velocity for robot i.
$\vec{v}_{ci}$	Composite command velocity for robot i.
$W_i(f_b)$	Weighting matrix defined by function f_b.
$\vec{x}_i$	Pose of robot i.

Manuscript received 20 October 2021; revised 23 June 2022 and 18 August 2022; accepted 18 September 2022. Date of publication 10 October 2022; date of current version 18 April 2023. Recommended by Technical Editor Jiafu Wan and Senior Editor Hong Qiao. *(Corresponding author: Shae T. Hart.)*

The authors are with the Department of Mechanical Engineering, Santa Clara University, Santa Clara, CA 95053 USA (e-mail: sthart@scu.edu; ckitts@scu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMECH.2022.3211155.

Digital Object Identifier 10.1109/TMECH.2022.3211155

I. Introduction

The field of multirobot systems (MRSs) is a promising area of robotic research. MRSs can improve performance over single-robot systems, potentially accomplishing challenging tasks impossible for single-robot systems. An MRS can be classified by its flexibility and robustness. Flexibility is the ability to adapt to environmental changes, while robustness is the tolerance to failures in the robots in MRS [1].

A subset of MRS is robotic swarms. Swarms consist of a large number of autonomous, generally homogeneous robots using decentralized control through the use of simple rules. Compared to other MRS approaches that are more deliberate in their formation control and task coordination, swarm technology is considered advantageous for flexibility, scalability, robustness, and economy [2], [3]. In addition to being flexible in their response to their environment, swarms use lightweight, simple algorithms, allowing them to be both highly scalable in number and adaptable to the addition and removal of many robots [4], [5]. Swarm robots are considered interchangeable if they have the same capacity. This idea, coupled with swarms' ability to scale in number, reduces the overhead for handling robot failures making swarms more robust [1]. Robots should have streamlined hardware, including memory, computation power, and communication range making them inexpensive to produce and more economical [6].

The central theme of swarm robotics is simple, interchangeable robots with simple behaviors. However, the field lacks consensus about what is "simple" and whether all robots should be homogeneous or fall into a few categories [7]. Additionally, some swarms emphasize reactivity while other swarms allow for substantial computational delays for social deliberation. Reactivity enables robots to respond quickly to the environment, while social deliberation focuses more on the long-term goal. Both approaches are valid for different applications [1].

Swarm robotics has four primary categories: 1) Supervisory swarms, focusing on leader–follower hierarchical organization [2], [8], 2) consensus swarms, using consensus filtering to help the swarm reach a consensus about the desired behavior [9], [10], 3) swarm intelligence, trying to find optimal behaviors [4], [11], and 4) reactive particle swarms (RPSs), focusing on simple and reactive algorithms.

RPS, used in our work, consist of many homogeneous robots. The control algorithm should be scalable, preferably mathematical, with lightweight, explainable, and reactive behaviors that do not require consensus filtering or a hierarchy of robots. RPS

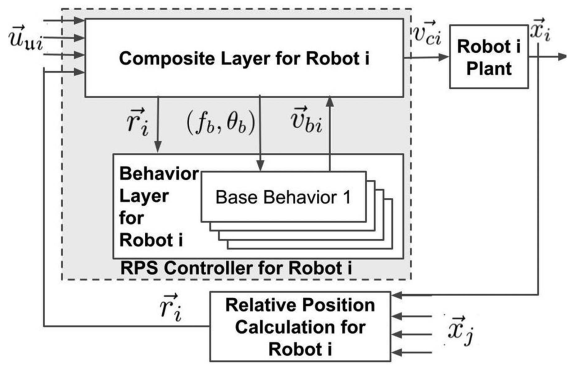

Fig. 1. Control architecture for single RPS robot.

algorithms should not require global knowledge to work and do not store past swarm state information because this leads to nonreactive control. RPS behaviors are based on the relative positions of robots within a limited communication range and use this information to accomplish simple behaviors like stable aggregation, collision avoidance [12], [13], [14], coverage [15], [16], formation control [13], [17], [18], [19], and gradient estimation [17], [18]. RPSs use the superposition of multiple simple behaviors to accomplish more complex tasks like cooperative hunting: a combination of formation control, obstacle avoidance, and target tracking [12].

While the control equations used in previous RPS research are mathematically very similar, there lacks a unifying control architecture that ties together their mathematics. The development of a unified framework is the primary motivation for this article. Such a unified framework would not only unite the existing behaviors but streamline the development of novel behaviors. In literature, Mesbahi and Egerstedt [16] present the most developed RPS control architecture. Their work presents methods using graph theory for analyzing the connections in stationary networks and mobile robots. The robot's command velocity is calculated as the weighted sum of the relative position vectors between robots. Different behaviors are designed by weighting the relative position vectors differently. Mesbahi and Egersted's method has been used for formation control, rendezvous problems, the containment problem [16], and maintaining connections with different barrier functions [20].

McLurkin proposed developing a library of base swarm behaviors [15]. Once created, base behaviors are superimposed to achieve more complex behaviors. When base behaviors are superimposed, certain base behaviors can be emphasized over others to control the swarm's performance. This plug-and-play design concept allows for the rapid development of novel combinations of base behaviors.

The primary contribution of this article is the presentation of the RPS control architecture, which is based on a unified, compact set of mathematics. This is accomplished through the extension of Mesbahi and Egersted's mathematics and the incorporation of McLurkin's plug-and-play approach. This architecture forms simple base behaviors as weighted sums of the relative position vectors between robots, and more complex composite behaviors are created as weighted sums of base behaviors. As a second contribution, this architecture's versatility is demonstrated by implementing a variety of swarm behaviors through a series of simulated and hardware-in-the-loop case studies. A third contribution consists of establishing a library of base behaviors that allow a control engineer to intentionally design and manipulate how the RPS performs by selecting specific behaviors and blending them with others through the scaling constants.

The rest of the article is arranged as follows. Section II examines the two layers of the control architecture. Then, the architecture's flexibility is demonstrated through a series of simulated and hardware case studies. Section III specifies design and assumptions used in the case studies. Sections IV, V, and VI present a series of case studies. Finally, Section VII summarizes the contributions of this work.

II. RPS Control Architecture

The following section outlines the planar RPS control architecture and its two levels, as shown in Fig. 1. In this block diagram, the pose of robot i, $\vec{x}_i$, is used along with the other swarm robots' poses, $\vec{x}_j$'s, to determine the relative position vector, $\vec{r}_i$. The RPS controller for robot i uses this relative position vector and external control inputs, $\vec{u}_{ui}$, to generate a velocity command for the robot.

The behavior layer is the lower level of the RPS controller and defines the control equation for a base behavior. Base behaviors are formed as the weighted sum of the relative position vectors between robots. The upper level is the composite layer, which defines the control equation for composite behaviors. Composite behaviors are designed as weighted sums of base behaviors and external inputs. For example, a simple flocking algorithm could include a base behavior of attracting toward the center of mass and an external input of go-to coordinates.

A. Behavior Layer Command Equations

This section describes the governing equation for the behavior layer, which calculates the command velocity for each base behavior. A reference vector is calculated as the weighted sum of a subset of the relative position vectors to the other robots in the swarm, using the weighting function f_b. Then, the reference vector is rotated by the angle θ_b to define the command velocity for the base behavior. Given this approach, the base behaviors are defined as a weighting function, f_b, and angle, θ_b pair. The command velocity, $\vec{v}_{bi}$, for the bth RPS base behavior for robot i is calculated in (1).

$$\vec{v}_{bi} = K(\theta_b) W_i(f_b) C_i(d_l) D_i \vec{r}_i. \quad (1)$$

In (1), $\vec{r}_i$ is the aggregate relative position vector, D_i is the normalization matrix, $C_i(d_l)$ is the artificial communication range matrix for artificial communication range d_l, $W_i(f_b)$ is the weighting matrix calculated by the weighting function f_b, and $K(\theta_b)$ is a rotation matrix using angle θ_b. The subsequent paragraphs will discuss the details of each term.

Starting on the right-hand side of (1), $\vec{r}_i \in \mathbb{R}^{2N}$ is the aggregate relative position vector which contains the relative position vectors pointing from robot i to all N robots in the swarm, $\vec{r}_{ij}$.

If a robot is outside the communication range from robot i, it is assumed to be infinitely far away at coordinates $[\infty\ \infty]^T$. The reason this is valid will be explained later in this section when discussing the $C_i(d_l)$ matrix.

$$\vec{r}_i = \begin{bmatrix} \vec{r}_{i1} & \vec{r}_{i2} & \dots & \vec{r}_{iN} \end{bmatrix}^T. \tag{2}$$

Next, matrix $D_i \in \mathbb{R}^{2N\times 2N}$ is the normalization matrix. Here d_{ij} is the distance between robot i and robot j. D_i is expressed using Iverson bracket and set notation in (3). This diagonal matrix normalizes the relative position vectors, $\vec{r}_{ij}$, to relative unit vectors, $\hat{u}_{ij}$, as in (4).

$$D_i = \text{diag}\left(\left\{[\![(j \neq i)]\!]\frac{1}{d_{ij}}I_2\right\}_{j=1}^{N}\right) \tag{3}$$

$$\begin{bmatrix} \hat{u}_{i1} & \hat{u}_{i2} & \dots & \hat{u}_{iN} \end{bmatrix}^T = D_i\vec{r}_i. \tag{4}$$

Next, matrix $C_i(d_l) \in \mathbb{R}^{2N\times 2N}$ is the artificial communication range matrix for robot i. This diagonal matrix has elements defined in terms of d_l, a constant distance parameter defined for the swarm. The equation for $C_i(d_l)$ is expressed in (5).

$$C_i(d_l) = \text{diag}\left(\{[\![(j \neq i) \wedge (d_{ij} \leq d_l)]\!]I_2\}_{j=1}^{N}\right). \tag{5}$$

The matrix $C_i(d_l)$ acts as a selection matrix. The product $C_i(d_l)D_i\vec{r}_i$ produces a $2N$ long vector containing the relative unit vectors pointing from robot i to all other robots within a communication range of d_l. For a robot j that is beyond a distance d_l from robot i, including robots that are out of communication range, the resultant vector is the zero vector $[0\ 0]^T$. Therefore, d_l acts as an artificial communication range, ACR. For this to be valid, d_l must be less than or equal to the actual communication range of the robots.

Next, a reference vector, $\vec{v}_{ri}$, is calculated as a weighted sum of the unit vectors found by $C_i(d_l)D_i\vec{r}_i$, as shown in (6)–(8). Instead of using scalar weights, the unit vectors are premultiplied by matrices, W_j.

$$\begin{aligned} \vec{v}_{ri} &= \begin{bmatrix} W_{i1} & W_{i2} & \dots & W_{iN} \end{bmatrix} C_i(d_l)D_i\vec{r}_i \quad (6)\\ &= \begin{bmatrix} f_b(i,1) & f_b(i,2) & \dots & f_b(i,N) \end{bmatrix} C_i(d_l)D_i\vec{r}_i \quad (7)\\ &= W_i(f_b)C_i(d_l)D_l\vec{r}_i. \quad (8) \end{aligned}$$

Each $W_{ij} \in \mathbb{R}^{2\times 2}$ is calculated using a weighting function $f_b(i,j)$. The weighting function is a mathematical equation that is a function of swarm state, sensor readings, and similar information, returning a weight matrix for the pair of robots. When augmented together, the N matrices form the weighting matrix $W_i(f_b) \in \mathbb{R}^{2\times 2N}$. The behavior of the swarm can be changed by weighting vectors differently. While not explicitly stated in the weighting functions, $f_b(i,j) = 0_2$ for robots that are outside communication range. This convention prevents ambiguity in the evaluation of the weighting function.

For example, the weighing function, expressed in (9), calculates the reference vector pointing from robot i to the local center of mass, CM. In this equation, $\mathbb{N}$ is the number of robots within the ACR of robot i. Conversely, the weighting function in (10) uniformly weights all robots within the ACR, finding a reference vector parallel to the average of the relative unit vectors. The reference vector for uniform weighting is less influenced by outlier robots far from the swarm than the CM reference vector.

$$f_{\text{CM}}(i,j) = \frac{d_{ij}}{\mathbb{N}}I_2 \tag{9}$$

$$f_{\text{uniform}}(i,j) = I_2. \tag{10}$$

The final matrix in (1), $K(\theta_b)$, is a rotation matrix and controls how robot i moves relative to this reference vector, using θ_b. The equation for $K(\theta_b)$ is expressed in (11).

$$K(\theta_b) = \begin{bmatrix} \cos\theta_b & -\sin\theta_b \\ \sin\theta_b & \cos\theta_b \end{bmatrix}. \tag{11}$$

Returning to the CM reference vector, we consider how changing θ_b changes robot i's motion. When θ_b is 0 deg, the robot will move along the reference vector toward the local CM, while if θ_b is 180 deg, the robot will move away from the local CM. When θ_b is 90 or 270 deg, the robot will orbit the CM. The robots orbit in a clockwise manner when $\theta_b = 90$ deg and counter-clockwise for $\theta_b = 270$ deg.

In conclusion, (1) calculates the command velocity for a base behavior using a weighting function, f_b, and angle, θ_b.

B. Composite Layer Command Equation

The behavior layer for robot i calculates the command velocity for each base behavior in a composite behavior. Then, the composite layer calculates the composite command velocity, $\vec{v_{ci}}$, as the weighted sum of the unit vectors of the base behavior command velocities as given in (12).

$$\vec{v}_{ci} = \sum_{b=1}^{p} k_b \frac{\vec{v}_{bi}}{||\vec{v}_{bi}||} + \sum_{\text{u}=1}^{q} k_{\text{u}} \frac{\vec{u}_{\text{u}i}}{||\vec{u}_{\text{u}i}||}. \tag{12}$$

Here there are p base behaviors with their associated command velocities, $\vec{v}_{bi}$, and q external control inputs, $\vec{u}_{\text{u}i}$. The external control inputs allow external controllers to influence the motion of robots. These external inputs can be trajectory following commands, go-to coordinates commands, and similar commands.

In this equation, the scaling constants, k_b and k_{u}, emphasize certain base behaviors over others. These scaling constants must be positive and satisfy the constraint equation given in (13). When this constraint is satisfied, the scaling constants represent what percent of the total composite behavior each behavior or control input contributes. Behaviors associated with higher scaling constants will dominate the performance of the composite behavior over those with lower scaling constants.

$$\sum_{b=1}^{p} k_b + \sum_{\text{u}=1}^{q} k_{\text{u}} = 1 \qquad k_b > 0 \qquad k_{\text{u}} > 0 \tag{13}$$

C. Summary of RPS Control Architecture

In summary, Fig. 1 shows the control architecture for RPS with the RPS controller highlighted in gray. First, the relative position vector, $\vec{r}_i$, is calculated from the poses of the robots. The composite layer then passes the necessary pairs of weighting

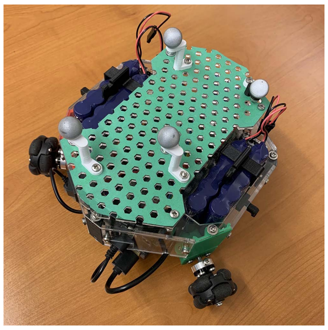

Fig. 2. One of the robots used by SBS for hardware trials.

functions and angles, (f_b, θ_b) for each base behavior to the behavior layer. At the behavior layer, the RPS controller calculates each base behavior's velocity commands, $\vec{v}_{bi}$, using (1). Then, the composite layer uses (12) with the scaling constants, k, that are constrained by (13), and external control inputs, $\vec{u}_{ui}$, to calculate the final command velocity, $\vec{v}_{ci}$. Robot i executes this velocity command, leading to the updated pose for robot i, $\vec{x}_i$.

III. Case Study Design

The subsequent three sections use a series of case studies to explore the RPS Architecture. These case studies are designed to establish a library of RPS behaviors based on existing swarm behaviors in the literature and highlight aspects of the RPS architecture. Since the focus of this article is on high-level control, the case studies use simple holonomic robots for generality instead of one specific type of nonlinear robot. Santa Clara University's swarm behavior simulator (SBS) was used to perform the case studies [21]. SBS uses Simulink for the rapid design of swarm behaviors. Evaluation of these behaviors is then supported via numerical simulation or through the control of real hardware robots. Simulation supports the inclusion of robot dynamics, swarm sizes of up to hundreds of robots, and the ability to perform bulk simulations for a behavior using randomized initial conditions to evaluate behaviors statistically. The hardware platform uses up to 12 custom omnidrive robots, shown in Fig. 2, that operate within a 75-square meter workspace that uses an Optitrack system for pose estimation. The specifics of the robots and the testbed are discussed in [22].

The robots moved at a constant speed of 0.1 m/s. This speed was selected based on the robot dynamics used in SBS to avoid saturating the wheel speeds. In each case study, the simulation times were selected based on the expected distances the robots would travel for that behavior.The ACR for each case study was generally set to be large enough to cover the entire workspace, allowing all robots in the field to be used to demonstrate the behavior clearly. The only exception is in Section IV-A which studies how the limited ACR affects the behavior performance.

To avoid collisions, SBS used a common obstacle avoidance function. If two robots were within some set distance of each other, d_{OA}, the velocity command was calculated using (14), instead of (12). In each sim, d_{OA} is at least 0.1 m because that is how far each of the wheels is from the center of the robot for

TABLE I
List of Investigative Simulations for Base Behaviors

Case study	N	ACR (m)	d_{OA} (m)	f_b	θ_b (deg)
CM attract	15	50	5	f_u	0
Uniform attract	15	50	5	f_{cm}	0
Nearest n neighbor	20	10	0.25	$f_{\tilde{n}}$	180
Ring attract	10	50	0.25	f_{ring}	0
Elliptical attract*	10	50	0.3	$f_{ellipse}$	0
Formation maintenance	12	10	0.1	f_{form}	0
Elliptical orbit	10	50	0.1	$f_{ellipse}$	270

* indicates the inclusion of a hardware trial.

TABLE II
List of Investigative Simulations for Composite Behaviors

Case study	N	ACR (m)	d_{OA} (m)	Behavior	Scaling
Uniform ring*	10	50	0.25	$(f_{ring}, 0)$	0.9
				$(f_{\tilde{n}}, 180)$	0.1
Simple flocking*	10	10	0.25	$(f_{CM}, 0)$	$0.3, 0.5, 0.7$
				$\vec{u}_{GoTo}$	$0.7, 0.5, 0.3$

* indicates the inclusion of a hardware trial.

the simulated and hardware robots. Similarly, some case studies presented require a boundary to contain the robots. When the robot was on the boundary, the velocity command was normal to the boundary, pointing inward. Otherwise, the commanded velocity followed that in (12).

$$v_{OA} = [\![d_{ij} \le d_{OA}]\!] \frac{-1}{d_{ij}^{-4}} \vec{r}_{ij} \tag{14}$$

As weighting functions are presented, they are analyzed for behaviors with $\theta_b = 0$ or $180\,\text{deg}$ as in Section IV and then with $\theta_b = 90$ or $270\,\text{deg}$, as in Section V. Then, Section VI presents two composite behaviors case studies. For some case studies, multiple simulations were performed to verify the weighting function behavior using performance parameters. The performance parameter's mean and standard deviation (SD) were reported for single batches of simulations. Multiple sets of simulations were performed for specific case studies to investigate different aspects of the behavior. If there were only two distributions to compare, then a t-test was performed, which returns a p-value. A p-value less than 0.05 indicates that the distributions are different. If more than two sets of simulations were performed, a one-way ANOVA calculated a p-value for the distributions of performance parameters. If $p < 0.05$, then at least one of the distributions of performance parameters is significantly different from the others, and a *post hoc* comparison is required. The Tukey–Kramer *post hoc* test was used, which returns a p-value for each pair of batch simulations. Again, if this value is less than 0.05, then the two distributions of performance parameters are statistically different. For all analyses, outliers were defined as 1.5 times the interquartile range above the 75th percentile or below the 25th percentile.

Table I summarizes the simulation parameters used for batch simulations for the base behavior case studies. Table II summarizes the parameters for composite behavior case studies.

IV. Base Behavior Case Studies With $\theta_b = 0$ or $180\,\text{deg}$

This section explores how weighting functions perform when moving parallel to the reference vector. Attract behaviors move

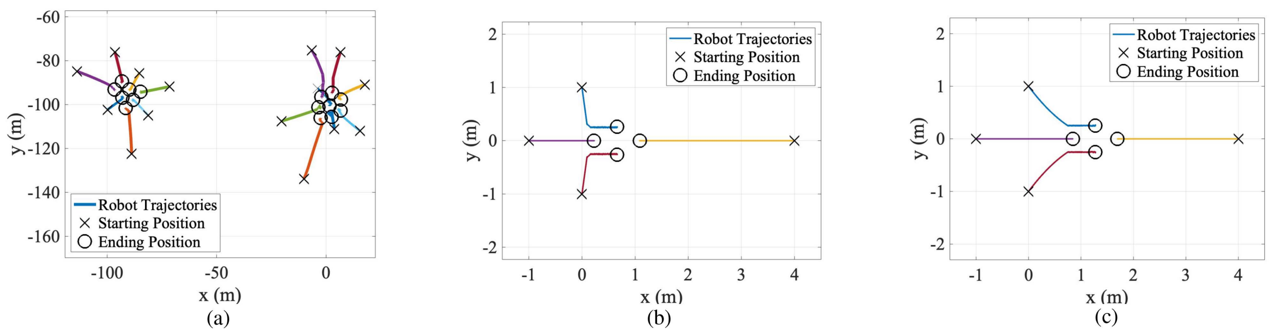

Fig. 3. Trajectories for (a) Universal attract with N = 15. (b) Universal attract with N = 4. (c) CM attract with N = 4.

along the reference vector using $\theta_b = 0\,\text{deg}$ while disperse behaviors move along the negative of the reference vector using $\theta_b = 180\,\text{deg}$.

A. CM and Uniform Weighting Functions

The first case study uses the two weighting functions introduced in Section II: 1) Center of mass (CM) weighting as in (9) and 2) uniform weighting as in (10). These behaviors are adapted from the behaviors presented in [14].

Next, three simulations demonstrate the nuances in these two formulations. First, 15 robots are simulated over 500 s using uniform attraction as in (10) and $\theta_b = 0\,\text{deg}$ with an ACR of 50 m and a d_{OA} of 5 m. Fig. 3(a) plots the swarm trajectory. Because the initial conditions result in some robots being outside of ACR of the others, two groups form. The robots settle, minimizing the distances between each other while honoring the d_{OA} set.

Second, Fig. 3(b) and (c) presents the trajectories for two simulations demonstrating the difference between uniform attraction and CM attraction, for the same initial conditions. The four robots had a d_{OA} of 0.5 m and an ACR of 10 m. As expected, the motion of uniform attraction is less influenced by outlier robots far from the swarm than CM attract.

For each of the two behaviors, 100 simulations were performed using 15 robots, an ACR of 50 m, and a d_{OA} of 5 m for 1000 s. For each simulation, three parameters were calculated: 1) The average final distance between the robots to the CM, 2) the maximum final distance between a robot and the CM, and 3) the minimum final distance between the robots.

The mean over 100 trials of the average distances to the CM was 6.869 m (SD = 0.074 m) for uniform attraction and 6.878 m (SD = 0.084 m) for CM attraction, indicating that both behaviors have robots converging together. Next, the mean maximum distance to the CM for the 100 trials was 10.145 m (SD = 0.357 m) for uniform attract and 10.139 m (SD = 0.336 m) for CM attract. This indicates that for both behaviors, the robot furthest from the CM in each simulation was not too far away from the overall swarm. Finally, the mean minimum distance between a pair of robots was 4.988 m (SD = 0.003 m) for both uniform and CM attraction, which indicates that the obstacle avoidance function is performing nominally. Overall, these three analyses indicate that both behaviors result in good clustering for an appropriate ACR while honoring the obstacle avoidance function.

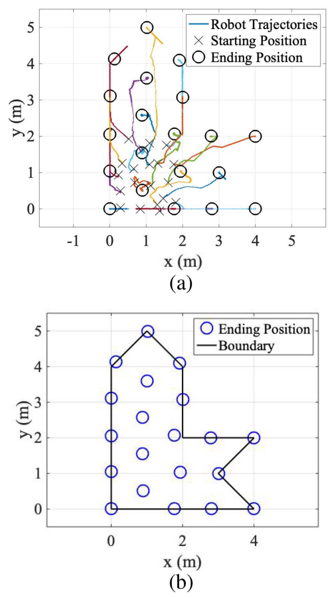

Fig. 4. Uniform disperse for n = 1, (a) trajectory. (b) final positions and boundaries.

B. Nearest n Neighbor Weighting Function

The coverage problem of evenly distributing robots through space has been extensively studied. While there are various methods in the literature, this work uses the simple yet effective version presented in [15]. Nearest n neighbor dispersion is dispersing from n closest neighbors. Changing the number of neighbors a robot disperses from will give different results. Dispersing from the one or two closest neighbors will give good coverage of the space, while dispersing from all neighbors will push the robots to the boundaries of the space. The distance to robot i's nth closest neighbor is denoted as $d_{\tilde{n}}$, then the weighting function for the *n*-nearest neighbor attraction is given in (15). Coupled with $\theta_b = 180\,\text{deg}$, (15) becomes a disperse behavior.

$$f_{\tilde{n}}(i,j) = [\![d_{ij} \leq d_{\tilde{n}}]\!] d_{ij} I_2. \tag{15}$$

The simulated results use $n = 1$ for 20 robots with a d_{OA} of 0.25 m and an ACR of 10 m. The trajectories are presented in Fig. 4(a) with robots starting at random initial conditions. As expected, the robots move to distribute evenly through the space.

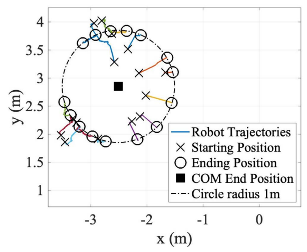

Fig. 5. Uniform ring attraction with $a = 1$ m and $N = 15$ trajectory.

An artificial boundary confined the motion of the robots to avoid the robots dispersing infinitely. Fig. 4(b) shows this boundary along with the final positions of the robots.

In 100 simulations, 20 robots were simulated for 500 s with random initial conditions in a 10 m by 10 m area. The robots had a d_{OA} of 0.25 m and an ACR of 10 m. In these simulations, all robots converged to be inside the desired boundary. Over the 100 simulations, the mean minimum distance between robots was 0.991 m (SD = 0.019 m), indicating that the robots were distributing through the space using $f_{\tilde{n}}$ instead of obstacle avoidance. These results agree with those found in [15].

C. Ring Attract Weighting Function

Another useful behavior is attracting until each robot is a distance a away from the local CM. Ring attraction uses the weighting function in (16) with $\theta_d = 0\,\mathrm{deg}$. Here $d_{i,\mathrm{cm}}$ is the distance robot i is from the local CM.

$$f_{\mathrm{ring}}(i,j) = (d_{i,\mathrm{cm}} - a)I_2. \quad (16)$$

A simulation of this attract behavior is presented for 15 robots moving to a circle with a radius of 1 m. The simulation uses an ACR of 50 m and a d_{OA} of 0.25 m. The trajectories for this simulation are presented in Fig. 5 with robots starting at random initial conditions. All the robots move to be 1 m away from the CM. However, the robots did not uniformly distribute around the circle.

In a simulation of 100 trials, 10 robots were commanded to a circle of radius 1 m. Robots had a d_{OA} of 0.25 m and an ACR of 50 m. Robots started in random conditions in a 40 m by 40 m area. The performance parameters were calculated at the end of the simulation. First, the mean minimum distance between robots was 0.260 m (SD = 0.007 m), indicating the obstacle avoidance function was determining the robot spacing on the circle. Second, the mean average distance to the CM was 1.000 m (SD = 0.002 m). This result indicates that the swarm accurately reached the desired ring radius. Finally, the mean maximum distance to the CM was 1.009 m (SD = 0.001 m), indicating that all robots converged well to the circle.

D. Elliptical Weighting Function

In some cases, it is useful for the robots to attract in an elliptical shape with a preferential direction of elongation as in [23]. This case study presents two versions of elliptical weighting, 1) where the orientation of the ellipse is given, and 2) where swarm robot positions determine the direction of elongation. Both versions use $\theta_b = 0\,\mathrm{deg}$.

1) Elliptical Weighting for Given Orientation:

Assume that the angle that the semi-major axis of the desired ellipse makes with the x-axis is given as ϕ . Then, the weighting function for elliptical weighting can be expressed as follows:

$$f_{\mathrm{ellipse}}(i,j) = \frac{d_{ij}}{\mathbb{N}}(\alpha M + \beta I_2) \quad (17)$$

$$M = \begin{bmatrix} -\cos 2\phi & -\sin 2\phi \\ -\sin 2\phi & \cos 2\phi \end{bmatrix}. \quad (18)$$

The second term in (17) attracts the robots toward the CM while the first term disperses along a line through the CM with an angle of ϕ to the x-axis. The CM attraction term keeps the robots from infinitely dispersing along that line. The terms α and β control how eccentric the resulting swarm is, requiring that $\alpha + \beta = 1$. As β increases, the robots will aggregate into less eccentric shapes. Note for stability $\beta \geq \alpha$ is required.

2) Elliptical Attract Without Given Orientation:

If a preferential direction ϕ is not provided, ϕ can be determined by the robot using the angle between the vector pointing to its furthest neighbor and the x-axis. Then, (17) can be used as before to weight the robots elliptically.

$$\phi = \arctan\left(\frac{y_{j\_\mathrm{furthest}} - y_i}{x_{j\_\mathrm{furthest}} - x_i}\right). \quad (19)$$

This case study presents three hardware trials for an elliptical attraction without a given ϕ. Each trial uses 10 robots, a d_{OA} of 0.3 m, and an ACR of 50 m. Robots start at the same initial conditions for each trial. The three trials use three different pairs of α and β: 1) $\alpha = 0.5$ and $\beta = 0.5$, 2) $\alpha = 0.45$ and $\beta = 0.55$, and 3) $\alpha = 0.4$ and $\beta = 0.6$. Fig. 6 presents the trajectories for the three case studies. The eccentricities of the final positions, calculated using a bounding ellipse, were 1, 0.9238, and 0.6663, respectively. As expected, the robots assembled into a line when α and β were equal. Then as β increased, the CM attraction was stronger, resulting in less eccentric shapes. Since ϕ was not defined, the ellipses were orientated arbitrarily.

In 100 simulations, 10 robots were simulated for 5000 s with an ACR of 50 m and a d_{OA} of 0.3 m. Robots started with random initial conditions in a 40 m by 40 m area. Three values of α were used: 0.1, 0.3, and 0.45. The eccentricity of the final swarm position was determined using a bounding ellipse. The mean and SD for eccentricity were 1) for $\alpha = 0.1$, 0.585 (SD = 0.067), 2) for $\alpha = 0.3$, 0.739 (SD = 0.112), and 3) for $\alpha = 0.45$, 0.915 (SD = 0.017).

A one-way ANOVA revealed that there was a statistically significant difference in the mean eccentricity for at least two groups $[F(2, 297) = 471.742, p < 0.0001]$. All p-values for the *post hoc* comparisons were less than 0.0001, indicating that these distributions are different, as expected. The larger the value of α, the more eccentric the resulting ellipse. However, for a given α, there is significant variation in the eccentricities of the resulting ellipse. The elliptical attract behavior found in [23] uses formation control to achieve coverage over an ellipse with specified parameters. While our formulation does not need formation control for coverage, it does have less control over the eccentricities of the resulting ellipse. Future work will develop algorithms that yield less variable properties of the ellipse.

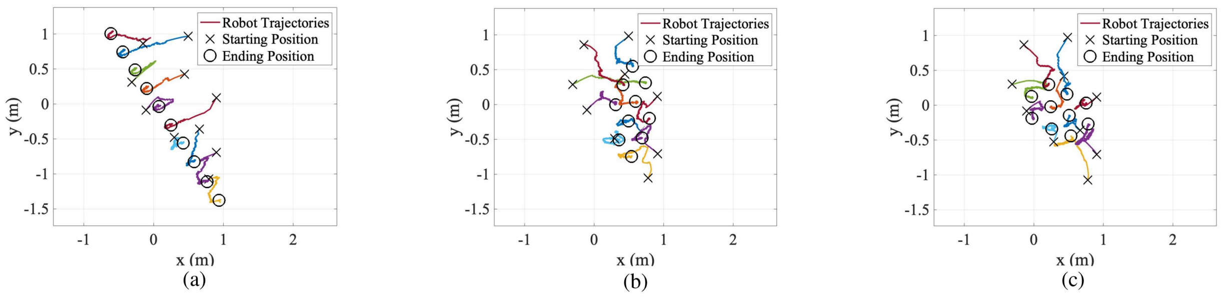

Fig. 6. Trajectories for elliptical attract with (a) $\alpha = 0.5,\ \beta = 0.5$; (b) $\alpha = 0.45,\ \beta = 0.55$; and (c) $\alpha = 0.4,\ \beta = 0.6$.

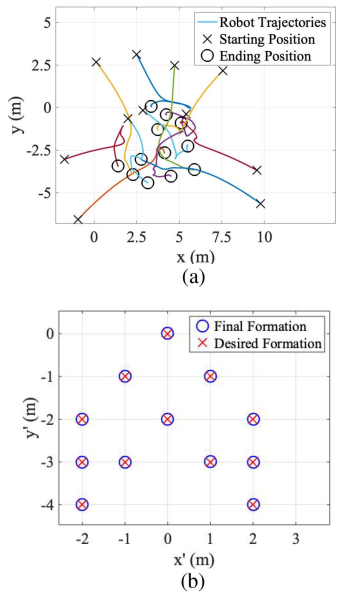

Fig. 7. Formation maintenance case study. (a) Trajectory in the global coordinate system. (b) Comparison of final and desired positions in the formation coordinate system.

E. Formation Maintenance Weighting Function

The next case study presents formation maintenance, keeping the robots in an already achieved formation. Pure formation maintenance consists of one attract behavior, $\theta_b = 0\,\text{deg}$, as described in (20), where s_{ij} is the desired distance between two robots in the formation [24]. While this case study, using the formulation in [13], presents successful formation acquisition, local minimums prevent global convergence to the desired formation [13]. Therefore, this formulation is much more valuable for formation maintenance than formation acquisition.

$$f_{\text{form}}(i,j) = (d_{ij} - s_{ij})I_2. \tag{20}$$

In this case study, formation maintenance commands 12 robots to a chevron formation represented as the red x's in Fig. 7(b). Fig. 7(b) shows the robots in the formation coordinate system. This system uses axes x' and y' centered at the peak robot with the y-axis parallel to the vector pointing from the center robot at coordinates $[0, -2]$ to the peak robot. The simulated robots have an ACR of 10 m and a d_{OA} of 0.1 m and start with random initial conditions. Fig. 7(a) presents the time history for the swarm representing the final positions of the robots as back circles. By transforming these final coordinates into the formation coordinate system and overlaying the desired formation as in Fig. 7(b), it is clear that the robots achieve close to the desired formation.

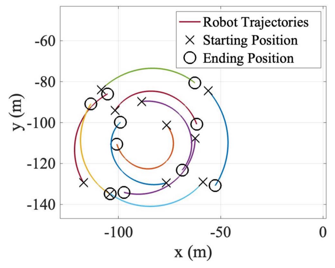

Fig. 8. Swarm trajectory for universal orbit case study.

Formation maintenance was simulated using 12 robots, an ACR of 10 m, a d_{OA} of 0.1 m, and random initial conditions in a 40 m by 40 m area for 1000 s. At the end of each simulation, the maximum absolute error in the desired distances between robots was determined. For the 100 simulations, the mean of the maximum absolute error was 0.73% (SD = 0.17%), which indicates very good formation maintenance. This error is due to the constant velocity used in SBS. There were six outlier simulations. The mean absolute error of the outliers was 1.32% (SD = 0.0294%). These outliers correspond to initial conditions that failed to converge to the desired formation.

V. Base Behavior Case Studies With $\theta_b = 90$ or $270\,\text{deg}$

This section explores how weighting functions perform when coupled with $\theta_b = 90$ or $270\,\text{deg}$, creating orbit or side-slipping behaviors.

A. Uniform and CM Weighting Functions

Just as (9) and (10) attract toward the CM or uniformly when paired with $\theta_b = 0\,\text{deg}$, the equations can orbit the CM or the average unit vector when paired with $\theta_b = 90$ or $270\,\text{deg}$. In this case study, 10 robots perform a uniform orbit behavior using (10) and $\theta_b = 90\,\text{deg}$. The simulated robots have a sensor range of 300 m and a d_{OA} of 0.5 m. The time history shown in Fig. 8 has random initial conditions. Here the robots rotate in a clockwise

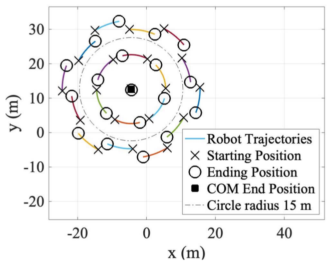

Fig. 9. Swarm trajectory for ring orbit case study.

manner around the average unit vector. If θ_b was $270\,\mathrm{deg}$, the robots would orbit counter-clockwise. Since the performance of these behaviors was already demonstrated in the previous section, no batch simulations were performed.

B. Ring Attract Weighting Function

The next case study examines using the ring weighting function as in (16) with $\theta_b = 90\,\mathrm{deg}$. A radius of 15 m is set for the ring. The simulation of 19 robots over 75 seconds uses an ACR of 300 m and a d_{OA} of 1 m. The robots were given initial conditions with one robot at the CM, six robots evenly spaced on a circle with a radius of 10 m, six robots on a circle of radius 17.32 m, and six robots on a circle of radius 20 m. The robots were nicely distributed to make a planar lattice to more easily demonstrate the behavior.

The trajectory for the simulation is plotted in Fig. 9. The square is the CM of the robots, and the dashed line shows a circle of radius 15 m centered at the CM. The robots less than 15 m all have reference vectors $\vec{v}_{ri}$ pointing away from the CM. Therefore, as expected, these robots orbit the CM counter-clockwise for $\theta_b = 90\,\mathrm{deg}$. Conversely, robots more than 15 m from the CM have reference vectors pointing toward the CM, resulting in a clockwise rotation about the CM. No batch simulations were performed since the performance of f_{ring} was demonstrated in the previous section.

C. Elliptical Weighting Function

Elliptical orbiting uses the same weighting function described in (17) but with $\theta_b = 270\,\mathrm{deg}$. Three simulations are presented using elliptical orbiting with $\phi = 45$ deg. All trials have 10 robots with an ACR of 50 m and a d_{OA} of 0.1 m. These three simulations have the same robot initial conditions but use three different sets of α and β: 1) $\alpha = 0.45, \quad \beta = 0.55$, 2) $\alpha = 0.4, \quad \beta = 0.6$, and 3) $\alpha = 0.3, \quad \beta = 0.7$. The resulting trajectories are presented in Fig. 10. As expected, the robots all orbit in a counter-clockwise manner. Visual inspection shows that as β increases, the eccentricities of the orbits decrease. Barnes et al. [25] demonstrated a similar orbiting behavior with a different formulation.

Next, 100 simulations were run for five different versions of elliptical orbit. Ten robots were simulated with an ACR of 50 m, a d_{OA} of 0.1 m, and a ϕ of 45 deg. Robots started with random initial conditions in a 40 m by 40 m area, and the simulation lasted 1000 s. For each simulation, a bounding ellipse determined the eccentricity and orientation of the trajectories. Five different values of α were used: 0.1, 0.2, 0.3, 0.4, and 0.45.

TABLE III
MEAN AND SD FOR ORIENTATION AND ECCENTRICITY FOR THE FIVE TYPES OF ELLIPTICAL ORBIT SIMULATIONS

	Eccentricity		Angle (deg)	
α	Mean	SD	Mean	SD
0.1	0.440	0.074	47.97	12.51
0.2	0.614	0.045	44.76	5.43
0.3	0.758	0.027	44.56	3.19
0.4	0.887	0.010	44.86	1.12
0.45	0.945	0.006	44.96	0.74

TABLE IV
TUKEY–KRAMER *POST HOC* COMPARISON p-VALUES FOR ELLIPTICAL ORBITING ANGLE DISTRIBUTIONS

		α_1				
		0.1	0.2	0.3	0.4	0.45
α_2	0.1	–	0.0029*	0.0012*	0.0043*	0.0064*
	0.2		–	0.9995	1.0000	0.9995
	0.3			–	0.9974	0.9921
	0.4				–	1.0000
	0.45					–

* indicates significant results.

Table III presents the mean and SD of the trajectory eccentricities for each α. The one-way ANOVA test indicated that there was at least one distribution of eccentricities that was statistically significantly different $[F(4, 495) = 2542.5, p < 0.0001]$. All pairs of p-values for the Tukey–Kramer test are less than 0.0001. Therefore, higher values of α yield more eccentric ellipses, as expected.

Next, Table III lists the mean and SD for the angle ϕ. The one-way ANOVA indicated that at least one distribution of angles was different $[F(4, 495) = 5.180, p < 0.0001]$. Table IV presents the p-values for the *post hoc* comparisons. From this analysis, only the distribution of angles with an α of 0.1 significantly differs from the others. This makes sense because at low values of α, the ellipse is much more circular, leading to larger errors when the bounding ellipse calculates the angle. All other groups have distributions that agree well with the prescribed 45° angle. But overall, for elliptical orbiting, there is good adherence for the angle ϕ, and higher values of α give more eccentric shapes.

VI. COMPOSITE BEHAVIOR CASE STUDIES

This section demonstrates how to design composite behaviors and how scaling constants selection leads to different performances.

A. Composite Case Study: Uniformly Spaced Ring

Section IV-C presented a base behavior that drove the robots to a nonuniform spacing on a ring of radius a. To have the robots uniformly distributed, we present a composite behavior with two base behaviors: 1) Ring attract as in (16) with $\theta_{\mathrm{ring}} = 0\,\mathrm{deg}$, and 2) nearest n neighbor disperse as in (15), with $n = 1$ and $\theta_{\tilde{n}} = 180\,\mathrm{deg}$. Because the dominant behavior should be driving to the ring, then $k_{b\_\mathrm{ring}} = 0.9$ and $k_{b\_\tilde{n}} = 0.1$.

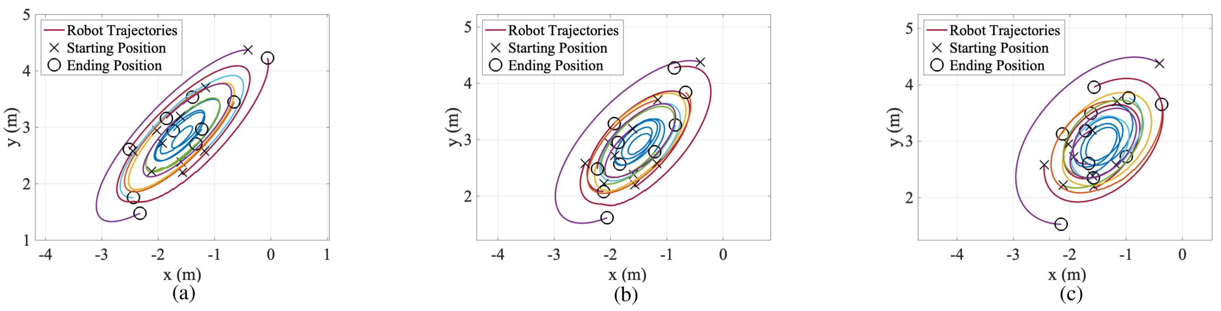

Fig. 10. Trajectories for elliptical orbit with (a) $\alpha = 0.45,\ \beta = 0.55$; (b) $\alpha = 0.4,\ \beta = 0.6$; and (c) $\alpha = 0.3,\ \beta = 0.7$.

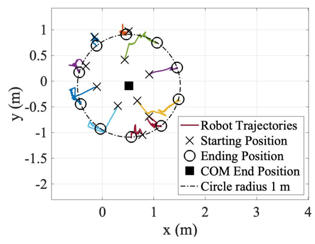

Fig. 11. Uniform ring attraction with $a = 1$ m and $N = 10$ trajectory.

In this hardware trial, 10 robots attract to a ring with a radius of 1 m with an ACR of 50 m and a d_{OA} of 0.25 m. Fig. 11 plots the robot trajectories. The distances between robots converge to the expected chord lengths for uniformly distributed robots, indicating successful uniform ring attraction.

In 100 uniform ring simulations, ten robots were commanded to a circle of radius 1 m. The initial conditions were randomly spaced throughout a 40 m by 40 m area. Each simulation was 1000 s, and the robots used an ACR of 50 m and a d_{OA} of 0.25 m. Then the following parameters were determined at the end of the simulation. The average minimum distance between robots was 0.614 m (SD = 0.002 m). This distribution is significantly different from the minimum distance distribution for the pure ring attraction behavior $[t(99) = -496.64, p < 0.0001]$. This indicates that the dispersion behavior, $f_{\tilde{n}}$, was controlling the spacing of the robots, not the obstacle avoidance function. Next, the mean of the average distance to the CM was calculated as 1.001 m (SD = 0.002 m). The mean of this distribution is statistically different from the mean for pure ring attraction, $[t(99) = -2.610, p = 0.0105]$. The mean of the maximum distances to the CM was 1.008 m (SD = 0.002 m). The mean of the distribution of maximum distances to the CM is significantly higher than that for pure ring attraction $[t(99) = 3.574, p < 0.0001]$. The previous two statistics indicate that the uniform ring attraction works less well at achieving the ring attraction. Since the uniform ring behavior has the addition of another behavior over pure ring attraction, this slight degradation in performance was expected. From a practical standpoint, however, uniform ring attraction still performs the desired attraction to the ring very well.

There are five unique chord lengths for 10 nodes equally spaced on a circle. The maximum percent error of the average of each unique chord length was 0.075%, indicating that the robots equally dispersed along the circle, as claimed. Barnes et al. [25] present a similar behavior of uniformly spacing along a ring, although with a different formulation. Their control uses artificial potential fields and limiting functions to move to the ring and an additional control law to space the robots along it. We achieve similar results with a more straightforward formulation and without limiting functions.

B. Composite Case Study: Simple Flocking

Flocking is a prevalent swarm behavior and has been extensively studied [26]. For RPS, simple flocking breaks down into two behaviors: 1) An attract behavior using CM weighting as in (9) with $\theta_{CM} = 0\,\text{deg}$, and 2) go-to coordinates, an external input. Equation (21) expresses the control input for go-to coordinates in terms of the position of robot i, $\vec{x}_i$, and the desired coordinates $\vec{r}_c$.

$$\vec{u}_{\text{u\_GoTo\_}i} = \vec{r}_c - \vec{x}_i. \tag{21}$$

To demonstrate how the selection of the scaling constants k_{b_CM} and $k_{\text{u\_GoTo}}$ changes the swarm's behavior, we present three versions of flocking: 1) $k_{b\_CM} = 0.1$, 2) $k_{b\_CM} = 0.25$, and 3) $k_{b\_CM} = 0.5$. The robots start at the same random initial conditions for each hardware trial with go-to coordinates of $\vec{r}_c = [\,2 \quad 0.5\,]^T$. Each hardware trial lasts for 120 s, and the robots have an ACR of 10 m and a d_{OA} of 0.25 m. Fig. 12 shows the trajectories for the three experimental trials. As k_{b_CM} increases, the clustering of the robots happens more quickly, impeding the motion toward the go-to coordinates.

Then, 100 simulations of three types of flocking were performed: 1) Medium attraction $k_{b\_CM} = 0.3$, 2) strong attraction $k_{b\_\text{CM}} = 0.5$, and 3) dominant attraction $k_{b\_\text{CM}} = 0.7$. Ten robots started at random initial conditions in a 10 m by 10 m area and used an ACR of 20 m and a d_{OA} of 0.25 m. The simulations were 5000 s long. The target coordinates were [10 m, 5 m]. Two statistics were determined for each simulation. First, the first time the CM of the swarm reaches within half the d_{OA} of the target coordinates was determined. Once the swarm gets close to the target, it can take a while for the CM converges to the target coordinates as robots settle without overall movement of the swarm. Using half the d_{OA} removes the effect of this settling. Second, the first time all robots were within twice the d_{OA} was determined for each simulation. This was chosen because a similar number of robots converged to be within about twice the d_{OA} in the CM case study.

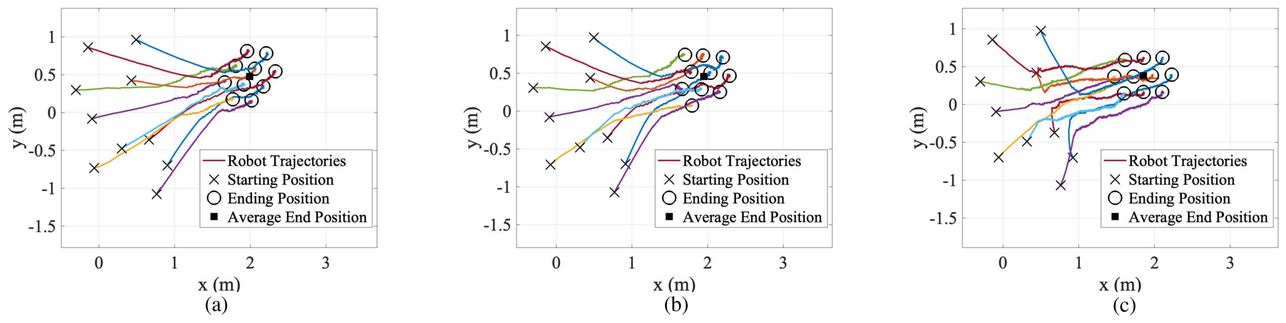

Fig. 12. Trajectory for 10 robots flocking with (a) $k_{b\_CM} = 0.1$, (b) $k_{b\_CM} = 0.25$, and (c) $k_{b\_CM} = 0.5$.

TABLE V
MEAN AND SD FOR CM CONVERGENCE TIME AND GOAL CONVERGENCE TIME FOR THE THREE TYPES OF FLOCKING SIMULATIONS

	CM convergence time		Goal convergence time	
Attraction Type	Mean (s)	SD (s)	Mean (s)	SD (s)
Medium	203.5	25.6	231.6	26.1
Strong	33.8	19.3	730.0	83.9
Dominant	12.1	3.7	2851.7	361.8

TABLE VI
LIBRARY OF BASE BEHAVIORS FROM THIS ARTICLE AND SOURCES IF APPLICABLE

f_b	Reference vector	Reference
9	Points to local CM	[14]
10	Points along average relative position vector	[14]
15	Average of nearest n neighbor positions	[15]
16	Vector is parallel to vector pointing to local CM. The sign is determined by being inside or outside the desired circle.	[25]
17	Sum of the vector pointing to local CM and a skew vector aligning swarm to a specific angle.	[23]
20	Points along vector that minimizes error in inter-robot distances	[13]

The mean and SD for target convergence time and goal convergence time are specified in Table V. The p-value for the one-way ANOVA for goal convergence time is less than 0.0001, $[F(2, 297) = 3146, p < 0.0001]$, and the pairwise p-values for the Tukey–Kramer test are all less than 0.0001. Therefore, as k_{b_cm} increases, the swarm takes longer to reach the goal. The p-value for the one-way ANOVA for CM convergence time is less than 0.0001, $[F(2, 297) = 4190, p < 0.0001]$, and the pairwise p-values for the Tukey–Kramer test are all less than 0.0001. Therefore, as k_{b_cm} increases, the swarm more rapidly converges to the CM. This example demonstrates how different selection of scaling constants changes the motion of the swarm.

VII. Discussion

The presented case studies demonstrate the flexibility of the control architecture. Different weighting functions, f_b, calculate different reference vectors. The robot's motion relative to the reference vector is manipulated by changing θ_b, leading to the different base behaviors. Changing the values of the scaling constants, k_b and k_u, emphasizes certain base behaviors, changing the performance of the composite behavior. Table VI summarizes the base behaviors presented in this article. This library includes multiple versions of stable aggregation, one coverage behavior, one formation maintenance behavior, and several orbiting behaviors. The only main area of RPS behavior that this work does not cover is gradient estimation for source seeking, which will be included in future work.

Finally, let us compare this novel RPS architecture with Mesbahi and Egerstedt's architecture in [16]. Mesbahi and Egerstedt's formulation calculates for a single-behavior weighted sum of relative position vectors using a weighting function. Their weighing functions calculate positive scalars that are a function of the position between robots. Our formulation relaxes these requirements. A weighting function, $f_b(i, j)$, calculates a weighting matrix using the swarm state, robot sensor reading, and similar information. Not only can the weights be negative, but the weighting matrix allows for more complicated linear transformations. The introduction of $K(\theta_b)$ allows for orbit behaviors, which are impossible in Mesbahi and Egerstedt's work. Our composite behaviors design allows for multiple active behaviors, while theirs does not. Therefore, our formulation is more generalized and extensible. Mesbahi and Egerstedt [16] also make several assumptions about the graph's connectivity that, coupled with Lyapunov stability analysis, leads to a stability analysis. Because our work relaxes these requirements, a stability analysis for behaviors is not currently available.

VIII. Conclusion

In the field of RPSs, there lacks a unifying set of control equations even though the overall mathematical equations in each article are very similar. This article's main contribution presents a unifying mathematical control architecture for RPS, which streamlines the development of the control laws for base behaviors. As a second contribution, a series of simulated and hardware-in-the-loop case studies demonstrate the flexibility of these equations. These studies demonstrate how to design composite behaviors swiftly by combining base behaviors. The article's third contribution is establishing a library of verified RPS base behaviors. Future work will develop more complex weighting functions, investigate using other values of θ_b, and demonstrate more complicated composite behaviors. Additional work will also develop better heuristics or automated methods for selecting scaling constants, investigate the stability of the base and composite behaviors, and explore nonlinear or time-varying scaling constants.

REFERENCES

[1] L. Iocchi, D. Nardi, and M. Salerno, *Reactivity and Deliberation: A Survey on Multi-Robot Systems, Ser. Lecture Notes in Computer Science*, vol. 2103, Berlin, Germany: Springer, 2001, pp. 9–32.

[2] Y. Leng, C. Yu, W. Zhang, Y. Zhang, X. He, and W. Zhou, "Task-oriented hierarchical control architecture for swarm robotic system," *Natural Comput.*, vol. 16, no. 4, pp. 579–596, Dec. 2017.

[3] L. Bayindir and E. Sahin, "A review of studies in swarm robotics," *Turkish J. Elect. Eng. Comput. Sci.*, vol. 15, pp. 115–147, 2007.

[4] K. Priya, "A concise chronological reassess of different swarm intelligence methods with multi robotics approach," *ICTAC J. Soft Comput.*, vol. 9, no. 2, pp. 1874–1879, Jan. 2019.

[5] P. Ghassemi and S. Chowdhury, "Informative path planning with local penalization for decentralized and asynchronous swarm robotic search," in *Proc. Int. Symp. Multi-Robot Multi-Agent Syst.*, 2019, pp. 188–194.

[6] I. Navarro and F. Matía, "An introduction to swarm robotics," *ISRN Robot.*, vol. 2013, 2013, Art. no. 608164. doi: 10.5402/2013/608164.

[7] T. Li, H.-S. Shin, and A. Tsourdos, "Efficient decentralized task allocation for UAV swarms in multi-target surveillance missions," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2019, pp. 61–68.

[8] M. A. Lewkowicz, R. Agarwal, and N. Chakraborty, "Distributed algorithm for selecting leaders for supervisory robotic swarm control," in *Proc. Int. Symp. Multi-Robot Multi-Agent Syst.*, 2019, pp. 112–118.

[9] S. Avrahami and N. Agmon, "The robotic swarm contamination problem," in *Proc. Int. Symp. Multi-Robot Multi-Agent Syst.*, Aug. 2019, pp. 119–125.

[10] L. Brinón-Arranz, A. Seuret, and C. Canudas-de Wit, "Collaborative estimation of gradient direction by a formation of UAVs under communication constraints," *Proc. IEEE Conf. Decis. Control*, 2011, pp. 5583–5588.

[11] M. S. Couceiro, R. P. Rocha, and N. M. F. Ferreira, "A novel multi-robot exploration approach based on particle swarm optimization algorithms," in *Proc. IEEE Int. Symp. Safety, Secur., Rescue Robot.*, 2011, pp. 327–332.

[12] H. Yamaguchi, "A cooperative hunting behavior by mobile robot troops," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1998, pp. 3204–3209.

[13] V. Gazi and K. Passino, "A class of attraction/repulsion functions for stable swarm aggregations," *Int. J. Control*, vol. 77, pp. 2842–2847 , 2003.

[14] L. Hou, F. Fan, J. Fu, and J. Wang, "Time-varying algorithm for swarm robotics," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 217–222, Jan. 2018.

[15] J. D. McLurkin, "Stupid robot tricks: A behavior-based distributed algorithm library for programming swarms of robots," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., 2004.

[16] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*, stu - student edition ed. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[17] S. Li, R. Kong, and Y. Guo, "Cooperative distributed source seeking by multiple robots: Algorithms and experiments," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 6, pp. 1810–1820, Dec. 2014.

[18] E. Rosero and H. Werner, "Cooperative source seeking via gradient estimation and formation control (part 1)," in *Proc. UKACC Int. Conf. Control*, 2014, pp. 628–633.

[19] M. Ji and M. Egerstedt, "Distributed formation control while preserving connectedness," in *Proc. 45th IEEE Conf. Decis. Control*, 2006, pp. 5962–5967.

[20] P. Glotfelter, I. Buckley, and M. Egerstedt, "Hybrid nonsmooth barrier functions with applications to provably safe and composable collision avoidance for robotic systems," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1303–1310, Apr. 2019.

[21] S. Hart, N. Metzger, M. Reese, R. McDonald, M. Neumann, and C. Kitts, "Robotics simulator for development and verification of swarm behaviors," in *Proc. Int. Des. Eng. Techn. Conf. Comput. Inf. Eng. Conf.*, 2019. doi: 10.1115/DETC2019-97622.

[22] S. Tomer et al., "A low-cost indoor testbed for multirobot adaptive navigation research," in *Proc. IEEE Aerosp. Conf.*, 2018, pp. 1–12.

[23] D. Roy, A. Chowdhury, M. Maitra, and S. Bhattacharya, "Geometric region-based swarm robotics path planning in an unknown occluded environment," *IEEE Trans. Ind. Electron.*, vol. 68, no. 7, pp. 6053–6063, Jul. 2021.

[24] V. Gazi, B. Fidan, L. Marques, and R. Ordonez, "Robot swarms: Dynamics and control," in *Mobile Robots for Dynamic Environments*, New York, NY, USA: ASME Press, 2015. doi: 10.1115/1.860526_ch4.

[25] L. Barnes, M. Fields, and K. Valavanis, "Swarm formation control utilizing elliptical surfaces and limiting functions," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 39, no. 6, pp. 1434–1445, Dec. 2009.

[26] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Aug. 1987.

Shae T. Hart (Graduate Student Member, IEEE) received the B.S. degree in physics, in 2016, from Carnegie Mellon University, Pittsburgh, PA, USA, and the M.S. degree in mechanical engineering, in 2019, from Santa Clara University, Santa Clara, CA, USA, where she is currently working toward the Ph.D. degree in mechanical engineering.

She was a Teaching Assistant with the Department of Mechanical Engineering and a Research Assistant with Robotic Systems Laboratory, Santa Clara University.

Christopher A. Kitts (Senior Member, IEEE) received the B.S.E. degree in mechanical and aerospace engineering from Princeton University, Princeton, NJ, USA, in 1987, the M.P.A. degree in international and defense policy from the University of Colorado, Boulder, CO, USA, in 1996, and the M.S. degree in aerospace engineering and the Ph.D. degree in mechanical engineering from Stanford University, Stanford, CA, USA, in 2006 and 1992, respectively.

He was a Satellite Constellation Mission Controller with U.S. Air Force and a Computer Scientist with Computational Sciences Division, NASA Ames Research Center. He is currently a Professor of mechanical engineering, the Director of the Robotic Systems Laboratory, and the Associate Dean of the Research and Interdisciplinary Programs with the School of Engineering, Santa Clara University, Santa Clara, CA, USA.

Dr. Kitts is a fellow of the American Society of Mechanical Engineers.