Analysis of 3D Position Control for a Multi-Agent System of Self-Propelled Agents Steered by a Shared, Global Control Input

Li Huang, Julien Leclerc, and Aaron T. Becker

***Abstract*— This paper investigates strategies for 3D multi-agent position control using a shared control input and self-propelled agents. The only control inputs allowed are rotation commands that rotate all agents by the same rotation matrix. In the 2D case, only two degrees-of-freedom (DOF) in position are controllable. We review controllability results in 2D, and then show that interesting things happen in 3D. We provide control laws for steering up to nine DOF in position, which can be mapped in various ways, including to control the x, y, z position of three agents, make four agents meet, or reduce the spread of n agents.**

I. INTRODUCTION

Interest in swarm robotics in the control and robotics communities has increased. Compared to highly intelligent and advanced robots, each agent in a swarm robot system is inexpensive, easy to manufacture, and suitable to deploy in large populations [1]. At macro scales, swarm robots such as micro aerial vehicles and 2D autonomous ground vehicles have great potential to be applied to sensing, mapping, localization, surveillance, rescue, etc; at micro scale, agents such as ferromagnetic microrobots, magnetotactic bacteria, and catalytic Janus particles are researched for target drug delivery, non-invasive surgery, micro assembly, etc.

In this paper, we consider a swarm of simple robots with limited communication capability such that agents are commanded by a central system, and agent-to-agent information exchange is not applicable. The swarm system might consist of hundreds or thousands of agents, but each agent receives a copy of the same control input. For example, one potential application is steering catalytic Janus particles with uniform magnetic fields. These particles are self-propelled by a reaction between platinum on the particle and the liquid the particle swims in. The particles are also magnetic. The external magnetic field applies a torque that aligns the magnetic dipole of the particle with the external field. The results in [2] demonstrated steering an individual particle to a desired location and demonstrated that multiple particles could be made to move in different directions when under the control of a single magnetic field.

Consider a swarm with n agents in free space, with their headings randomly initialized. The agents are modeled as self-propelled points that move at a constant velocity along a thrust vector that is fixed in their local coordinate frame. A central system controls agents via a broadcast mechanism. The only control inputs allowed are rotation commands that rotate each agent's local coordinate frame by the same rotation matrix.

Work was supported by National Science Foundation, Grant No. 1619278.

L. Huang, J. Leclerc, and A. Becker are with the Dept. of Electrical and Computer Engineering, University of Houston, Houston, TX 70004, USA lhuang21@uh.edu.

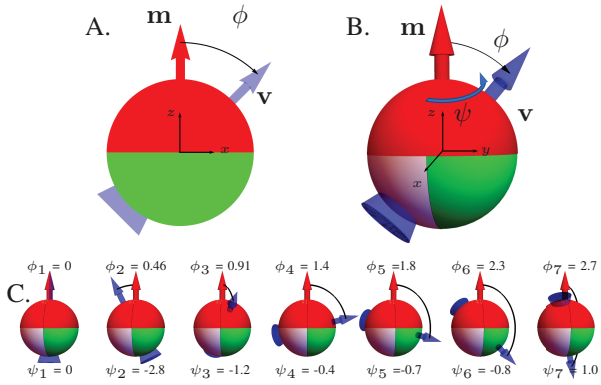


Fig. 1. Schematic of self-propelled agent in 2D (A) and in 3D (B). 2D motion is defined by the offset angle ϕ of the thrust vector (blue) from the local coordinate frame, while the 3D motion is defined by angles ϕ and ψ. (C) Seven agents with different thrust vector orientations. See video at https://youtu.be/sSSQgnmjmJw.

This model is similar to ensemble control systems [3]–[13]. In these problems an ensemble of nearly identical agents that differ only in a set of one or more parameters, are each steered by the same control input. However, the major challenge in this paper is the constrained control input in (28), where $R(t)$ needs to be a rotation matrix.

This paper address the 3D position control problem of a multi-agent system using a shared rotation control input, including (i) with no state perturbations, simultaneously steering up to three agents to arbitrary locations; (ii) with independent rotation perturbations, enabling simultaneously steering many agents ($n > 3$) to arbitrary locations. While boundary interactions could be another method to steer large populations of agents [14], this paper focuses on simple agents in free-space that are affected only by their thrust, local coordinate frames, and the shared control command.

Previous works have addressed control strategies for 3D multi-agent navigation and planning, for example, [15] discussed a distributed control scheme based on Navigation Functions to drive aircraft-like vehicles towards their targets while avoiding colliding into each other or obstacles; [16] proposed an acyclic minimally structural persistent graph based formation control to steer a group of autonomous agents in 3D space; and in [17], a decentralized 3D formation control algorithm was developed for a multi-agent system with unknown dynamics. However, these control schemes rely on independent control of each agent, which is not

applicable within the scope of this paper.

The key results of the this paper include techniques to make n agents follow orbits (section III-B), steering a swarm of agents to arbitrary x, y, z positions, and reducing the variance of the swarm (sections III-D to III-F). Simulation code is available at GitHub [18] and Wolfram Demonstration Projects [19], and the video demo is on Youtube [20].

II. 2D Control of Self-Propelled Agents

We begin our analysis with a 2D version of the broadcast control problem. Figure 1 shows a schematic of a self-propelled point robot in 2D (a disk) and 3D (a sphere).

Bretl [21] and Das et al. [22] have discussed the shared control problem in a 2D plane. Bretl proposed a control law that directs two agents to meet at the same location simultaneously, and with input perturbations. Das demonstrated the possibility of achieving position consensus for large number of agents.

Consider n agents in a 2D xy plane: the origin of each local coordinate frame coincides with the individual center of mass, and let the local coordinate frame be initially aligned with the global coordinate frame. The kinematics of the i^{th} agent is given by

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + R_\theta \mathbf{v}_i(t), \tag{1}$$

with $\mathbf{x}_i(t) \in \mathbb{R}^{2\times 1}$ the position at time t, $\mathbf{v}_i(t) \in \mathbb{R}^{2\times 1}$ the thrust vector, and R_θ the shared control command that rotates the agent along its local z-axis:

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2}$$

Given two self-propelled agents with any initial positions $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{2\times 1}$, and thrust vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^{2\times 1}$ ($\mathbf{v}_1 \nparallel \mathbf{v}_2$), they can meet at only one unique location $\hat{\mathbf{x}}_{1,2}$,

$$\hat{\mathbf{x}}_{1,2} = \mathbf{x}_2 + \frac{1}{2}\begin{bmatrix} 1 & \tan\frac{2}{\phi_2-\phi_1} \\ \tan\frac{-2}{\phi_2-\phi_1} & 1 \end{bmatrix}(\mathbf{x}_1 - \mathbf{x}_2) \tag{3}$$

where $\phi_i = \arctan(\mathbf{v}_i)$. For $n > 2$ such agents in a 2D workspace, there are $n(n-1)/2$ potential collision locations, as shown in Fig. 2.

In general, a sequence of commands for shared control cannot make large number of agents ($n > 2$) to meet simultaneously except for some special initial conditions. If thrust vector perturbations are permitted, Das shows the possibility of point convergence for all agents in [22].

III. 3D Control of Self-Propelled Agents

The effect of any command sequence for 2D position control (without perturbations) of self-propelled agents can be replicated by three commands: an initial rotation along the local z-axis, a translation, and a final rotation along the local z-axis. Therefore, the configuration space for these agents' position is two-dimensional, no matter how many agents are used. This is because the ending position of each agent is the result of the same rigid body transformation modulo an initial rotation. Interestingly, in 3D-space multiple rotation and translation operations can be concatenated to control more than three degrees of freedom.

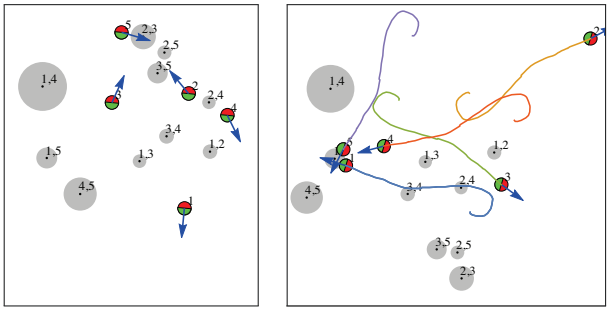


Fig. 2. The configuration space for self-propelled agents in 2D is in $\mathbb{R}^2$. The potential collision locations are drawn as gray disks determined by the initial agent location and orientation. The radius of the collision disk is a function of agent radius and difference in orientation. Right panel shows a set of control inputs that brings agents 1 and 5 into collision. Online demonstration available at [19].

This work reports on simultaneous 3D position control for multiple self-propelled agents. Without loss of any generality, each agent is initialized with different thrust vectors. The origin of each agent's coordinate frame is at the agent's position, and the x, y, and z axes are aligned with the global coordinate frame. The agents implement shared rotation commands based on their local coordinate frames. For simplification, the actuation time for rotations is negligible compared to the time of translation.

A. Steer one self-propelled agent

To deliver a single self-propelled agent from its initial position $\mathbf{x} \in \mathbb{R}^{3\times 1}$ to a goal location $\hat{\mathbf{x}}$, the agent must be rotated so that the thrust vector points toward $\hat{\mathbf{x}}$. Let $\mathbf{v}$ be the initial thrust vector defined in the global coordinate frame. The desired thrust vector can be described as

$$\hat{\mathbf{v}} = \frac{\hat{\mathbf{x}} - \mathbf{x}}{\|\hat{\mathbf{x}} - \mathbf{x}\|_2}. \tag{4}$$

First, identify the normal vector $\mathbf{k}$ of a plane containing both $\mathbf{v}$ and $\hat{\mathbf{v}}$.

$$\mathbf{k} = \mathbf{v} \times \hat{\mathbf{v}}. \tag{5}$$

Next, rotate θ about $\mathbf{k}$ to align the thrust vector with $\hat{\mathbf{v}}$, where

$$\theta = \arccos(\mathbf{v}, \hat{\mathbf{v}}). \tag{6}$$

B. Station-keeping (orbits) with multiple agents

The following sections provide control laws for steering 3D agents to goal positions. A preliminary challenge is to *keep* multiple agents at given locations. The solution in 2D is to revolve about the local z-axis at a constant rate, where the z-axis is perpendicular to the motion plane. The orbital radius is the thrust velocity divided by the angular frequency: $r = |\mathbf{v}|/\omega$. The faster we revolve, the tighter the orbit. In three dimensions this technique no longer works.

The position change of the ith agent under a shared command that rotates $\theta(t)$ radians about an axis k is given by $\int_0^t R_{k,\theta(\tau)}\mathbf{v}_i\, d\tau$. It is easy to show that a rotation about the x-axis does not, in general, return all agents to the initial

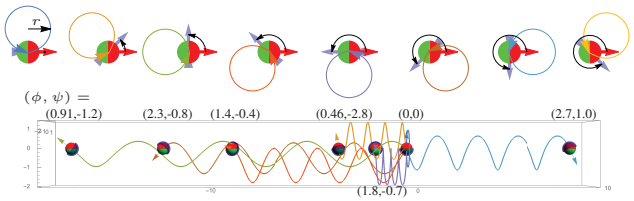


Fig. 3. (Top) In 2D, revolving about the local z-axis results in circular orbits for self-propelled agents with orbit radius $r = |\mathbf{v}|/\omega$. This is not generally true in 3D (Bottom). Paths of the seven agents shown in Fig. 1C when revolving four times around the local x-axis. Single axis rotation does not return the agents to the origin.

location. As shown in Fig. 3, rotating θ radians at 1 radian per second about the local x-axis moves the agents to

$$[\theta v_x, v_z(\cos\theta - 1) + v_y \sin\theta, v_y(1 - \cos\theta) + v_z \sin\theta]^\top.$$

After one revolution, each agent has been displaced by $[2\pi v_x, 0, 0]$, and any agent with a non-zero v_x has followed a helical trajectory. Self-propelled agents with positive v_x will have moved in the positive x direction, and the others in the negative x.

A solution that returns all self-propelled agents to their initial positions is given by eight revolutions that toggle between revolving about the local x and y-axes. All revolutions proceed at a constant angular velocity and, as in 2D, the maximum deviation scales linearly with the inverse of the angular velocity:

$$R_{x,\pi}R_{y,-\pi}R_{x,-\pi}R_{y,\pi}R_{x,-\pi}R_{y,\pi}R_{x,\pi}R_{y,-\pi} = I_3. \quad (7)$$

All x, y, z subscripts for any rotation R refer to the current local x, y, z-axes in the following sections. The positions for an agent initially at the origin after each rotation are

$$\dot{\theta}^{-1}\left(\begin{bmatrix} \pi v_x \\ -2v_z \\ 2v_y \end{bmatrix}, \begin{bmatrix} \pi v_x + 2v_z \\ -\pi v_y - 2v_z \\ 2v_x + 2v_y \end{bmatrix}, \begin{bmatrix} 2v_z \\ -\pi v_y \\ 2v_x + 4v_y \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 4v_x + 4v_y \end{bmatrix}, \begin{bmatrix} \pi v_x \\ 2v_z \\ 4v_x + 2v_y \end{bmatrix}, \begin{bmatrix} \pi v_x - 2v_z \\ -\pi v_y + 2v_z \\ 2v_x + 2v_y \end{bmatrix}, \begin{bmatrix} -2v_z \\ -\pi v_y \\ 2v_x \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) \quad (8)$$

Trajectories of this input sequence are shown in Fig. 4.

C. Simultaneous Position Control

For a system of $n \geq 2$ arbitrary self-propelled agents in a 3D free-space, there are $6n$ DOF: the position vectors $\mathbf{x}_i \in \mathbb{R}^{3\times 1}$ and the thrust vectors $\mathbf{v}_i \in \mathbb{R}^{3\times 1}$. This work provides both open-loop and closed-loop algorithms to control up to nine DOF in positions with no state perturbations.

For ease of exposition, the shared controller only uses R_{x,θ_x} and R_{y,θ_y} as rotation primitives to control the thrust vector heading, because any 3D rotations $R_{k,\theta}$ about a local axis k can be represented by a series of rotations about the current local x, y, and x axis:

$$R_{k,\theta} = R_{x,\theta_1}R_{y,\theta_2}R_{x,\theta_3}. \quad (9)$$

If the shared command steers agents to rotate θ_x about the current local x-axis, the corresponding rotation matrix and

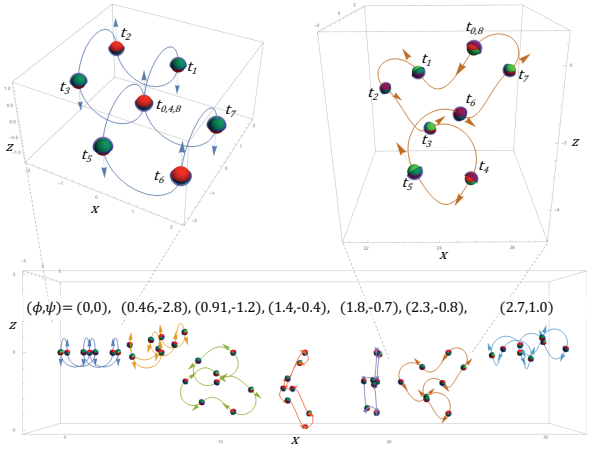


Fig. 4. Periodic orbits of the seven agents shown in Fig. 1C, under the open-loop input (7). The agents and current thrust arrows are redrawn at $t_i = k\pi$.

resultant thrust orientations in the global coordinate frame are

$$R_{x,\theta_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}, \quad (10)$$

$$\hat{\mathbf{v}}_i = R_{x,\theta_x}\mathbf{v}_i, \quad (11)$$

where the subscripts x, θ_x in R_{x,θ_x} denote the local x-axis and rotation angle. If the next command is to rotates θ_y about the current local y-axis, the corresponding rotation matrix and thrust orientations in the global coordinate frame are

$$R_{y,\theta_y} = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \quad (12)$$

$$\hat{\mathbf{v}}_i = R_{x,\theta_x}R_{y,\theta_y}\mathbf{v}_i. \quad (13)$$

These rotations are performed about the current local x or y-axis instead of the fixed global coordinate frame, and both (11) and (13) give a thrust vector $\hat{\mathbf{v}}_i$ defined in the global frame.

Each rotation is followed by a translation motion. To simplify the derivation, the time required for any rotation is assumed to be negligible compared to the translation actuation time.

In 3D space multiple self-propelled agents cannot be driven to arbitrary goal locations $\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_n \in \mathbb{R}^{3\times 1}$ with one rotation and translation. However, concatenating the rotation operations (10) and (12) followed by translations enables controlling additional DOF.

This can be written in matrix form as

$$\begin{bmatrix} \Delta\mathbf{x_1} \\ \vdots \\ \Delta\mathbf{x_n} \end{bmatrix} = \begin{bmatrix} R_1\mathbf{v}_1 & R_2\mathbf{v}_1 & \cdots & R_N\mathbf{v}_1 \\ R_1\mathbf{v}_2 & R_2\mathbf{v}_2 & \cdots & R_N\mathbf{v}_2 \\ \vdots & \vdots & \vdots & \vdots \\ R_1\mathbf{v}_n & R_2\mathbf{v}_n & \cdots & R_N\mathbf{v}_n \end{bmatrix} \mathbf{t} = R_v\mathbf{t} \quad (14)$$

with $\Delta\mathbf{x}_i = \hat{\mathbf{x}}_i - \mathbf{x}_i$, $R_j = R_{x,\theta_1}R_{y,\theta_2}\cdots R_{x,\theta_j}$, $R_{j+1} = R_jR_{y,\theta_{j+1}}$, and $\mathbf{t} = [t_1, t_2, \cdots, t_N]^\top$. If all R_j and $\mathbf{t}$ are

unknown, solving (14) directly is computationally intensive. Instead, randomly generating the N angles is computationally cheap and works well in simulation, though methods shown later can outperform this.

Given the N rotation angles, $\Delta\mathbf{x}_i$, and $\mathbf{v}_i$, where $i \in \{1, 2, \cdots, n\}$, the goal is to choose $\mathbf{t}$ for (14) that minimizes $\|\mathbf{t}\|_1$ with $t_j \geq 0,\ \forall j \in [1, N]$.

Any three independent vectors in $\mathbb{R}^{3\times 1}$ forms a basis for the 3D space. Without loss of generality, assume that $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ in R_v (14) are independent and all $\mathbf{v}_i$ are unique. So any $\mathbf{v}_i$ of R_v can be expressed as a linear combination of $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$

$$\mathbf{v}_i = \lambda_{i,1}\mathbf{v}_1 + \lambda_{i,2}\mathbf{v}_2 + \lambda_{i,3}\mathbf{v}_3, \tag{15}$$

where $\lambda_{i,1}, \lambda_{i,2}, \lambda_{i,3} \in \mathbb{R}$. Recall that row $3i-2$ to row $3i$ in R_v has the form

$$R_v(3i-2:3i,*) = [R_1\mathbf{v}_i \quad R_2\mathbf{v}_i \quad \cdots \quad R_N\mathbf{v}_i], \tag{16}$$

where $\mathbf{v}_i \in \mathbb{R}^{3\times 1}$. Substitute $\mathbf{v}_i$ with (15),

$$\begin{array}{rcccccc} R_v(3i-2:3i,*) & = & \lambda_{i,1}[R_1\mathbf{v}_1 & R_2\mathbf{v}_1 & \cdots & R_N\mathbf{v}_1] \\ & + & \lambda_{i,2}[R_1\mathbf{v}_2 & R_2\mathbf{v}_2 & \cdots & R_N\mathbf{v}_2] \\ & + & \lambda_{i,3}[R_1\mathbf{v}_3 & R_2\mathbf{v}_3 & \cdots & R_N\mathbf{v}_3] \end{array}$$

which indicates that any row of R_v is a linear combination of its first nine rows. So the row rank of R_v is at most nine.

Recall that column j in R_v has the form

$$R_v(*,j) = \begin{bmatrix} R_j\mathbf{v}_1 \\ R_j\mathbf{v}_2 \\ \vdots \\ R_j\mathbf{v}_n \end{bmatrix} \tag{17}$$

Assuming there are at most k independent columns of R_v, without loss of generality, let $R_v(*, 1:k)$ be these columns. Hence, $R_v(*,j)$ can be expressed as

$$R_v(*,j) = l_{j,1}R_v(*,1) + l_{j,2}R_v(*,2) + \cdots + l_{j,k}R_v(*,k)$$

$$\begin{bmatrix} R_j\mathbf{v}_1 \\ R_j\mathbf{v}_2 \\ \vdots \\ R_j\mathbf{v}_n \end{bmatrix} = \begin{bmatrix} (l_{j,1}R_1 + l_{j,2}R_2 + \cdots + l_{j,k}R_k)\mathbf{v}_1 \\ (l_{j,1}R_1 + l_{j,2}R_2 + \cdots + l_{j,k}R_k)\mathbf{v}_2 \\ \vdots \\ (l_{j,1}R_1 + l_{j,2}R_2 + \cdots + l_{j,k}R_k)\mathbf{v}_n \end{bmatrix}$$

That is, $\forall i \in \{1, 2, \ldots, n\}$

$$(l_{j,1}R_1 + l_{j,2}R_2 + \cdots + l_{j,k}R_k - R_j)\mathbf{v}_i = 0_{3\times 1} \tag{18}$$

Therefore

$$l_{j,1}R_1 + l_{j,2}R_2 + \cdots + l_{j,k}R_k = R_j \tag{19}$$

We write R_k as

$$R_k = \begin{bmatrix} r_{k,11} & r_{k,12} & r_{k,13} \\ r_{k,21} & r_{k,22} & r_{k,23} \\ r_{k,31} & r_{k,32} & r_{k,33} \end{bmatrix}, \tag{20}$$

and let $\mathbf{r}_k = [r_{k,11}, r_{k,12}, r_{k,13}, \cdots, r_{k,31}, r_{k,32}, r_{k,33}]^\top$, $\mathbf{l}_j = [l_{j,1}, l_{j,2}, \cdots, l_{j,k}]^\top$. Then (19) can be written as

$$[\ \mathbf{r}_1 \quad \mathbf{r}_2 \quad \cdots \quad \mathbf{r}_k\ ]\mathbf{l}_j = \mathbf{r}_j. \tag{21}$$

To have a unique solution to $\mathbf{l}_j$, $[\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_k]$ needs to be invertible, that is, $k = 9$. We can conclude that the column rank of R_v is at most nine. Because the maximum rank of R_v is nine, $n \leq 3$ self-propelled agents can be steered to arbitrary goal locations, and only if these agents have linearly independent initial thrust vectors.

Only nine of $3n$ DOF can be manipulated if $n > 3$. A special case is to make four agents to meet simultaneously. There is a unique position $\hat{\mathbf{x}}$ where four such agents can meet.

$$\begin{aligned} \hat{\mathbf{x}} &= \mathbf{x}_1 + \sum_{i=1}^{N} R_i\mathbf{v}_1 t_i = \mathbf{x}_2 + \sum_{i=1}^{N} R_i\mathbf{v}_2 t_i \\ &= \mathbf{x}_3 + \sum_{i=1}^{N} R_i\mathbf{v}_3 t_i = \mathbf{x}_4 + \sum_{i=1}^{N} R_i\mathbf{v}_4 t_i \end{aligned} \tag{22}$$

Let $R_t = \sum_{i=1}^{N} R_i t_i$,

$$\begin{aligned} \mathbf{x}_1 - \mathbf{x}_2 &= R_t(\mathbf{v}_2 - \mathbf{v}_1) \\ \mathbf{x}_2 - \mathbf{x}_3 &= R_t(\mathbf{v}_3 - \mathbf{v}_2) \\ \mathbf{x}_3 - \mathbf{x}_4 &= R_t(\mathbf{v}_4 - \mathbf{v}_3) \end{aligned} \tag{23}$$

Flatten R_t as a vector $\mathbf{r}_t$. Rewrite the right side of (23) as

$$\begin{bmatrix} \mathbf{x}_1 - \mathbf{x}_2 \\ \mathbf{x}_2 - \mathbf{x}_3 \\ \mathbf{x}_3 - \mathbf{x}_4 \end{bmatrix} = \begin{bmatrix} R_\Lambda(\mathbf{v}_2, \mathbf{v}_1) \\ R_\Lambda(\mathbf{v}_3, \mathbf{v}_2) \\ R_\Lambda(\mathbf{v}_4, \mathbf{v}_3) \end{bmatrix} \mathbf{r}_t = R_{\Lambda\mathbf{v}}\mathbf{r}_t. \tag{24}$$

where

$$R_\Lambda(\mathbf{v}_i, \mathbf{v}_j) = \begin{bmatrix} \mathbf{v}_i^\top - \mathbf{v}_j^\top & 0_{1\times 3} & 0_{1\times 3} \\ 0_{1\times 3} & \mathbf{v}_i^\top - \mathbf{v}_j^\top & 0_{1\times 3} \\ 0_{1\times 3} & 0_{1\times 3} & \mathbf{v}_i^\top - \mathbf{v}_j^\top \end{bmatrix} \tag{25}$$

So $\mathbf{r}_t$ (i.e., R_t) has a unique solution if and only if $R_{\Lambda\mathbf{v}}$ is invertible. Hence we can derive the meeting point

$$\hat{\mathbf{x}} = \mathbf{x}_1 + R_t\mathbf{v}_1. \tag{26}$$

D. Open-loop control using linear programming

One way to solve (14) is via linear programming. The objective is to find a vector $\mathbf{t}$ such that the total control time $\|\mathbf{t}\|_1$ is minimized subject to (14) with $n = 3$.

$$\begin{aligned} &\min \sum_{j=1}^{N} t_j, \quad \text{s. t. } t_j \geq 0, j = 1, ..., N \text{ and} \\ &\begin{bmatrix} R_1\mathbf{v}_1 & R_2\mathbf{v}_1 & \ldots & R_N\mathbf{v}_1 \\ R_1\mathbf{v}_2 & R_2\mathbf{v}_2 & \ldots & R_N\mathbf{v}_2 \\ R_1\mathbf{v}_3 & R_2\mathbf{v}_3 & \ldots & R_N\mathbf{v}_3 \end{bmatrix} \mathbf{t} = \begin{bmatrix} \Delta\mathbf{x}_1 \\ \Delta\mathbf{x}_2 \\ \Delta\mathbf{x}_3 \end{bmatrix} \end{aligned} \tag{27}$$

where the initial thrust vectors are linearly independent. Because the t_j must be nonnegative, the number of rotation matrices (N) should be much greater than nine, as shown in Fig. 7a.

E. Feedback control

Let the state space representation of n such agents be

$$[\ \dot{\mathbf{x}}_1 \quad \cdots \quad \dot{\mathbf{x}}_n\ ] = R(t)\,[\ \mathbf{v}_1 \quad \cdots \quad \mathbf{v}_n\ ]. \tag{28}$$

where $\mathbf{x}_i$ denotes position of the i^{th} agents, $\mathbf{v}_i$ describes the initial thrust vector, and $R(t)$ is the shared control input. We

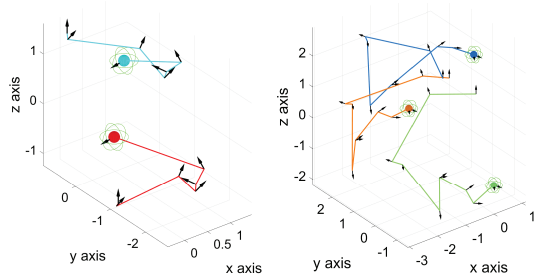


Fig. 5. Open-loop control simulations using linear programming. The left figure has two self-propelled agents, and the right figure has three agents. The goal locations are indicated by green orbits. In each figure, all spheres move the same total distance and reach the goal location at the same time. A colored line describes the trajectory of each sphere. Black arrows indicate the local coordinate frame z-axis for the subsequent move.

construct an objective function based on the sum of squared Euclidean distance error

$$V(t) = \frac{1}{2}\sum_{i=1}^{n} \left(\hat{\mathbf{x}}_i - \mathbf{x}_i(t)\right)^\top \left(\hat{\mathbf{x}}_i - \mathbf{x}_i(t)\right), \tag{29}$$

where $\hat{\mathbf{x}}_i$ is the i^{th} goal location. We hold $R(t)$ constant for time interval $[t_k, t_k + \tau_k)$, such that τ_k minimizes the convex objective function. Consider the first-order necessary condition:

$$\begin{aligned} \dot{V}(t_k + \tau_k) = \tfrac{d}{d\tau_k} V(t_k + \tau_k) = 0, \\ -\sum_{i=1}^{n} (\hat{\mathbf{x}}_i - \mathbf{x}_i(t_k) - \tau_k R(t_k)\mathbf{v}_i))^\top R(t_k)\mathbf{v}_i = 0, \end{aligned} \tag{30}$$

$$\text{and thus} \qquad \tau_k = \frac{\sum_{i=1}^{n} (\hat{\mathbf{x}}_i - \mathbf{x}_i(t_k))^\top R(t_k)\mathbf{v}_i}{\sum_{i=1}^{n} \|R(t_k)\mathbf{v}_i\|_2^2}. \tag{31}$$

Note that the time interval τ_k must be non-negative, so if there exists an $R(t_k)$ such that $\tau_k > 0$, then $\int_{t_k}^{t_k+\tau_k} \dot{V}(t)dt < 0$, and the total distance error decreases monotonically with time; otherwise, the objective function has reached the minimum. Thus the system kinematics can also be written as

$$\mathbf{x}_i(t_{k+1}) = \mathbf{x}_i(t_k) + R(t_k)\mathbf{v}_i\tau_k. \tag{32}$$

F. Greedy optimal control

We discussed the optimal actuation time to minimize the objective function $V(t)$ with a constant rotation matrix $R(t)$ within time interval $[t, t + \tau_k)$. By carefully choosing this $R(t)$, the objective function follows the steepest gradient during each time interval:

$$R(t_k) = \underset{\alpha_k,\beta_k,\gamma_k}{\operatorname{argmin}} \dot{V}(t_k), \tag{33}$$

with $R(t_k)$ a function of rotation angles α_k, β_k, and γ_k

$$R(t_k) = R(t_{k-1})R_{x,\alpha_k}R_{y,\beta_k}R_{x,\gamma_k}. \tag{34}$$

This indicates that at t_k, each agent implements the shared rotation control $R(t_k)$ which is equivalent to rotating α_k about the current local x-axis, then β_k about the current y-axis, and finally rotating γ_k about the current x-axis.

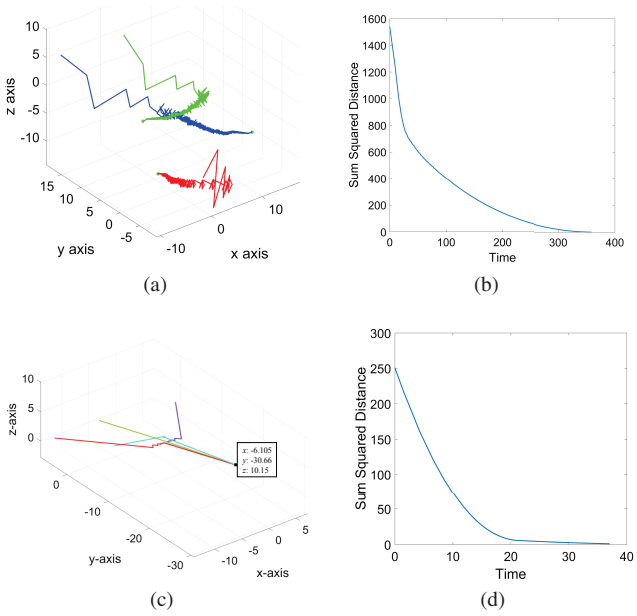


Fig. 6. Simulations of self-propelled agents using greedy optimal control. All agents reach the goal location at the same time. Colored lines describe the trajectories of each agent. (a) Three agents reach arbitrary goal locations simultaneously. (b) The corresponding objective function plot with simulation time. (c) Four agents move towards their mean position till they meet. (d) The corresponding objective function plot with simulation time.

G. Control with state perturbations

Under ideal conditions, agent motions are assumed to be perfectly implemented without errors. Up to four agents can be steered to meet simultaneously with a shared control input. In general, for $n > 4$ agents, the objective function reaches a local minimum, and the shared control cannot bring all agents to their target locations simultaneously.

In practice, agent rotations and translations may not be precise due to process noise and measurement noise. Inspired by the 2D results in [22], it is possible to escape local minima in 3D, if we relax previous assumptions and assume agents are subjected to independent random disturbances on their thrust vectors after each translation. In the following we show that there always exist disturbances that enable steering the agents closer to their goal. In simulation, we show that randomly perturbing each agent's thrust vector enables convergence.

Section III-E shows that $\tau_k \leq 0$ if the objective function reaches a local minimum, and according to 31,

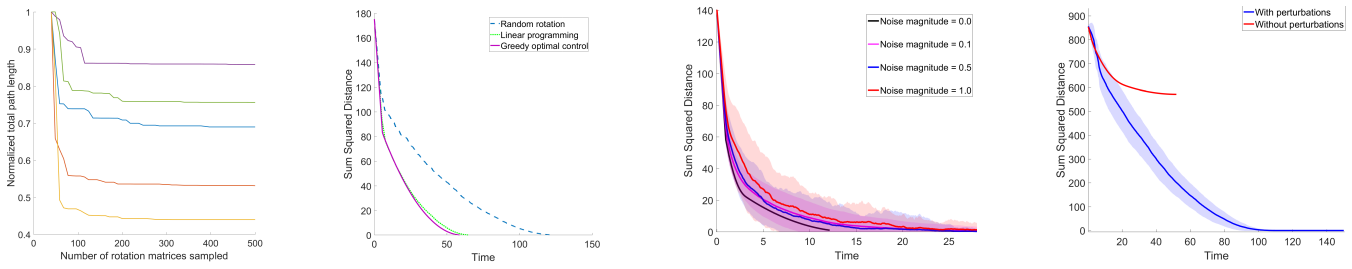
$$\sum_{i=1}^{n} (\hat{\mathbf{x}}_i - \mathbf{x}_i(t_k))^\top \mathbf{v}_i(t_k) \leq 0. \tag{35}$$

where $R(t_k)\mathbf{v}_i$ is replaced by $\mathbf{v}_i(t_k)$. Let $\boldsymbol{\delta}_i(t_k) \in \mathbb{R}^3$ be a perturbation of the i^{th} thrust vector, such that the perturbed thrust vector $\mathbf{v}'_i(t_k) \parallel (\hat{\mathbf{x}}_i - \mathbf{x}_i(t_k))^\top$, where

$$\mathbf{v}'_i(t_k) = \frac{\mathbf{v}_i(t_k) + \boldsymbol{\delta}_i(t_k)}{\|\mathbf{v}_i(t_k) + \boldsymbol{\delta}_i(t_k)\|_2} \|\mathbf{v}_i(t_k)\|_2. \tag{36}$$

Therefore

$$\sum_{i=1}^{n} (\hat{\mathbf{x}}_i - \mathbf{x}_i(t_k))^\top \mathbf{v}'_i(t_k) > 0 \tag{37}$$



(a) (b) (c) (d)

Fig. 7. Representative parameter optimization for controlling three self-propelled agents in 3D. (a) In open-loop control with linear programming, path lengths decrease monotonically with the number of rotation matrices N. (b) Performance comparison of feedback control using random rotation matrices, linear programming and greedy optimal control. (c) Greedy optimal control with or without position noises. The solid lines (with noises) are the average results of 50 simulations, and the shaded areas represent the corresponding standard deviation. (d) Control of ten agents given the same initial conditions, with or without thrust vector perturbations. The blue line (with perturbations) is the average results of 50 simulations, and the shaded area represents the corresponding standard deviation. The greedy optimal control without perturbations gets trapped in a local minimum (red line).

except for the case that the total position error is 0.

This example proves the existence of a perturbation that makes the objective function monotonically decreasing, even if the system is in a local minimum. In this paper's simulations, the thrust vector perturbations are sampled from a zero-mean normal distribution.

IV. CONTROL SIMULATIONS OF SELF-PROPELLED SPHERES

In this section, we show the controllability for self-propelled agents in 3D free-space, including steering up to three spheres to arbitrary goals with a shared open-loop control (linear programming) and a shared closed-loop control, and control of four agents to meet. With perturbations on thrust vectors, we show the capability of driving ten agents to the origin.

In Figure 5, open-loop control with linear programming is implemented with up to three self-propelled agents and drives them to arbitrary goals. According to section III-D, N angles are randomly generated to provide a large number of rotation matrix candidates, but linear programming selects N_k among N ($N_k \ll N$) matrices to apply actuation and steer the spheres to goal locations. In the simulation, a rotation matrix could be generated by $R(t) = R_{x,\alpha}R_{y,\beta}R_{x,\gamma}$, where $\alpha, \beta, \gamma \in [0, 2\pi]$. N is set to 200, and usually $N_k = 9$. Figure 7a shows that N can affect the performance of linear programming: if N is small, the provided paths to goals might be much longer than the optimal solution. The total path length decreases monotonically with N. The total path length has little change when $N \geq 200$.

Section III-E introduced a closed-loop control law such that a rotation matrix $R(t)$ is held constant for each actuation time interval τ_k. Aptly choosing each τ_k, the cost function decreases monotonically along the trajectory. In simulation, $R(t)$ is generated with the following methods: (i) random angle generation, (ii) using the N_k rotation matrices selected by linear programming, and (iii) minimizing $\dot{V}(t)$ in (30) with respect to $\alpha_k, \beta_k, \gamma_k$. Figure 6 shows the trajectories of greedy optimal control with 3 and 4 agents using method (iii). Not all $\alpha_k, \beta_k, \gamma_k$ must be non-zero values. For example, we could let $\alpha_k, \beta_k = 0$, or $\alpha_k, \gamma_k = 0$, or $\gamma_k = 0$ for greedy optimal control. For more simulation results, please refer to the repository on GitHub [18].

A performance comparison of the above three methods is shown in Figure 7b, which indicates that methods (ii) and (iii) have competitive performance, while method (i) takes about twice as long to converge.

Section III-C showed we can control up to nine DOF with no state perturbations. Figure 6 gives two examples of controlling up to four self-propelled agents: steering three agents to predefined goal locations, and moving four spheres to their mean positions, as shown in (26).

When thrust vector perturbations are considered, the shared control is capable of moving out of local minima and bringing many agents ($n > 4$) to arbitrary locations, as shown in Figure 7d. In addition, the influence of position noises are compared in Figure 7c.

V. CONCLUSIONS

This paper proved limitations on control for self-propelled agents that all receive the same rotation commands, but extended the existing literature which focused on two dimensional results to show that nine degrees-of-freedom of position can be controlled. In 2D only one agent can be steered to an arbitrary position, and two agents have only one possible meeting point. In 3D up to three agents may be steered to arbitrary positions, and four agents have only one possible meeting point.

There are many avenues for future work. These include optimal control results and analytical solutions to the optimal rotations for the controllers in section III. In particular, we would like to calculate the meeting location for two spheres that requires the shortest control sequence. This problem is trivial in 2D, but potentially hard in 3D.

These controllers have potential insights for real-world systems that are self-propelled and can be steered by the orientation of a global field. The size of micro-scale robots makes it difficult to include onboard computation, so they are often steered by external fields. Examples include steering magnetized single-celled organisms [23]–[26], magnetotactic bacteria [27]–[30] and catalytic Janus particles with magnetic cores [2], [31]–[33].

REFERENCES

[1] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1699–1706. IEEE, 2017.

[2] Sambeeta Das, Edward B Steager, Kathleen J Stebe, and Vijay Kumar. Simultaneous control of spherical microrobots using catalytic and magnetic actuation. In *Manipulation, Automation and Robotics at Small Scales (MARSS), 2017 International Conference on*, pages 1–6. IEEE, 2017.

[3] Jr-Shin Li and Navin Khaneja. Ensemble control of bloch equations. *IEEE Transactions on Automatic Control*, 54(3):528–536, 2009.

[4] Jr-Shin Li. Ensemble control of finite-dimensional time-varying linear systems. *IEEE Transactions on Automatic Control*, 56(2):345–357, 2011.

[5] Jr-Shin Li and Navin Khaneja. Control of inhomogeneous quantum ensembles. *Physical review A*, 73(3):030302, 2006.

[6] Jr-Shin Li and Navin Khaneja. Ensemble control of linear systems. In *Decision and Control, 2007 46th IEEE Conference on*, pages 3768–3773. IEEE, 2007.

[7] Justin Ruths and Shin Li. Optimal ensemble control of open quantum systems with a pseudospectral method. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3008–3013. IEEE, 2010.

[8] Justin Ruths and Jr-Shin Li. Optimal control of inhomogeneous ensembles. *IEEE Transactions on Automatic Control*, 57(8):2021–2032, 2012.

[9] Aaron Becker and Timothy Bretl. Approximate steering of a unicycle under bounded model perturbation using ensemble control. *IEEE Transactions on Robotics*, 28(3):580–591, 2012.

[10] Aaron Becker and Timothy Bretl. Approximate steering of a plate-ball system under bounded model perturbation using ensemble control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5353–5359. IEEE, 2012.

[11] Aaron Becker and Timothy Bretl. Motion planning under bounded uncertainty using ensemble control. In *Robotics: Science and Systems*, 2010.

[12] Aaron Becker, Cem Onyuksel, and Timothy Bretl. Feedback control of many differential-drive robots with uniform control inputs. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2256–2262. IEEE, 2012.

[13] Aaron Becker and James McLurkin. Exact range and bearing control of many differential-drive robots with uniform control inputs. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3338–3343. IEEE, 2013.

[14] Shiva Shahrokhi, Arun Mahadev, and Aaron T Becker. Algorithms for shaping a particle swarm with a shared input by exploiting non-slip wall contacts. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 4304–4311. IEEE, 2017.

[15] Giannis Roussos, Dimos V Dimarogonas, and Kostas J Kyriakopoulos. 3d navigation and collision avoidance for nonholonomic aircraft-like vehicles. *International Journal of Adaptive Control and Signal Processing*, 24(10):900–920, 2010.

[16] Wenbiao Xu, Xuejing Lan, and Yonaii Wang. Distance-based formation control of multi-agent systems moving in 3d space based on an amsp graph. In *2018 37th Chinese Control Conference (CCC)*, pages 6676–6681. IEEE, 2018.

[17] Alexandros Nikou, Christos K Verginis, and Dimos V Dimarogonas. Robust distance-based formation control of multiple rigid bodies with orientation alignment. *IFAC-PapersOnLine*, 50(1):15458–15463, 2017.

[18] Li Huang and Aaron T. Becker. Janus Particle Control Simulation. `https://github.com/RoboticSwarmControl/JanusParticleControl`, 2018.

[19] Aaron T. Becker and Javier Garcia. Steering Multiple Radio Control (RC) Cars with One Joystick. `http://demonstrations.wolfram.com/SteeringMultipleRadioControlRCCarsWithOneJoystick/`, 2018.

[20] Aaron T. Becker and Li Huang. 3D position control for a multi-agent system of self-propelled agents steered by a global control input. `https://youtu.be/sSSQgnmjmJw`, Feb 2019.

[21] Timothy Bretl. Control of many agents using few instructions. In *Robotics: Science and Systems*, 2007.

[22] Kaushik Das and Debasish Ghose. Broadcast control mechanism for positional consensus in multiagent systems. *IEEE Transactions on Control Systems Technology*, 23(5):1807–1826, 2015.

[23] Dal Hyung Kim, U Kei Cheang, László Kőhidai, Doyoung Byun, and Min Jun Kim. Artificial magnetotactic motion control of tetrahymena pyriformis using ferromagnetic nanoparticles: A tool for fabrication of microbiorobots. *Applied Physics Letters*, 97(17):173702, 2010.

[24] Dal Hyung Kim, David Casale, László Kőhidai, and Min Jun Kim. Galvanotactic and phototactic control of tetrahymena pyriformis as a microfluidic workhorse. *Applied Physics Letters*, 94(16):163901, 2009.

[25] Yan Ou, Dal Hyung Kim, Paul Kim, Min Jun Kim, and A Agung Julius. Motion control of magnetized tetrahymena pyriformis cells by a magnetic field with model predictive control. *The International Journal of Robotics Research*, 32(1):129–140, 2013.

[26] Dal Hyung Kim, Paul Seung Soo Kim, Anak Agung Julius, and Min Jun Kim. Three-dimensional control of tetrahymena pyriformis using artificial magnetotaxis. *Applied Physics Letters*, 100(5):053702, 2012.

[27] Dumitru Loghin, Charles Tremblay, Mahmood Mohammadi, and Sylvain Martel. Exploiting the responses of magnetotactic bacteria robotic agents to enhance displacement control and swarm formation for drug delivery platforms. *The International Journal of Robotics Research*, 36(11):1195–1210, 2017.

[28] Sylvain Martel, Mahmood Mohammadi, Ouajdi Felfoul, Zhao Lu, and Pierre Pouponneau. Flagellated magnetotactic bacteria as controlled mri-trackable propulsion and steering systems for medical nanorobots operating in the human microvasculature. *The International journal of robotics research*, 28(4):571–582, 2009.

[29] Ouajdi Felfoul, Mahmood Mohammadi, Samira Taherkhani, Dominic De Lanauze, Yong Zhong Xu, Dumitru Loghin, Sherief Essa, Sylwia Jancik, Daniel Houle, Michel Lafleur, et al. Magneto-aerotactic bacteria deliver drug-containing nanoliposomes to tumour hypoxic regions. *Nature nanotechnology*, 11(11):941, 2016.

[30] Ouajdi Felfoul and Sylvain Martel. Assessment of navigation control strategy for magnetotactic bacteria in microchannel: toward targeting solid tumors. *Biomedical microdevices*, 15(6):1015–1024, 2013.

[31] Larysa Baraban, Denys Makarov, Robert Streubel, Ingolf Monch, Daniel Grimm, Samuel Sanchez, and Oliver G Schmidt. Catalytic janus motors on microfluidic chip: deterministic motion for targeted cargo delivery. *ACS nano*, 6(4):3383–3389, 2012.

[32] Larysa Baraban, Denys Makarov, Oliver G Schmidt, Gianaurelio Cuniberti, Paul Leiderer, and Artur Erbe. Control over janus micromotors by the strength of a magnetic field. *Nanoscale*, 5(4):1332–1336, 2013.

[33] Islam SM Khalil, Veronika Magdanz, Samuel Sanchez, Oliver G Schmidt, and Sarthak Misra. Precise localization and control of catalytic janus micromotors using weak magnetic fields. *International journal of advanced robotic systems*, 12(1):2, 2015.