Decentralized Flocking Control for Dynamic Object Tracking: Implementation and Analysis

May 12, 2025

Abstract

Flocking refers to a collective behavior that emerges when multiple agents exchange information and work toward a common goal. Flocking algorithms, which have diverse applications across various domains, have been developed, with a notable example being the work of Olfati-Saber. Building upon this foundation, La and Sheng implemented a flocking-based algorithm for tracking dynamic agents using a mobile sensor network. This paper evaluates the performance of La and Sheng's approach in terms of flocking behavior, obstacle avoidance, tracking accuracy, and robustness to adversarial agents. Furthermore, this work explores the algorithm's limitations and potential directions for future improvement.

I. Introduction

Flocking refers to a collective behavior that emerges when multiple agents exchange information and work toward a common goal. [1] This phenomenon is prevalent in nature and can be seen in examples such as bird herds, fish schools, and ant colonies. In the study of artificial flocks, Reynolds introduced three rules to simulate flocking behavior in computer graphics: I) Collision Avoidance: avoid collisions with nearby flockmates. II) Velocity Matching: attempt to match velocity with nearby flockmates. III) Flock Centering: attempt to stay close to nearby flockmates. [2]

Building upon Reynolds' principles, Olfati-Saber developed a theoretical framework for flocking algorithms, addressing flocking of distributed system in free space and in environments with multiple obstacles. [1] Such algorithms have many applications, including automated manufacturing, coordination of unmanned vehicle systems, and distributed sensing. Specifically, in the field of mobile sensor networks, La and Sheng built upon Olfati-Saber's framework and propose a scalable algorithm designed to track a moving target in the presence of obstacles. [3]

This paper aims to evaluate the performance of La and Sheng's algorithm in terms of flocking behavior, obstacle avoidance, tracking precision, and robustness against adversarial agents. It also seeks to identify the algorithm's limitations and explore potential directions for future improvements. Section II provides a review of the algorithms proposed in [1] and [3]. Section III presents the simulation results, while Section IV offers a detailed evaluation of the algorithm. Conclusions are drawn in Section V, and code and video resources are provided in Section VI.

II. Methodology

II. A. Flocking Algorithm

A graph G is defined as a pair (V, E) consisting of a set of vertices $V = \{1, 2, \ldots, n\}$ and a set of edges $E \subseteq \{(i, j) : i, j \in V, j \neq i\}$. Here, each node corresponds to a member within the flock, and each edge represents a communication channel between two members. [3]

For each $i \in V$, let $q_i,\ p_i \in \mathbb{R}^m$ (e.g., $m = 2, 3$) denote the position and velocity of node i, respectively. The set of neighbors of agent i is defined as

$$N_i = \{j \in V : \|q_j - q_i\| \leq r,\ j \neq i\}$$

where r is the interaction range between two agents, with dynamics described by $\dot{q}_i = p_i,\ \dot{p}_i = u_i$.

The control law for flocking of multiple agents with obstacle avoidance proposed by Olfati-Saber consists of three components [1]

$$u_i = f_i^\alpha + f_i^\gamma + f_i^\beta \tag{1}$$

The first component in (1) is defined as

$$f_i^\alpha = c_1^\alpha \sum_{j \in N_i} s_{ij}(q)(q_j - q_i) + c_2^\alpha \sum_{j \in N_i} a_{ij}(q)(p_j - p_i) \tag{2}$$

where c_1^α and c_2^α are positive constants; the terms $s_{ij}(q)$ and $a_{ij}(q)$ represent the stress and adjacency elements, respectively, and are given by

$$s_{ij}(q) = \frac{\phi_\alpha(\|q_j - q_i\|_\sigma)}{1 + \epsilon\|q_j - q_i\|_\sigma}, \quad a_{ij}(q) = \rho_h\left(\frac{\|q_j - q_i\|_\sigma}{r_\alpha}\right), \quad a_{ij}(q) \in [0, 1]$$

and the definition of the activation function $\phi_\alpha(z)$, the bump function ρ_h, and the sigma norm $\|z\|_\sigma$ can be found in [1].

Olfati-Saber derived that $s_{ij}(q)$ is positive when agent i is far from agent j and negative when they are close. Thus, the first summation in (2) enforces repulsion at short ranges and attraction at long ranges, allowing separation and cohesion in position. The second summation in (2) forms a consensus term that facilitates alignment in velocity. Hence, f_i^α captures all three Reynolds' rules.

However, it is shown in [1] that applying only f_i^α to an agent i will lead to a behavior called fragmentation instead of flocking in free space. The missing piece to form flocks turns out to be a common group objective, which is addressed by the second component of the control law as follows.

$$f_i^\gamma = -c_1^\gamma(q_i - q_\gamma) - c_2^\gamma(p_i - p_\gamma) \tag{3}$$

Equation (3), with positive constants $c_1^\gamma,\ c_2^\gamma$, provides a navigational feedback that allows node i to match the position and velocity of a virtual leader with configuration $(q_\gamma,\ p_\gamma)$. The selection of virtual leader is discussed later in section II.B. Nonetheless, once the virtual leader is defined,

whether static or dynamic, applying the control law $u_i = f_i^\alpha + f_i^\gamma$ to each agent i enables flocking in free space.

To achieve flocking in the presence of obstacles, the third component of the control law becomes essential and is defined as follows.

$$f_i^\beta = c_1^\beta \sum_{k \in N_i^\beta} \phi_\beta(\|\hat{q}_{i,k} - q_i\|_\sigma)\hat{n}_{i,k} + c_2^\beta \sum_{k \in N_i^\beta} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \tag{4}$$

The concrete definition and details of (4) are provided in [1]. To interpret the formula, for each node i, a virtual β agent is created for every obstacle. Each β agent is positioned at $\hat{q}_{i,k}$, the closest point on the obstacle to agent i, and is assigned a velocity $\hat{p}_{i,k}$ that steers agent i away from colliding with the obstacle. Similar to (2), the control law then drives the agent to align its velocity with that of the β agent, resulting in motion to traverse around the obstacle, modulated by a weight $b_{i,k}(q)$. For the position component, the control input acts as a repulsive force directed along the unit vector $\hat{n}_{i,k}$ from the obstacle to the agent, with a strength determined by ϕ_β, which increases as the distance decreases. In this way, (4) effectively achieves obstacle avoidance.

II. B. Object Tracking Algorithm

To apply the flocking algorithm to a mobile sensor network for tracking a moving target while avoiding collisions, we revisit the definition of the virtual leader in (3). Suppose there exists a target with position and velocity $(q_{mt},\ p_{mt})$. A natural approach is to set $q_\gamma = q_{mt}$ and $p_\gamma = p_{mt}$, so that the control law provides a feedback mechanism for agents to follow the moving target.

However, as noted in [3], it is more effective for the center of mass (CoM) of the sensor network to track the target. This allows all agents to surround and monitor the target more closely and uniformly. Directly setting $q_\gamma = q_{mt}$ and $p_\gamma = p_{mt}$ often fails to guide the CoM to the target in environments with obstacles. To address this limitation, a simple modification is introduced, and f_i^γ is defined as

$$f_i^\gamma = -c_1^{mt}(q_i - q_{mt}) - c_2^{mt}(p_i - p_{mt}) - c_1^{sc}(\bar{q} - q_{mt}) - c_2^{sc}(\bar{p} - p_{mt}) \tag{5}$$

where $\bar{q}$ and $\bar{p}$ denote the average position and velocity of all agents in the network, respectively.

Equation (5), however, requires global information where an agent need access to the states of all other agents. This limits scalability in large sensor networks. To address this, a decentralized version of the control law is proposed. Instead of computing the global CoM, each agent computes a local center of mass using only its neighbors' information. [3]

$$f_i^\gamma = -c_1^{mt}(q_i - q_{mt}) - c_2^{mt}(p_i - p_{mt}) - c_1^{mc}(\overline{q}_{(i+N_i^\alpha)} - q_{mt}) - c_2^{mc}(\overline{p}_{(i+N_i^\alpha)} - p_{mt}) \tag{6}$$

where $\overline{q}_{(i+N_i^\alpha)}$ of an agent i is the average position of q_i and its neighbors' positions and $\overline{p}_{(i+N_i^\alpha)}$ denotes the corresponding average velocity.

III. Simulation Results

To implement and evaluate the algorithm, simulation experiments were conducted using Python. The simulation involves 50 agents initially placed at rest with positions uniformly distributed within a $[0, 75] \times [0, 75]$ square region. The object tracking algorithm is applied to all agents using a discrete time step of 0.02s. The control gains are set as follows: $c_1^{\alpha} = 5$, $c_1^{mt} = 15$, $c_1^{mc} = 10$, $c_1^{\beta} = 300$, and $c_2^{v} = 2\sqrt{c_1^{v}}$ for $v \in \{\alpha, mt, mc, \beta\}$. The interaction range is set to 9, and all other parameters are defined as in [1].

The first experiment is conducted to track a moving target in free space. The target begins at position $(20, 50)$ and moves at a constant speed of 10 m/s along the x-axis for the duration of the experiment. Fig. 1 captures the simulation results at $t = 0$s, $t = 0.5$s, and $t = 2$s. (Movie 1 in the Resources Section) The red star in the figure denotes the position of the target, and the flock members are represented by the blue dots. It is shown that the agents form flocking with its CoM aligned with the target.

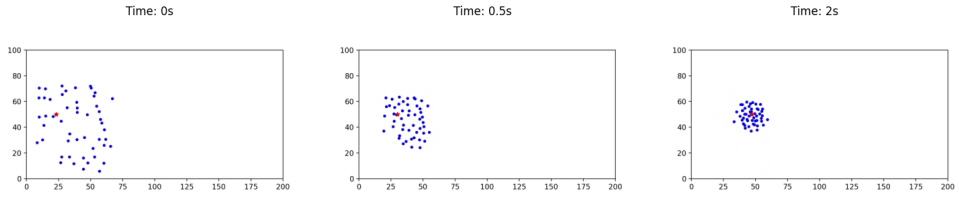

Figure 1: Tracking in Free Space

The second experiment examines target tracking in the presence of an obstacle. The same moving object is tracked as it encounters a circular obstacle centered at $(100,\ 50)$ with a radius of 10. The agents perform a split-and-rejoin maneuver to navigate around the obstacle, as illustrated in Fig. 2 and in Movie 2. Here, the obstacle is represented by a circle with its boundary indicated by black dashed lines. The green dots represent the β agents associated with the obstacle.

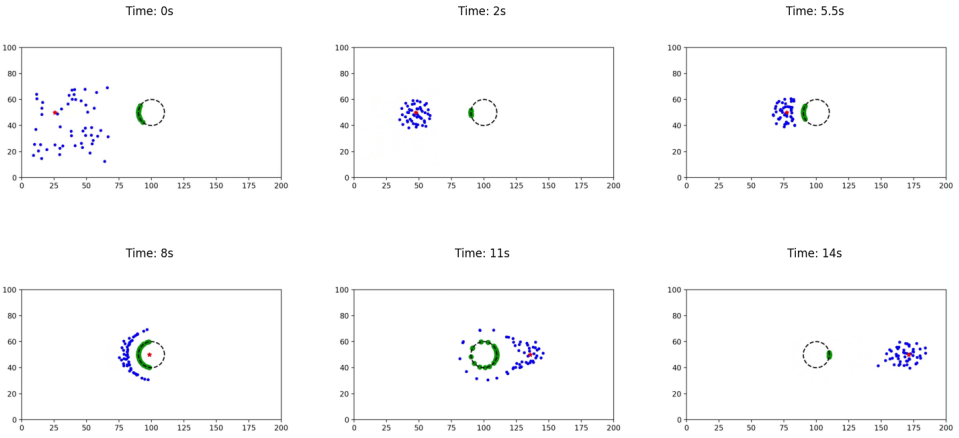

Figure 2: Tracking with Circular Obstacle

The third experiment demonstrates target tracking while avoiding multiple obstacles. Three circu-

lar obstacles are placed at coordinates (100, 50) with radius 10, (75, 60) with radius 3, and (70, 35) with radius 5. The simulation results are presented in Movie 4 and illustrated in the appendix.

IV. Analysis

IV. A. Performance

IV. A.1. Flocking Verification

To evaluate the flocking behavior of the agents, four metrics are computed as proposed in [1]:

1. Deviation Energy: $E(q) = \frac{1}{(|E(q)|+1)} \sum_{i=1}^{n} \sum_{j \in N_i} (\|q_j - q_i\| - d)^2/d^2$ measures how much the current configuration deviates from the flocking formation, where $\|q_j - q_i\| = d \quad \forall j \in N_i(q)$. This value remains small during flocking.

2. Velocity Mismatch: $K(v) = (1/2) \sum_i \|v_i\|^2$ quantifies the differences in agent velocities. It approaches zero as all agents align their motion with the target.

3. Cohesion Radius: $R(t) = \max_{i \in \mathcal{V}} \|q_i(t) - q_c(t)\|$, where q_c is the average position of all agents. A small value indicates that the agents are closely clustered.

4. Relative Connectivity: $C(t) = \frac{1}{n-1} \text{rank}(L(q(t))) \in [0, 1]$, where $L(q) = \{a_{ij}\}$ is the adjacency matrix. This value equals 1 when the graph is connected.

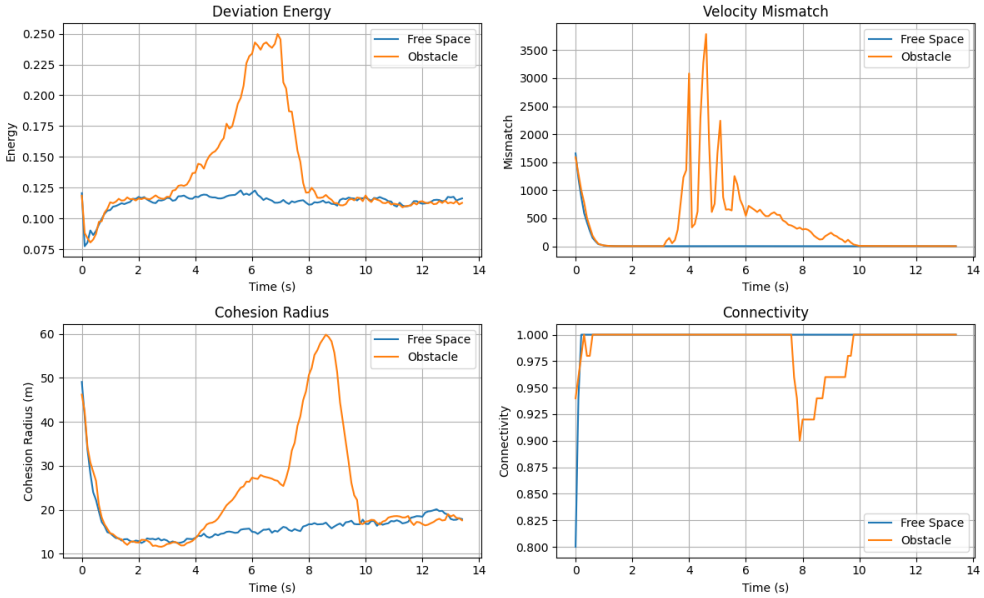

Figure 3: Flocking Verification

Fig. 3 presents the computed metrics for the first two experiments: the one in free space and the other in the presence of a circular obstacle. In the free-space scenario, the metrics align with the results reported in [1]. Meanwhile, when the obstacle is involved (from $t = 4\,\text{s}$ to $t = 11\,\text{s}$

in the figure), the agents temporarily prioritize collision avoidance over maintaining the flock, as indicated by the changes in the metrics. After successfully navigating around the obstacle, they return to the flocking formation.

IV. A.2. Tracking Precision

To quantify the performance in tracking a moving target, the mean squared error (MSE) between the object and the center of mass (CoM) of the flock is computed. The results are shown in Fig. 4, along with the trajectory plots for both experiments. An increase in MSE is observed when the agents encounter the obstacle at $t = 4\,\text{s}$. Comparing this with the trajectory plot, it suggests a time lag where the CoM follows the target along the same path, but with a delay. Once the obstacle is passed, the CoM closely and promptly tracks the target again.

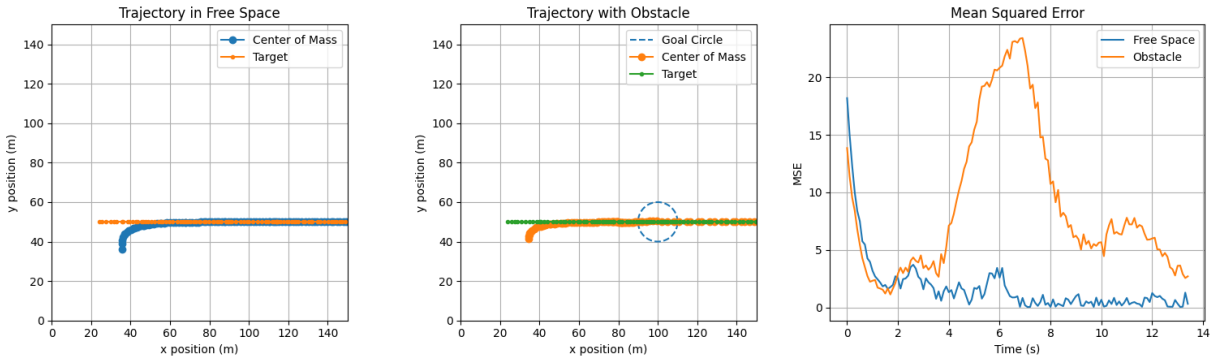

Figure 4: Tracking Precision

IV. A.3. Obstacle Avoidance

To evaluate the obstacle avoidance capability of the algorithm, the number of collisions between agents and the obstacle is recorded. Across 10 randomized trials, where agents are initially positioned at different locations within the $[0, 75] \times [0, 75]$ box, all agents successfully navigated around the obstacle without a single collision. This suggests that, with a properly chosen set of control gains, the algorithm is capable of guiding the flock to track the target while avoiding obstacles effectively.

However, selecting appropriate control gains $(c_1^\alpha, c_1^{mt}, c_1^{mc}, c_1^\beta, c_2^\alpha, c_2^{mt}, c_2^{mc}, c_2^\beta)$ can be subtle. The constraints $c_1^\alpha < c_1^{mt}, c_1^{mc} < c_1^\beta$ and $c_2^v = 2\sqrt{c_1^v}$ for $v \in \alpha, mt, mc, \beta$ reduce the tuning parameter space to four dimensions. Nonetheless, during the implementation phase of this work, several gain combinations were tested, and many of them failed to produce the desired outcome. The main challenge lies in balancing obstacle avoidance with target tracking. When the obstacle avoidance gain is too low, the agents tend to ignore the obstacle and collide with it to chase the target. On the other hand, if the avoidance gain is too high, the agents become overly cautious and refrain from moving forward, until the target has moved far enough for the tracking term to dominate and ignore avoiding the obstacle.

In this work, to tune the control gains, we fixed the tracking and flocking gains to small values and gradually increased the obstacle avoidance gain until collisions were eliminated.

IV. A.4. Balance of Flocking and Tracking

Through experimentation, we observed that when the ratio of tracking gain to flocking gain exceeds 10, oscillatory behavior emerges. This is because a high tracking gain pulls all agents toward the target, causing them to gather closely. As a result, the separation term then pushes the agents apart. Once separated, the cohesion and tracking terms pull them back together, creating a cycle of oscillation. Thus, a careful choice of tracking and flocking gains is essential to achieve a balance between target tracking and flocking behavior. In this work, the tracking gain is three times larger than the flocking gain.

IV. B. Robustness

To evaluate the robustness of the algorithm, two adversarial scenarios are tested. In the first scenario, 5 out of 50 agents ignore the control law in (1) and remain stationary. In the second, the same amount of agents apply random control inputs sampled from a uniform distribution over $[-150, 150]$. For both cases, metrics including deviation energy, velocity mismatch, and connectivity are recorded to assess the flocking behavior. MSE between the global CoM and the target is computed to evaluate tracking precision. The results are compared with the base case (no adversaries) and shown in Fig. 5.

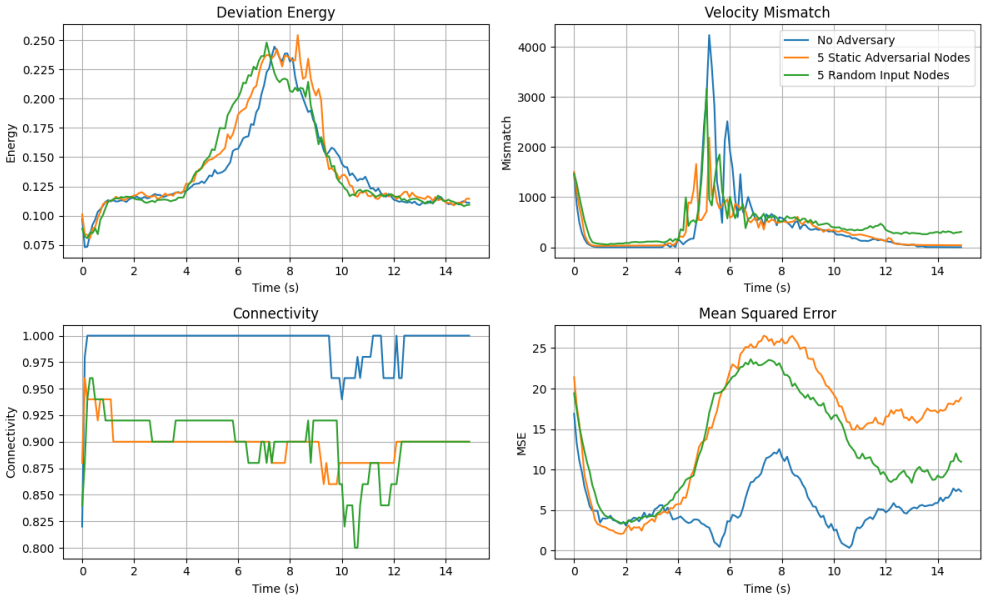

Figure 5: Performance with Adversarial Agents

In the figure, the deviation energy is slightly higher in the adversarial scenarios when the obstacle is

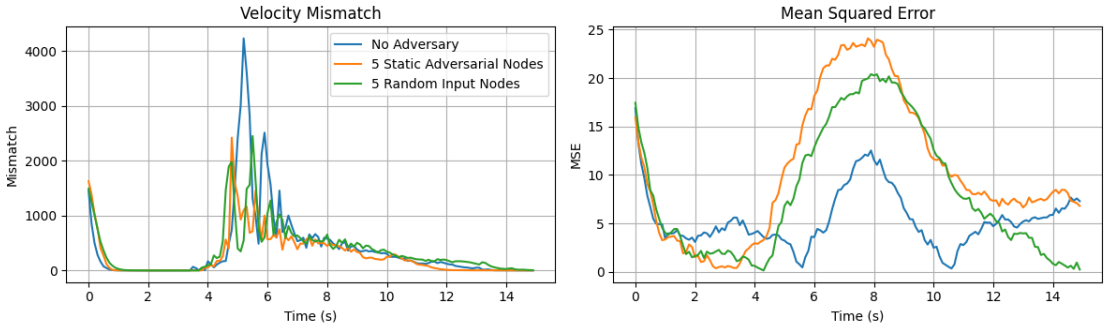

Figure 6: Performance of Non-Adversarial Agents

encountered but eventually converges to a value similar to that of the base case. This is because deviation energy only accounts for distances between neighboring agents, and over time, adversarial nodes tend to be excluded from most local neighborhoods. Velocity mismatch, however, increases under adversarial conditions due to the inconsistent velocities introduced by the adversarial agents. Connectivity drops and converges to 0.9, which aligns with expectations where 10% of the nodes act adversarially, and the remaining 90% still maintain connected. The MSE increases, as the adversarial nodes shift the global center of mass.

Given that the remaining flock members maintain separation and cohesion in position and network connectivity, as indicated by deviation energy and connectivity observations. We now examine velocity mismatch and MSE while excluding the adversarial agents. This allows us to verify whether the normal agents maintain consistent velocities and accurately track the target via the center of mass.

As shown in Fig. 6, the velocity mismatch converges to zero, indicating that the non-adversarial nodes match the target velocity. Meanwhile, the MSE remains high when the obstacle is encountered but eventually converges to a value comparable to that of the base case. This behavior is expected because the control law is based only on local center of mass computations, making it robust to the presence of adversarial agents.

Thus, when adversarial agents are in presence, the non-adversarial members can still perform flocking and track the target with the center of mass.

IV. C. Limitations & Extensions

IV. C.1. Agent Modeling

In [1] and [3], flock members are modeled as point particles, whereas in real-world applications, mobile sensors or robots are rigid bodies with physical dimensions. However, this limitation can be addressed with a simple modification. The cohesion radius, which represents the ideal distance between agents, can be adjusted by tuning the parameters of the bump function and the action function in (2). By doing so, the separation between agents can be increased to account for their

physical sizes, ensuring safe operation without collisions.

IV. C.2. Moving Obstacle

The obstacle in the original algorithm is assumed to be static, and moving obstacles are not explicitly discussed in the paper. However, since the virtual β agents are recalculated at every discrete time step, changes in the obstacle's position are inherently addressed by the algorithm. This allows the method to adapt to dynamic environments. A working example is demonstrated in Fig. 8 and Movie 3, where the obstacle moves in the x-direction at a constant speed of 5 m/s.

IV. C.3. Obstacle Shapes

The algorithm in [1] and [3] is limited to circular and hyperplane obstacles, which restricts its application. For example, consider a rectangular object with vertices at (50, 40), (120, 40), 120, 60), (50, 60). A natural adaption is to find the smallest possible circle to enclose obstacle. However, it is shown in Fig. 7 that the resulting circular obstacle completely blocks the path to reach the target position. This suggests that for non-circular obstacles, simply identifying the smallest obstacle is insufficient. Therefore, the algorithm needs to inherently handle obstacles of various shapes. Fortunately, to overcome this limitation, an extension to the original algorithm is proposed in [4], which allows obstacles with arbitrary shapes.

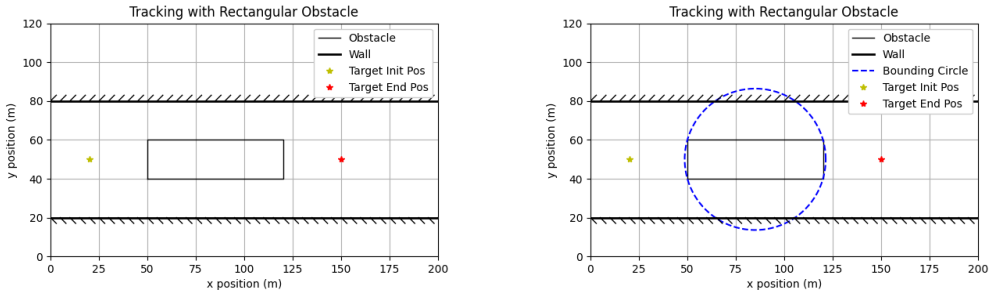

Figure 7: Tracking with Rectangular Obstacle

V. Conclusions

In this work, performance and robustness of the algorithm proposed in [3] is evaluated. In free space, the algorithm enables agents to form flocking and accurately track a moving target. When obstacles are introduced, the agents temporarily prioritize obstacle avoidance over maintaining formation, but resume proper flocking and tracking behavior once the obstacle is surpassed. In scenarios involving adversarial agents such as static nodes or agents with random control inputs, the remaining agents still maintain effective flocking and tracking performance. Additionally, limitations related to agent modeling, moving obstacles, and obstacle shape constraints, and potential solutions to address these challenges are discussed.

VI. Resources

The project code uses Penn MEAM 6240 HW skeleton code as a template and is available at GitHub. The demo videos (Movie 1 through 4) are available at YouTube via following the links M1, M2, M3, M4.

VII. References

[1] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[2] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 25–34, 1987.

[3] H. M. La and W. Sheng, "Flocking control of a mobile sensor network to track and observe a moving target," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3129–3134, IEEE, 2009.

[4] J. Li, W. Zhang, H. Su, and Y. Yang, "Flocking of partially-informed multi-agent systems avoiding obstacles with arbitrary shape," *Autonomous Agents and Multi-Agent Systems*, vol. 29, pp. 943–972, 2015.

Appendix

Simulation Video Frame sampling

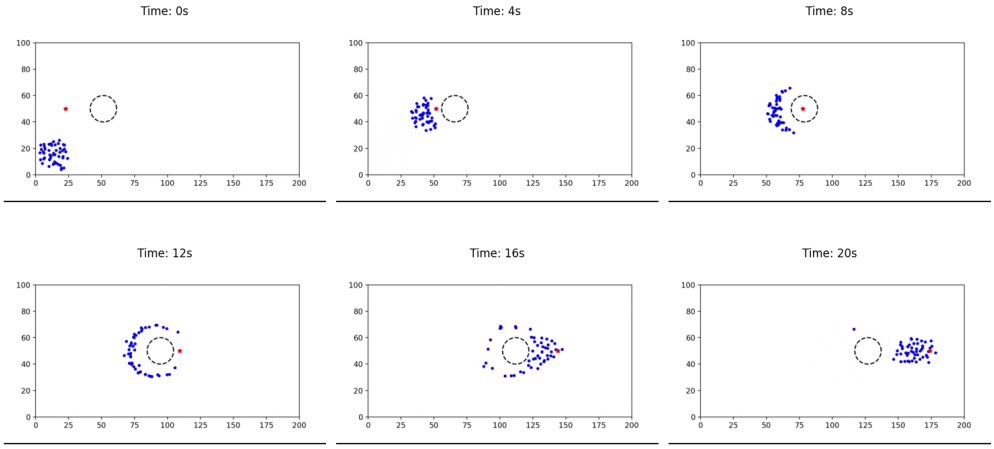

Figure 8: Tracking with Moving Obstacle

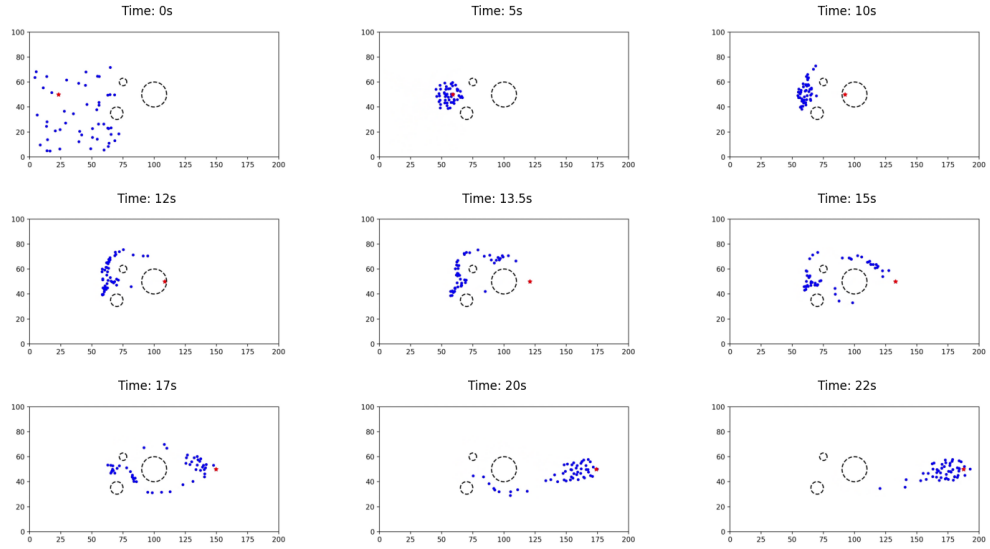

Figure 9: Tracking with Multiple Obstacles

Experiments and Results

Exp	Obstacle Type	Settings	Key Observations
#1	None	Target @ (20, 50) w/ 10m/s in x-dir	Quickly form flocking and track the target accurately
#2	Circular Obstacle @ (100, 50) with radius 10	Target @ (20, 50) w/ 10m/s in x-dir	Prioritizes obstacle avoidance over maintaining formation, then returns to flocking after bypass. CoM tracks the target following the same path, but with time delay, in presence of obstacle. Achieves collision-free
#3	Circular Obstacle @ (100, 50) with radius 10	Target @ (20, 50) w/ 10m/s in x-dir. 5 adversarial agents	In adversarial cases, non-adversarial agents can still perform flocking and tracking
#4	Circular Obstacle @ (100, 50) w/ r 10, @ (75, 60) w/ r 3, @ (70, 35) w/ r 5	Target @ (20, 50) w/ 10m/s in x-dir	Able to track the target while avoiding multiple obstacles.
#5	Circular Obstacle @ (100, 50) with radius 10 w/ 5m/s in x-dir	Target @ (20, 50) w/ 10m/s in x-dir	Able to track the target while avoiding moving obstacle.

Table 1: Summary of Experiment Settings and Key Results