2022 International Conference on Unmanned Aircraft Systems (ICUAS)
Dubrovnik, Croatia. June 21-24, 2022

Target Capturing in an Ellipsoidal Region with a Swarm of Quadrotor UAVs

M. Alp Merzi, Veysel Gazi, Giuseppe Fedele, Luigi D'Alfonso, and Antonio Bono

2022 International Conference on Unmanned Aircraft Systems (ICUAS) | 978-1-6654-0593-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICUAS54217.2022.9836166

***Abstract*— In a recent paper [1], we developed a target capturing strategy for swarms composed of double integrator agents. In this paper, we build on the results in [1] and focus on the implementation of the methodology for a swarm composed of quadrotor UAVs. We utilize both MATLAB and ROS environments under realistic conditions for that purpose. In ROS, we utilize Ultra-Violet LED blinkers based Ultra-Violet Direction And Ranging (UVDAR) system for relative localization. The sensing and localization errors serve as realistic errors for the assumptions in [1] for the uncertainties of the velocity and acceleration of the target. We verify in a realistic scenario that the agents converge to a user-defined ellipsoidal ring around the target despite the uncertainties and the more realistic agent dynamics.**

I. Introduction

Interest in Unmanned Aerial Vehicles (UAVs) and swarms of UAVs has constantly been increasing in the past decade. The collective dynamics of swarm systems are not only a fascinating natural phenomenon by themselves but also have various engineering applications. Potential applications include search and rescue operations, surveillance/reconnaissance, exploration, and defence, to name a few. Tracking and capturing/enclosing a target is also a significant problem that may arise in some of the mentioned applications [2]. The objective of target capturing by a swarm of agents can be stated as to achieve a formation enclosing or surrounding a specific area or an object (a possibly moving target) by using local information [3]. One difficulty with the capturing problem could be that the agents may have access to only imperfect information about the target. In [1], a target capturing strategy for swarms composed of double integrator agents was developed assuming uncertainties on the target velocity and acceleration. This paper further extends and implements this work on a swarm of quadrotor UAVs. We perform simulations in MATLAB and Robot Operating System (ROS) environments, utilizing generic UAV models and realistic assumptions.

Potential functions have been utilized extensively for developing swarm coordination methodologies [4]. They have also been used in studies on target capturing [5]–[8]. In [5] and [6] a 2D space is considered and potential functions methods are developed so that the agents surround a given target. The work in [6] utilizes a gradient descent based decentralized method using local information about the target. In [7] more realistic, fully actuated agent dynamics with model uncertainties are considered and capturing/enclosing of a moving target is achieved utilizing potential functions and sliding mode control based strategy. In [8] the results are further extended to swarms composed of non-holonomic agents with model uncertainties. In [9] an energy based model is used for the dynamics of a swarm of quadrotors using Lagrangian approach. Artificial potential energy and the corresponding potential forces are also utilized for modeling the interaction between the quadrotors.

Cyclic pursuit is another approach that has been utilized in the literature for solving the target capturing problem [3], [10]–[12]. In particular, in [11] the problem has been considered in a planar 2D environment whereas in [3] the solution has been extended to the 3D case.

One shortcoming of most of the studies above is that the agents are assumed to have access to exact information about the target without any uncertainty. In many practical scenarios, however, this assumption may be too strong. Usually, in an actual application, there are sensing errors. Moreover, sometimes especially if the target is moving, some or all the agents may lose their connection with the target. In other words, the agents may possess imperfect information about the target's location and/or velocity and acceleration. Furthermore, most studies assume simplistic agent dynamics such as single integrators or double integrators, and implementing the developed algorithms on real agent swarms requires further development. In this paper, we perform target capturing on an ellipsoidal region using quadrotor UAVs and realistic assumptions.

Advances in sensing, processing, and communication technologies, enable the implementation and realization of technologies that were not possible before, including engineering swarms. For instance, the developments in onboard sensing and localization technologies such as Ultra-Violet Direction And Ranging (UVDAR) [13] allow the realization of relative self-localization algorithms without the need for Global Positioning System (GPS) data. In implementations in the ROS environment, we utilize UVDAR for agent sensing and localization, allowing for local interactions and decentralized behaviour of the UAV swarm system.

In this paper, we use the double integrator model in [1] as a reference model for the swarm of quadrotor UAVs. In other words, the trajectories generated by the model in [1] (the same is possible with single integrator models) serve as

M. Alp Merzi is with Marmara University, Faculty of Engineering, Dept. Electrical and Electronics Eng., Istanbul, Turkey, alpmerzi@gmail.com.

V. Gazi is with Yildiz Technical University, Faculty of Electrical and Electronics Engineering, Dept. Control and Automation Eng., Istanbul, Turkey, vgazi@yildiz.edu.tr.

G. Fedele, L. D'Alfonso and A. Bono are with University of Calabria, Dept. of Informatics, Modeling, Electronics and Systems Eng., 87036 Rende, Italy, {giuseppe.fedele, luigi.dalfonso,antonio.bono}@unical.it.

reference trajectories for the quadrotors. Then, considering the specific agent dynamics, the local controllers ensure that the desired capturing behaviour on an ellipsoidal ring is achieved. Both MATLAB and ROS environments are utilized for implementation and verification.

As a capturing region of the target, we utilize an ellipsoidal ring centred at the target. The ring is formed by an inner distancing (from the target) region, which is forbidden by the agents and an outer containment region within which the agents need to converge. Although the ellipse is a convex region, the resultant ring constitutes a simple nonconvex region within which the agents need to converge. In [1], the swarm is assumed to know the target position but to have only estimates of its velocity and acceleration. The estimation errors are assumed to be bounded by known bounds. We utilize the same assumption for our MATLAB implementation, whereas in our ROS implementation, the target's position is measured through UVDAR with some measurement errors. Also, the agents can sense the other agents through UVDAR (with some errors) for collision avoidance purposes.

The remainder of this paper is organized as follows: Section II defines the capturing problem for a swarm of double integrator agents, quadrotor dynamics and control are discussed in Section III, Section IV presents the obtained simulation results in Matlab and ROS. Finally, concluding remarks and future directions are provided in Section V.

Notation

Here we present some of the notations which will be used throughout this paper.

- $\mathcal{C}^k$ denotes the set of functions $f : \mathbb{R} \to \mathbb{R}^d$ which are continuous and have continuous derivatives up to the order k.
- Given $P \in \mathbb{R}^{d\times d}, P = P^T > 0$ and $c \in \mathbb{R}^d$, $\mathcal{E}(P, c)$ denotes an ellipsoidal region centered at c and shaped by P, i.e., $\mathcal{E}(P, c) \triangleq \{x \in \mathbb{R}^d \,|\, (x-c)^T P(x-c) \leq 1\}$.

II. Problem Formulation with Point Mass Agents

In this section, we will briefly discuss the formulation in [1]. Consider a swarm consisting of n agents, whose dynamics evolve in $d \geq 2$ dimensional Euclidean space. For now, let us assume that the dynamics of the agents are based on the point mass (double integrator) equations

$$\begin{aligned} \dot{x}_i(t) &= v_i(t), \\ \dot{v}_i(t) &= \frac{1}{m_i} u_i(t), \quad i = 1, 2, \ldots, n \end{aligned} \tag{1}$$

where $m_i > 0$ denotes the mass, $x_i(t) \in \mathbb{R}^d$ denotes the position, and $v_i(t) \in \mathbb{R}^d$ denotes the velocity of the i'th agent at time $t \geq 0$. The control (force) input is represented by $u_i(t) \in \mathbb{R}^d$. Later in Section III, we will discuss how we can design controllers for quadrotor UAVs with more complicated dynamics so that to achieve the behavior of the swarm composed of the double integrator agents in (1). The objective is that the agents track and capture/enclose a possibly moving target whose position and velocity at time t are denoted by $x_T(t)$ and $v_T(t) = \dot{x}_T(t)$, respectively. We assume that is twice continuously differentiable or basically that $x_T(t) \in \mathcal{C}^2$. Define the position and velocity errors for agent i (with respect to the target) as [1]

$$\begin{aligned} z_i(t) &= x_i(t) - x_T(t) \\ w_i(t) &= v_i(t) - v_T(t), \quad i = 1, 2, \ldots, n. \end{aligned} \tag{2}$$

Under these definitions, the agent's error dynamics become

$$\begin{aligned} \dot{z}_i(t) &= w_i(t) \\ \dot{w}_i(t) &= \frac{1}{m_i} u_i(t) - a_T(t), \quad i = 1, 2, \ldots, n \end{aligned} \tag{3}$$

where $a_T(t) \in \mathbb{R}^d$ denotes the acceleration of the target. Assume that each agent i possesses only an estimate of the target velocity and acceleration, namely $\hat{v}_T^i(t), \hat{a}_T^i(t) \in \mathcal{C}^0$, and that the errors related to these estimates are bounded with known bounds. In other words, assume that

$$\begin{aligned} \left\| v_T(t) - \hat{v}_T^i(t) \right\| &\leq b_v, \\ \left\| a_T(t) - \hat{a}_T^i(t) \right\| &\leq b_a \end{aligned}$$

where the bounds $b_v \geq 0$ and $b_a \geq 0$ are known. As stated in [1], this assumption is reasonable and necessary. The assumption that the exact values are not known is reasonable since, in implementations, there are sensing errors. For example, for the implementation here in the ROS environment, we utilize UVDAR for relative position sensing, in which there are measurement errors. Similar statements can hold if other types of sensors are used. The assumption that the errors are bounded is necessary since otherwise, the considered problem would be infeasible if the errors are unbounded. The assumption that the upper bounds are known can be justified because the maximum sensing errors are usually known for a given sensor.

As stated earlier, it is required that the agents capture/enclose the target. However, the agents need also to keep a distance from the target for collision avoidance (or other reasons). We call the inner region the distancing region, whereas the region in which the agents need to gather is the containment region. Let us consider these regions as ellipsoids. Then, the region where the agents need to gather around the target forms an ellipsoidal ring centred at the target. The problem is formally defined as follows.

Problem: Target Capturing Problem with Uncertain Data (TCPUD) [1]. Given a swarm composed of n agents with the double integrator dynamics expressed in (3), consider a target with known position and uncertain velocity and acceleration. Determine the agent's control law $u_i(t) \in \mathbb{R}^d, i = 1, 2, \ldots, n$, such that for a given shaping matrix $P = P^\top > 0 \in \mathbb{R}^{d\times d}$ the following conditions hold:

- Distancing: all agents remain outside a distancing region defined by the ellipsoid $\mathcal{E}(P, x_T(t))$, i.e., $\forall i = 1, 2, \ldots, n$ and $\forall t > t_0, x_i(t_0) \notin \mathcal{E}(P, x_T(t_0)) \Rightarrow x_i(t) \notin \mathcal{E}(P, x_T(t))$
- Containment: all agents reach and remain inside the containment region defined by the ellipsoid $\mathcal{E}\left(P/c^2, x_T(t)\right)$. The parameter c that defines the width

of this region, must be greater than a lower bound that takes into account the uncertainty on the target velocity and acceleration. In other words, $\exists \underline{c}\,(b_v, b_a) \geq 1$ such that, at steady-state, $x_i(t) \in \mathcal{E}\left(P/c^2, x_T(t)\right), c > \underline{c}\,(b_v, b_a)\, \forall i = 1, 2, \ldots, n$.

A figure representing ellipsoidal rings, containment and distancing regions can be seen in Figure 1.

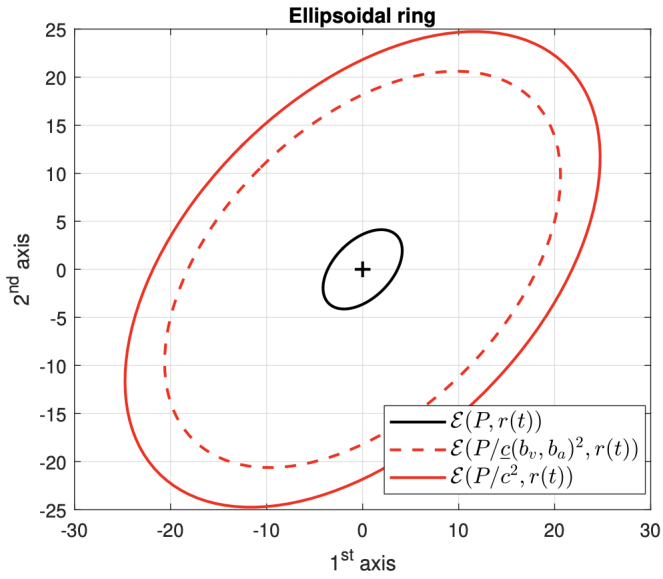

Fig. 1. An example of the ellipsoidal ring where agents have to remain to solve the TCPUD. The ellipse drawn with the dashed line represents the limit on the smallest containment region that the user can choose in this example according to the uncertainty of the target's velocity and acceleration. Figure is taken from [1]

It was shown in [1] that the TCPUD problem defined above for double integrator agents can be solved using the control protocol

$$u_i(t) = -\alpha_i P z_i(t) + \beta_i h_i(t) + \frac{P z_i(t)}{z_i(t)^\top P z_i(t) - 1} \\ -\gamma\left(v_i(t) - \hat{v}_T^i(t)\right) + m_i \hat{a}_T^i(t), \quad i = 1, 2, \ldots, n \tag{4}$$

where $\alpha_i \in \mathbb{R}^+, \beta_i \in \mathbb{R}^+$and $\gamma \in \mathbb{R}^+$ are controller parameters. The term $h_i(t)$ is a swarm interaction term in the form

$$h_i(t) = \sum_{j \in \mathcal{N}_i(t)} g\left(\|z_i(t) - z_j(t)\|\right)\left(z_i(t) - z_j(t)\right) \tag{5}$$

where $\mathcal{N}_i(t)$ represents the set of neighbors of agent i at time instant t. This term is used for collision avoidance, although it is possible to incorporate also attraction between agents. The function $g \in \mathcal{C}^0 : \mathbb{R}^+ \to \mathbb{R}$ is assumed to satisfy the following conditions:

$$\exists\, \sigma \in \mathbb{R}^d \ : \ g(\|y\|)\|y\| \leq \sigma \ \ \forall y, \tag{6}$$

$$g(\|y\|)y = 0_d \quad \text{if} \quad \|y\| \geq r_v \tag{7}$$

for some $r_v > 0$. This means that the inter-agent repulsive force is bounded by σ, becomes zero at some distance r_v, and is zero for all distances greater than r_v. In other words, r_v denotes the inter-agent repulsion range.

The constraints on the controller parameters for achieving the desired behaviour as well as some specific values can be found in [1]. We utilized the same parameters in our MATLAB simulations in this paper. These parameters are $\alpha = 11.57$, $\beta = 1$, $\gamma = 10$, and $P = \mathrm{diag}([1/16, 1/9, 1/4])$. Other parameters are also possible as long as the constraints derived in [1] are satisfied. For the simulation in the ROS environment slightly different values have been used depending on the UAV mass.

III. Quadrotor Dynamics, Kinematics, and Control

A. Quadrotor Model

1) Dynamics and Kinematics: The equations of motion of a quadrotor having plus $(+)$ configuration using Newton-Euler formalism can be expressed as [14], [15]

$$m\ddot{p} = -mge_3 + FR^W e_3 \\ I\dot{\Omega} = -\Omega \times I\Omega + \tau \tag{8}$$

where m is the quadrotor mass, $p = [x, y, z]^\top \in \mathbb{R}^3$ is the quadrotor position in Cartesian coordinates, $I \in \mathbb{R}^{3\times3}$ is the inertia tensor, $F \in \mathbb{R}$ is the total thrust generated by the rotors. Similarly, $\tau \in \mathbb{R}^3$ is the input torque, $\Omega = [\bar{p}, \bar{q}, \bar{r}]^\top$ is the angular velocity, $R^W \in \mathbb{R}^{3\times3}$ is the rotation matrix from the body frame to the world frame, g is the gravitational acceleration constant, and $e_3 = [0, 0, 1]^\top$ is the unit vector along the corresponding z-axis.

The configuration of the rotors and their spin direction can be seen in Figure 2. As can be seen in Figure 2 four

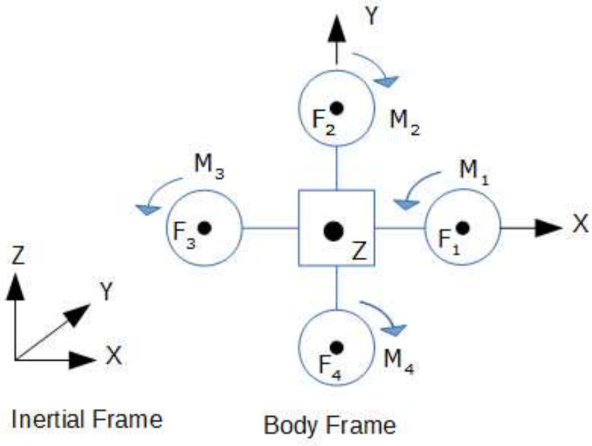

Fig. 2. Frames, forces, and moments acting on the quadrotor.

identical rotors produce thrust along the z-axis of the body frame and moments about the body frame axes. Those thrusts and moments depend on the angular speeds of the rotors. As mentioned, the total thrust is along the z-axis of the body frame (i.e., along with e_3 expressed in the body frame) and

the moments about the body frame axes. The total thrust is

$$F = F_1 + F_2 + F_3 + F_4 \tag{9}$$

where F_i denotes the thrust generated by the corresponding rotor and can be calculated as

$$F_i = k_F \omega_i^2, \quad i = 1, \ldots, 4, \tag{10}$$

where k_F is a motor constant, and w_i are the corresponding rotors' angular speeds (rad/s). Similarly, the rotational moments of the body frame axes can be calculated as

$$\begin{aligned} \tau_\phi &= l\left(F_2 - F_4\right), \\ \tau_\theta &= l\left(F_3 - F_1\right), \\ \tau_\psi &= M_1 - M_2 + M_3 - M_4. \end{aligned} \tag{11}$$

where F_i are the above mentioned thrusts and M_i denote the moments generated by the individual rotors and are given by

$$M_i = k_M \omega_i^2, \quad i = 1, \ldots, 4, \tag{12}$$

where τ_ϕ, τ_θ, and τ_ψ are the momenta about the body axes, k_M are moment constant of the motors, and w_i are once again the rotor angular speeds. Note that the moments about the body axes τ_ϕ, τ_θ, and τ_ψ in (11) lead to the roll, pitch, and yaw motions, respectively.

As in [15] we model the DC motors at the rotors with first-order systems in the form

$$\dot{\omega}_i = \tau_m\left(\omega_i^{des} - \omega_i\right), \quad i = 1, \ldots, 4 \tag{13}$$

where τ_m and w_i^{des} represent the motor gain and desired angular speed of the i'th rotor, respectively. More complicated models are also possible, although the first-order model is sufficient for the application we consider here.

In order to represent the relative orientation we utilize Z-Y-X Euler angles for which the rotation matrix R^W which represents the quadrotor body frame in the inertial/world frame can be expressed as

$$R^W = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{14}$$

where s_ℓ and c_ℓ respectively stand for $\sin(\ell)$ and $\cos(\ell)$, where ℓ represents the Euler angles ϕ, θ, ψ.

2) Quadrotor Control: The quadrotor control architecture is usually a cascade-based control loop. Cascade-based control architectures are based on the singular perturbation theory [16]. This approach assumes that the inner loop, in our case, attitude control is exponentially stable and that the inner loop bandwidth is greater than the dynamics of the outer loop, so the controller of the outer loop can be designed without considering the dynamics of the inner loop [17].

In this paper, we utilize a feedback linearization-based controller for each loop similar to the work in [15]. The details of the controller are described below.

Altitude Control: Assuming that the attitude angles are small and ensuring $(\cos(\theta)\cos(\phi) \neq 0)$, the feedback linearization-based approach can be adopted from [18] as

$$F = \left(g + u_z^{sw}\right) \frac{m}{\cos(\theta)\cos(\phi)} \tag{15}$$

where the term g cancels the effect of gravity and the input u_z^{sw} is generated such that to achieve and keep a predefined altitude for the agent (and the swarm).

Attitude Control: Feedback linearization based controller can also be used for attitude stabilization of a quadrotor [15], [19]. We utilize the approach in [15] for attitude stabilization which is a PD type controller, which is described below. The body moments are expressed in terms of the control inputs given the desired angular motion as

$$\begin{aligned} \tau_\phi &= \left(u_\phi + u_\phi^{pd}\right) \frac{I_{xx}}{l} \\ \tau_\theta &= \left(u_\theta + u_\theta^{pd}\right) \frac{I_{yy}}{l} \\ \tau_\psi &= \left(u_\psi + u_\psi^{pd}\right) I_{zz} \end{aligned} \tag{16}$$

where I_{xx}, I_{yy}, I_{zz} are the diagonal entries of the inertia tensor matrix, I. The terms u_ϕ, u_θ, u_ψ are the expressions from the system dynamics which can be expressed as

$$\begin{aligned} u_\phi &:= \dot{\theta}\dot{\psi}\frac{I_{yy} - I_{zz}}{I_{xx}} - \dot{\theta}\omega_r \frac{J_r}{I_{xx}} \\ u_\theta &:= \dot{\phi}\dot{\psi}\frac{I_{zz} - I_{xx}}{I_{yy}} + \dot{\phi}\omega_r \frac{J_r}{I_{yy}} \\ u_\psi &:= \dot{\phi}\dot{\theta}\frac{I_{xx} - I_{yy}}{I_{zz}} \end{aligned} \tag{17}$$

where J_r is the rotor's inertia and $w_r := -w_1 + w_2 + -w_3 + w_4$. Rotors with the odd id are subtracted since they are rotating in the opposite direction to the others. The low level PD controllers for the rotational dynamics are given by

$$\begin{aligned} u_\phi^{pd} &= k_{p,\phi}\left(\phi_d - \phi\right) + k_{d,\phi}\left(\dot{\phi}_d - \dot{\phi}\right) \\ u_\theta^{pd} &= k_{p,\theta}\left(\theta_d - \theta\right) + k_{d,\theta}\left(\dot{\theta}_d - \dot{\theta}\right) \\ u_\psi^{pd} &= k_{p,\psi}\left(\psi_d - \psi\right) + k_{d,\psi}\left(\dot{\psi}_d - \dot{\psi}\right) \end{aligned} \tag{18}$$

where ϕ, θ, and ψ are the Euler angles, $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ are the Euler angle rates. Similarly, ϕ_d, θ_d, and ψ_d are the desired Euler angles, and $\dot{\phi}_d$, $\dot{\theta}_d$, and $\dot{\psi}_d$ are the desired Euler angle rates.

Assuming that the quadrotor is in hover, referring to [14], [19] one can convert the outputs of the position controllers to the desired angles (ϕ_d and θ_d) using the equations

$$\begin{aligned} \phi_d &= \frac{1}{g}\left(u_x \sin(\psi) - u_y \cos(\psi)\right) \\ \theta_d &= \frac{1}{g}\left(u_x \cos(\psi) + u_y \sin(\psi)\right) \end{aligned} \tag{19}$$

Note here that we take the desired yaw angle as $\psi_d = 0$. Moreover, the desired Euler Angle rates $\dot{\phi}_d$, $\dot{\theta}_d$, and $\dot{\psi}_d$ are also taken as 0. The variables u_x and u_y in (19) represent the control variables for the Cartesian coordinates (x, y) and are discussed below.

Position Control in the Cartesian Space: Position controllers (x, y) in the Cartesian space are considered as the outer loops in quadrotor control systems. This is achieved by two different methods. The first one is through the controller developed for the double integrator agents in (4). In other

words, we utilize the x and y components of the control input in (4) in (19) as $u_x = u_x^{sw}$ and $u_y = u_y^{sw}$. Alternatively, one can employ also a PD controller on top of the controller in (4). In that case, the u_x^{sw} and u_y^{sw} calculated in (4) are fed into the PD controller as

$$\begin{aligned} u_x &= k_{p,x}\left(u_x^{sw} - x\right) - k_{d,x}\dot{x} \\ u_y &= k_{p,y}\left(u_y^{sw} - y\right) - k_{d,y}\dot{y} \end{aligned} \tag{20}$$

where k_p and k_d are the controller parameters. The outputs u_x and u_y of the PD controller are used in (19), instead of directly using u_x^{sw} and u_y^{sw} from (4).

As will be shown in Section (IV), employing a PD controller on top of the existing controller leads to some performance improvements overall. However, the system works well without the PD controller as well.

The overall procedure can be summarized as follows: (i) From (4) the desired (x, y) control inputs u_x^{sw} and u_y^{sw} are calculated, the control input u_z^{sw} is calculated based on the desired altitude; (ii) Desired rates are calculated using (19) which are used to calculate the control inputs u_ϕ^{pd}, u_θ^{pd}, and u_ψ^{pd} in (18); (iii) The values of u_ϕ, u_θ, and u_ψ are obtained from (17) and are utilized in calculating the torques in (16); (iv) the desired thrust is obtained from (15); (v) finally, equations (9) and (11) are simultaneously solved for the motor speeds, as discussed below.

Control of the Rotors At the final stage, by simultaneously solving (9) and (11) the output of the controllers are transferred into the squared desired rotor speeds as

$$\begin{bmatrix} \left(\omega_1^{\text{des}}\right)^2 \\ \left(\omega_2^{\text{des}}\right)^2 \\ \left(\omega_3^{\text{des}}\right)^2 \\ \left(\omega_4^{\text{des}}\right)^2 \end{bmatrix} = \frac{1}{k_F} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & l & 0 & -l \\ -l & 0 & l & 0 \\ \kappa & -\kappa & \kappa & -\kappa \end{bmatrix}^{-1} \begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{21}$$

where $\kappa = k_M/k_F$. As in [15] it is possible to saturate the obtained desired rotor speeds between a predefined minimum and maximum values, which are usually chosen as nonnegative constants. Then, the desired motor speeds ω_i^{des} are computed using the square root operation. The obtained desired speeds are applied as input to the motor dynamics in (13).

B. MRS UAV System

For ROS implementations, we used the MRS UAV System, which was developed by the Czech Technical University in Prague [17]. The widely-used dynamical model of a multi-rotor aerial vehicle [20] is used. The model is, in principle, the same discussed in the previous section with the difference that in the MRS UAV system, they use the rotation matrix variation for angular dynamics representation instead of the body rates used in (8). The positional dynamics representation is the same as in (8).

The MRS UAV System is a highly-capable platform that consists of different parts. It is usable for many different scenarios. We are interested in its swarming capabilities. Trackers are a core part of the system. They generate references for the controllers. In most of the scenarios, the MPC tracker is used. However, the speed tracker is needed in our case, which requires direct access to desired states. It allows direct control of the desired speed and/or acceleration of the UAV while maintaining the desired height and heading. The speed tracker only constrains the first derivative of references given by a low-pass filter, which provides extra safety measures.

UVDAR is a novel onboard relative localization method that can be chosen as one of the many localization methods in the system. It does not require any communication between drones and any infrastructures. In the circular following case, according to the [13] on average, the follower lags by $4m$. Also, the maximum distance for reliable detection is $15m$. It performed well in our simulations except for some shortages.

For low-level control, a commercially available Pixhawk controller is used. First, our controller produces reference values and passes them to the speed tracker. The speed tracker puts a low-pass filter to those values for smoother flight and passes them to the MPC controller. MPC controller calculates the desired attitude rate and thrust commands and passes them to the Pixhawk for conversion to motor speeds.

As a realistic simulator, Gazebo is running. It uses a physical engine for gravity, inertia, illumination, etc. For instance, in our simulations, drones may collide at low speeds if parameters are not set right and the collision avoidance algorithm does not perform well. Since it is low speeds, they do not necessarily fall, but they tilt and recover.

IV. Simulation Results

This section will show the simulation results we made according to the models discussed above. In the MATLAB simulation, we consider the case where the target is stationary, and the swarm is settling around a target. In the ROS simulations, on the other hand, we will consider the case of a moving target following a particular trajectory at a constant speed. As will observe in the following, the swarm can capture and enclose the target as expected. Two simulations in the ROS environment are presented, one with five agents and the other one with four agents.

A. Matlab

For the simulations in Matlab, we utilize the model and parameters of Crazyflie 2.0 nano-quadrotor with the same parameters as in [15]. The agents first hover to the desired altitude (the target's altitude) and then move toward the target and achieve the capture/enclose objective. The target is at rest, and the swarm is positioned in the containment region. Due to the repulsive forces, they do not enter the distancing region of the target, as can be seen in Figures 3 and 4. Figure 3 shows the top view of the agent trajectories together with the ellipsoidal region plotted. The innermost ellipse (in red) marks the distancing region, whereas the outermost ellipse (in green) marks the containment region. As can be seen from the figure, the agents achieve the desired behaviour. The blue ellipse between the distancing and the containment regions represents the distance at which the attraction and repulsion forces to the target balance.

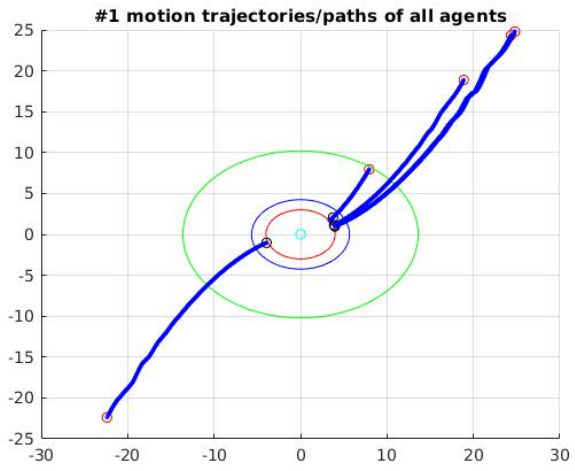

Fig. 3. Trajectories of the swarm, along the x-y plane.

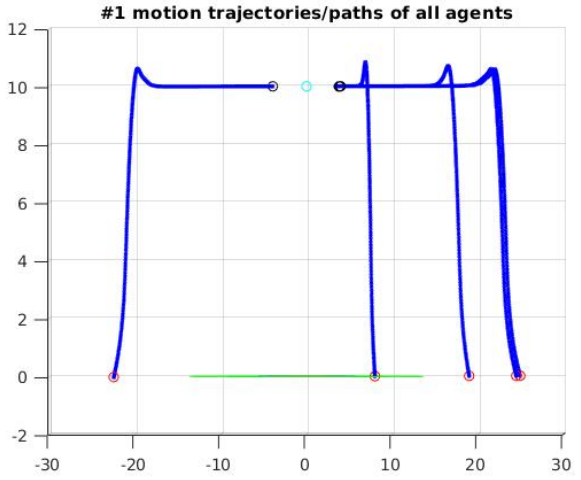

Fig. 4. Trajectories of the swarm, movement along the z direction.

However, due to the uncertainties about the target velocity and acceleration and the the inter-agent repulsion forces usually the agents do not settle on that ellipsoid. Figure 4 shows the side view of the same simulation, in which the agent motion to the desired altitude can also be observed.

We used the duration for agents to settle in the containment region to highlight the performance with and without a PD controller employed with the corresponding results shown in Figures 5 and 6. The conditions for both simulations are identical except for the randomized starting positions. In the simulation shown in Figure 5 shows a result for the case in which the PD controller was employed. As can be seen, all agents settle inside the containment region in about $10s$. $z^T Pz$ is the 'ellipsoidal' distance to the target. In other words, $z^T Pz = 1$ corresponds to the distancing region, whereas $z^T Pz = c$ to the containment region. Ellipsoidal distances of the agents to the target converge between those two lines as can be seen from the figures. Figure 6 shows the result for a simulation in which the PD controller was

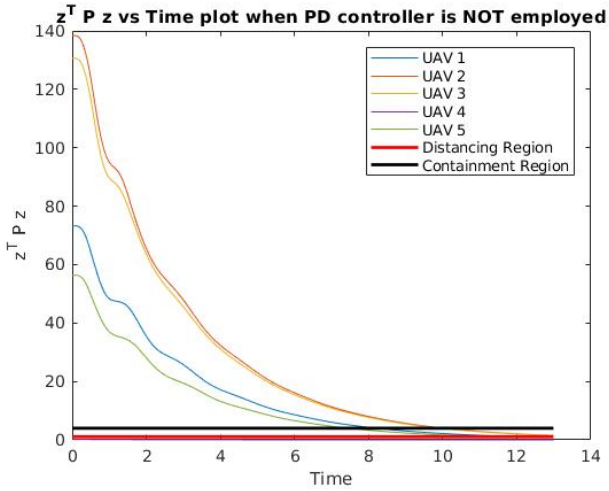

Fig. 5. $z^\top Pz$ vs. time graph for the case with a PD-controller employed for Cartesian position control. All agents settle inside the distancing region just after $t = 10s$

not utilized. This time again all agents exhibit the desired behaviour although this time it takes about $16s$ for all agents to settle in the containment region. We observed that, in general, utilization of the PD controller led to a faster converge into the containment region.

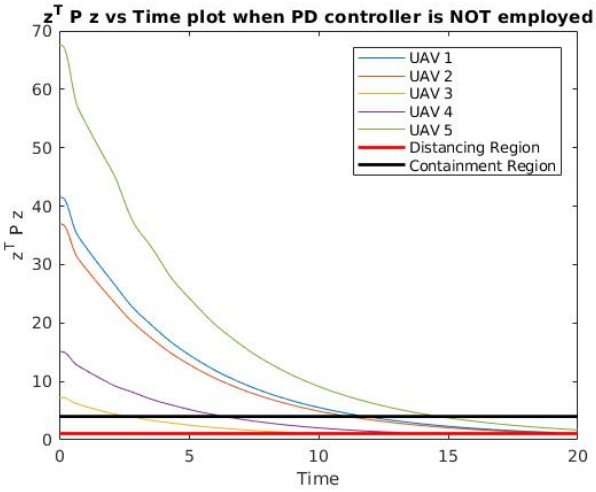

Fig. 6. $z^\top Pz$ vs. time graph for the case without a PD-controller for Cartesian position control. All agents settle inside the distancing region just after $t = 16s$

B. MRS UAV System Example

First, consider a group of five agents in the 3D space whose masses are $2.5kg$. Agents are modelled after the commercially available DJI F450 platform. The detection radius is $4m$. Agents detect their neighbours and the target by employing the UVDAR system. The target is moving with $2m/s$ constantly in both x and y directions. The target and the follower agents have a fixed altitude of $10m$. The α, β, and γ coefficients are chosen as $11.57, 10, 1$, respectively. However, initially attractive and repulsive forces were weaker

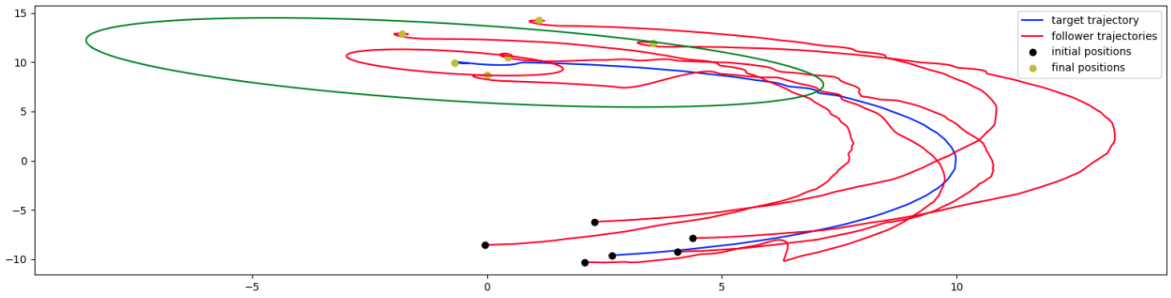

Fig. 7. Trajectories of the swarm and the target. Disturbances due to UVDAR shortages.

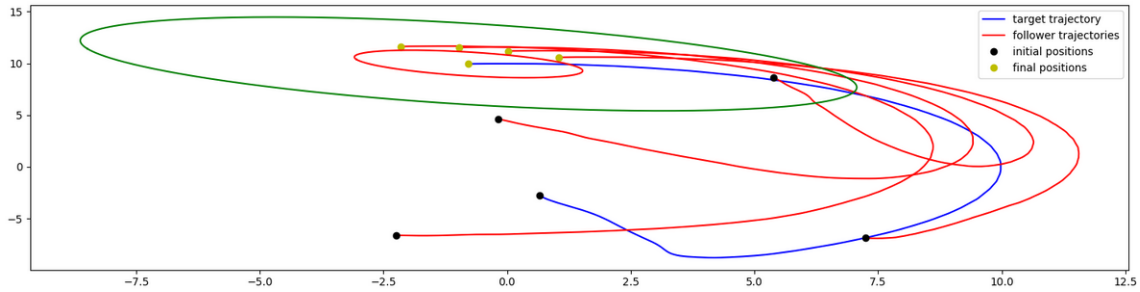

Fig. 8. Trajectories of the swarm and the target.

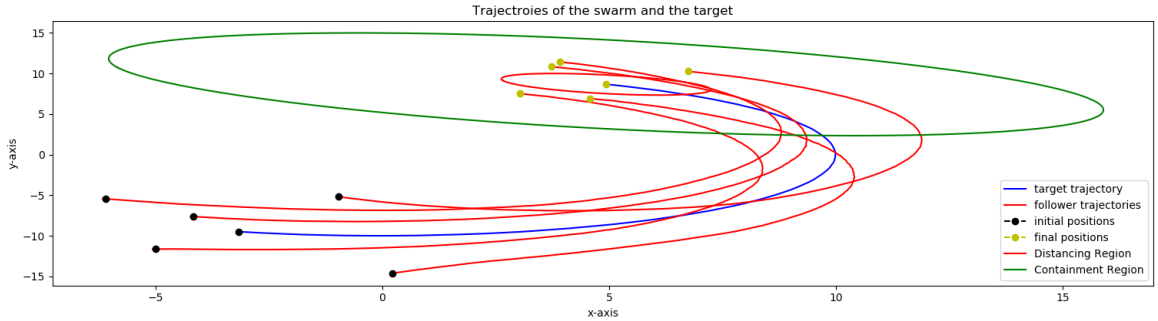

Fig. 9. Trajectories of the swarm and the target. Agents in the containment region are clearly visible.

than needed; therefore, we amplified the velocity, acceleration, and the ellipsoidal term by a constant gain of 10, which resulted in slightly jerky drone movements. Therefore, the overall controller output was scaled by 0.1. We used a larger ellipsoidal region than [1] since now UAVs have actual body sizes. They could not squeeze into a small ellipsoid without colliding with each other.

We performed simulations with four and five followers. Figure 7 shows a result for five agents in which there was a shortage with the UVDAR system. Therefore, there are some disturbances with the UAV trajectories, which can be seen in the figure. However, the outcome was still successful, and UAVs managed to stay in the containment region.

Figure 8 shows a simulation with 4 UAVs, with no UVDAR shortages. As can be seen the UAVs managed to follow smooth trajectories. At the start of that simulation, due to the randomized starting positions of the agents, one of the agents was close to the target; therefore, there was an unexpected change in the target trajectory. However, the simulation continued as expected once they were sufficiently far away. The figures show the evolution of the trajectories up to $t = 60s$. Figure 9 shows another simulation for the case we utilized $alpha = 22$, i.e., a larger value for the parameter. It is visible that agents move to the containment region. A video of the simulation can be seen from the link https://youtu.be/QBzuLjv4rLM. Note that the video is $1/3$ time of the real-time.

C. Performance Comparison

To evaluate the algorithm's performance, we ran the algorithm for four different scenarios and measured the convergence time of the tracking errors between agents and the target. The tracking error is the norm of the distance between the target and the agents. Those scenarios are as follows;

- A static target enclosed by six followers. Followers' initial positions are randomized.

- A moving target along a circular trajectory followed by six followers. Followers' initial positions are randomized but restricted to the bottom right corner.
- A moving target along a circular trajectory followed by six followers. Followers are split into two groups. The first group is randomly initialized at the bottom right corner, and the second group is randomly initialized at the upper left corner.
- A moving target along a flower-like trajectory followed by three followers. Followers' initial positions are randomized. The trajectory is inspired by the hypotrochoid shape that is used in [1]. Target moves along the x direction according to the equation, $x = 10\cos(0.08t) + 10\cos(0.053t)$ and moves along the y direction according to the equation $y = 10\sin(0.08t) + 10\sin(0.053t)$

The obtained results are shown in the Figures from 10 to 17. In the Figures 10, 12, 14, and 16 the x-axis starts around $60s$, since this amount of time is needed for the drones to initialize and get ready before the algorithm starts. As can be observed from the figures, in all the cases the agents successfully fulfill the containment objective. In all scenarios the agents reach a steady-state in approximately similar duration. The slight difference in the actual times of convergence depends on factors such as initial positions of the agents, agent number, and trajectory complexity. The UVDAR system seems effective neighbor position sensing methodology. Still, it imposes some computational requirement, although its computational load might be lower than some alternative methods [13]. The algorithm is in general scalable although in implementations on real hardware the sensing and computation capabilities of the agents might impose practical limitations on the number of agents. The obtained results show that the methodology developed for simple models in [1] can easily be implemented on realistic agents such as quadrotor UAVs and verify the effectiveness of the method.

V. Concluding Remarks

In this paper we implemented a recently proposed strategy for the uncertain target capturing problem with generic UAV models in MATLAB and ROS environments. The validity of the proposed strategy has been verified for swarms composed of complicated agent dynamics and realistic agent sensing methodologies. It has been shown that the swarm enters an ellipsoidal ring around the target and remains there. In the simulations performed in ROS, a UVDAR based relative localization system is employed, which uses only local relative sensing information and does not require access to global GPS data. The obtained results support the theoretical conclusions and verify the effectiveness of the methodology. Future research may focus on achieving target capturing on more complicated regions around the target, such as polygonal regions or other complex shapes. Alternative sensing and/or communication approaches and better control algorithms can also be employed. Uncertainties in the agent dynamics, in addition to the sensing errors, can also be considered.

References

[1] A. Bono, L. D'Alfonso, G. Fedele, and V. Gazi, "Target capturing in an ellipsoidal region for a swarm of double integrator agents," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 5, pp. 801–811, 2022.

[2] R. Sepulchre, D. A. Paley, and N. E. Leonard, "Group coordination and cooperative control of steered particles in the plane," in *Group Coordination and Cooperative Control*. Springer, 2006, pp. 217–232.

[3] T.-H. Kim and T. Sugie, "Cooperative control for target-capturing task based on a cyclic pursuit strategy," *Automatica*, vol. 43, no. 8, pp. 1426–1431, 2007.

[4] V. Gazi and K. M. Passino, *Swarm Stability and Optimization*. Springer Verlag, January 2011.

[5] L. Blázovics, T. Lukovszki, and B. Forstner, "Target surrounding solution for swarm robots," in *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*. Springer, 2012, pp. 251–262.

[6] Y. Kobayashi, K. Otsubo, and S. Hosoe, "Design of decentralized capturing behavior by multiple mobile robots," in *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*. IEEE, 2006, pp. 13–18.

[7] J. Yao, R. Ordonez, and V. Gazi, "Swarm tracking using artificial potentials and sliding mode control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 749–754, 2007.

[8] V. Gazi, B. Fidan, R. Ordóñez, and M. İlter Köksal, "A target tracking approach for nonholonomic agents based on artificial potentials and sliding mode control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 134, no. 6, 2012.

[9] B. Halcı, V. Gazi, and O. Cihan, "Modelling and coordination of a swarm of quadrotors using lagrange dynamics and potential functions," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 963–970.

[10] Z. Lin, M. Broucke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 622–629, 2004.

[11] J. A. Marshall, *Coordinated autonomy: Pursuit formations of multi-vehicle systems*. University of Toronto, 2005.

[12] A. Sinha and D. Ghose, "Generalization of the cyclic pursuit problem," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 4997–5002.

[13] e. a. Walter, Viktor, "Uvdar system for visual relative localization with application to leader–follower formations of multirotor uavs," *IEEE Robotics and Automation Letters*, no. 4.3, pp. 2637–2644, 2019.

[14] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, p. 664–674, 2012.

[15] M. A. Toksöz, S. Oğuz, and V. Gazi, "Decentralized formation control of a swarm of quadrotor helicopters," in *15th IEEE International Conference on Control and Automation (ICCA)*. IEEE, 2019, pp. 1006–1013.

[16] M. R. Roussel, *Nonlinear Dynamics: A hands-on introductory survey*. Morgan Claypool Publishers, 2019.

[17] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," *Journal of Intelligent and Robotic Systems*, vol. 102, no. 1, pp. 1–28, 2021.

[18] A. Abdessameud and A. Tayebi, "Formation control of vtol unmanned aerial vehicles with communication delays," *Automatica*, vol. 47, no. 11, pp. 2383–2394, 2011.

[19] J. Seo, Y. Kim, S. Kim, and A. Tsourdos, "Consensus-based reconfigurable controller design for unmanned aerial vehicle formation flight," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 226, no. 7, pp. 817–829, 2021.

[20] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," *IEEE conference on decision and control (CDC)*, vol. 49, 2010.

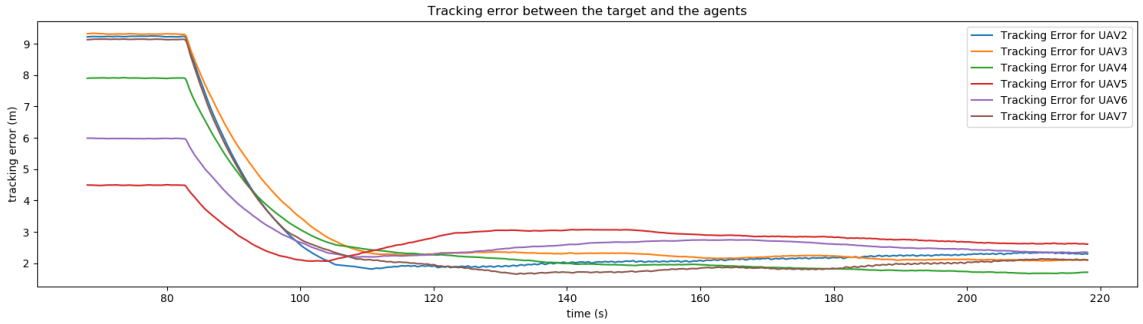

Fig. 10. Tracking error between the target and the agents. A static target enclosed by six followers. Followers initial positions are randomized.

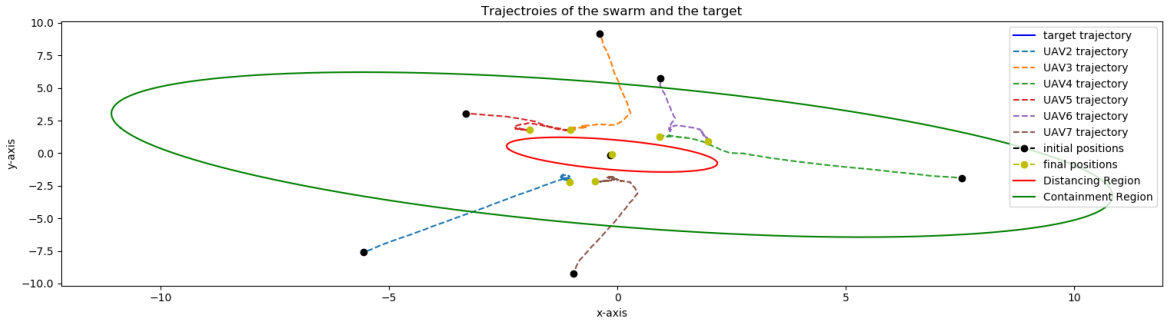

Fig. 11. Trajectories of the target and the agents. A static target enclosed by six followers. Followers initial positions are randomized.

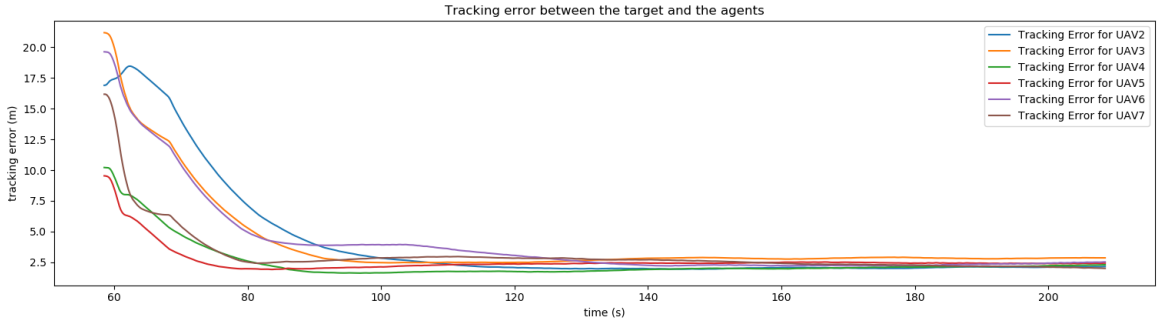

Fig. 12. Tracking error between the target and the agents. A moving target along a circular trajectory followed by six followers. Followers initial positions are randomized but restricted to bottom right corner.

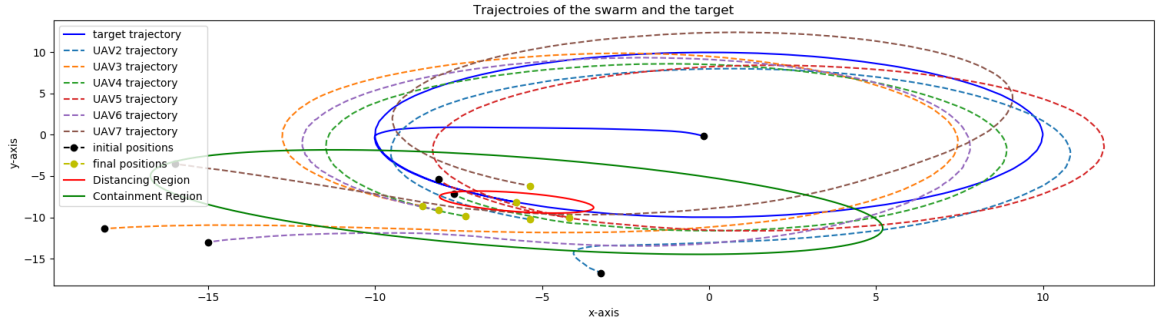

Fig. 13. Trajectories of the target and the agents. A moving target along a circular trajectory followed by six followers. Followers initial positions are randomized but restricted to bottom right corner.

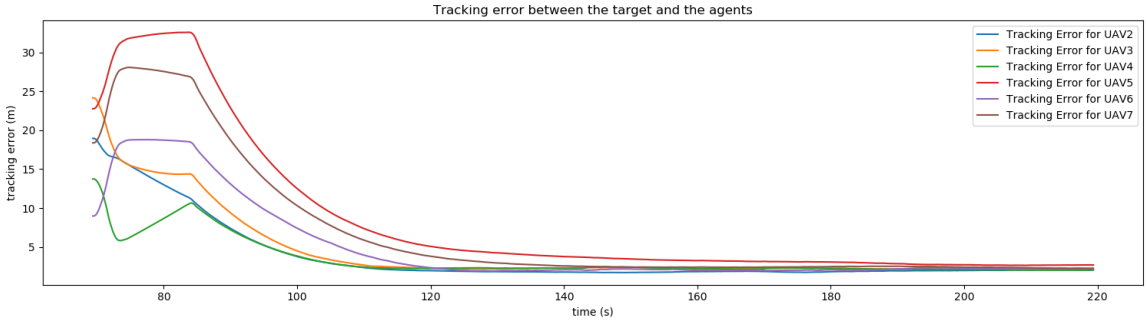

Fig. 14. Tracking error between the target and the agents. A moving target along a circular trajectory followed by six followers. Followers are split to two groups. First group is randomly initialized at the bottom right corner, second group is randomly initialized at the upper left corner.

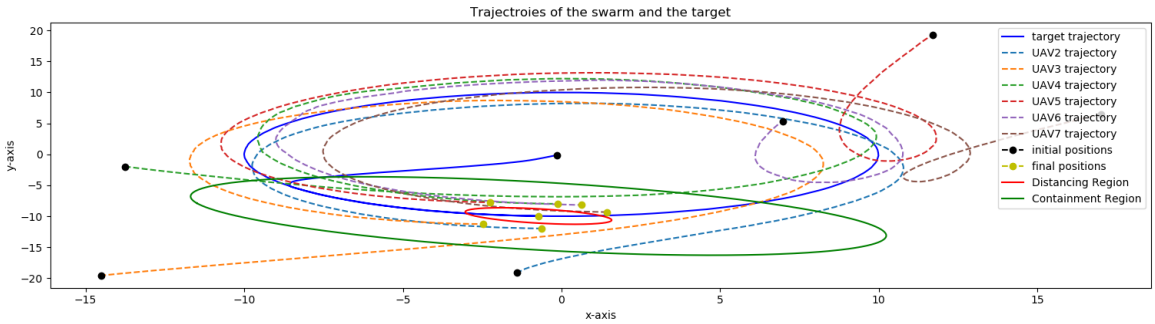

Fig. 15. Trajectories of the target and the agents. A moving target along a circular trajectory followed by six followers. Followers are split to two groups. First group is randomly initialized at the bottom right corner, second group is randomly initialized at the upper left corner.

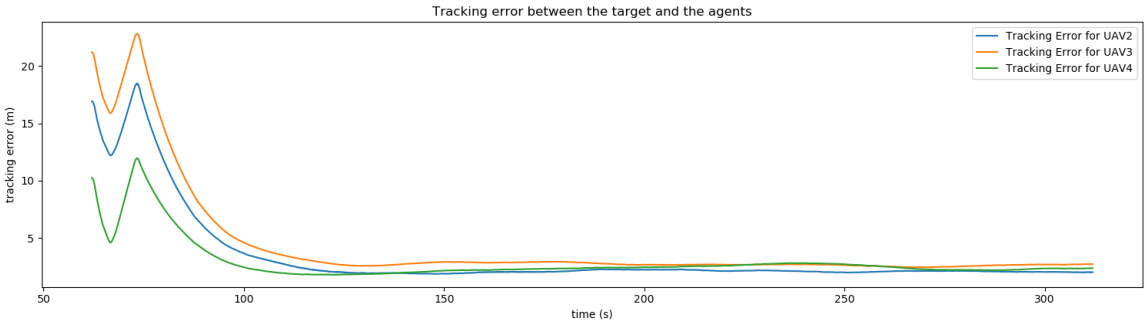

Fig. 16. Tracking error between the target and the agents. A moving target along a flower-like trajectory followed by three followers. Followers initial positions are randomized.

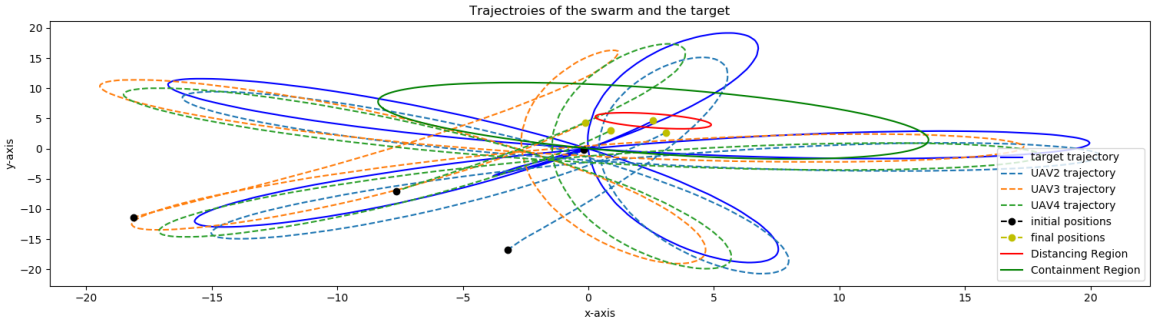

Fig. 17. Trajectories of the target and the agents. A moving target along a flower-like trajectory followed by three followers. Followers initial positions are randomized.