2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561054

Model Predictive Control for Cooperative Hunting in Obstacle Rich and Dynamic Environments

Jacky Liao and Che Liu and Hugh H.T. Liu

Abstract— **This paper studies the cooperative hunting problem, where a group of agents encircle a target while avoiding collisions with each other and with obstacles in the environment. The paper deals with obstacle rich environments and dynamic (moving obstacle) environments by formulating the problem as both a control problem and a planning problem. A model predictive control (MPC) method is proposed which integrates a multi-agent planner with the cooperative hunting objective while also accounting for UAV dynamics. The effectiveness of the proposed method is verified through a comparative analysis with optimal reciprocal collision avoidance (ORCA), and then validated through experiments with quadrotor UAVs. Using the proposed method, agents no longer get stuck in local minima for obstacle rich environments and capture the target faster with shorter trajectories in moving obstacle environments.**

I. INTRODUCTION

Control of multi-agent systems is a popular topic in robotics with various applications for UAVs, such as formation flight, surveying, and coordinated package delivery. The methodology traces back to the theory of flocking behaviour [1]. Agents in the flock follows a set of rules based on their local observation. The rule of alignment matches the speed and heading direction of neighbouring agents, and the rules of cohesion and separation allows agents to group without risking collision. By following these simple rules, the agents behave as a collective and express emergent behaviour.

Solutions based on flocking methods scale more easily with the number of agents compared to centralized methods, as the agents use mainly local information. However, flocking rules need to be modified based on their necessity for the task at hand, and global signals may need to be added to enforce the objective. For example, alignment and cohesion may not be necessary for surveying a wide area, and additional rules may be added for maximizing coverage. Leader-follower methods use a hybrid, where leader agents receive global information, and follower agents use local information.

This paper focuses on the cooperative hunting problem. The objective is to coordinate a group of agents to surround a static or moving target. Cooperative hunting is seen in nature by pack hunters such as wolves and chimpanzees, with the goal being to capture prey. Due to its connection with nature and with multi-agent systems, many existing solutions are biologically inspired or derived from flocking theory. Weitzenfeld et al. [2] developed a leader-follower algorithm based on wolf pack hierarchies. The leader agent is designated as the alpha wolf and prioritizes target capture, while the followers are beta wolves and maintain a formation behind the alpha wolf. Muro et al. [3] simulated wolf hunting behaviour without imposing any hierarchical structure. Instead, the agents move toward the target until a minimum safe distance is reached, and afterwards move away from other agents who are also within the safe distance. Similarly, Madden et al. [4] modeled wolf hunting behaviour as a set of foraging states with transition probabilities rather than as a social hierarchy.

Many methods enforce target surrounding by formulating the encirclement objective in polar coordinates, where the agents maintain a desired distance away from the target and a desired angular separation with other agents. These works therefore share same objective, but differ in their control strategy. Examples of such strategies include nonlinear model predictive control [5], Lyapunov based control [6], and decentralized control adapted to different robot types [7]. For multi target cases, auction algorithms [8] and hybrid dynamic task allocation [9] can be used for assigning targets to each hunter. Wu et al. [10] predicts the target motion and used the prediction to optimize the agent encirclement points. Yao et al. [11] developed a control law which allowed any arbitrary desired angular spacing among agents in 3D.

The methods presented are tested in simple scenarios containing few obstacles. These algorithms use some form of local planning where nearby obstacles are considered, but not the entire environment. Notable examples of such algorithms for multi-agents are potential field based methods [12], flocking algorithm modifications [13], and more recently optimal reciprocal collision avoidance (ORCA) [14].

However, such methods struggle in more complex scenarios such as obstacle rich environments and moving obstacle environments. In the former case, agents can get stuck in local minima where the obstacles block the path to the target. In the latter case, the agents produce poor trajectories as they do not plan for future obstacle locations. To better resolve these complex scenarios, the agents must plan ahead in the environment. This extends the cooperative hunting problem to a cooperative path planning problem.

Examples of multi-agent path planning algorithms include extensions to A* search for multiple agents by decoupling the problem into single agent searches [15]. The agents plan in an assigned priority sequence and store their path into a reservation table, which is taken into account by the subsequent agents. Another approach is to have agents first plan independently, and in the event of a conflict, lower priority agents shift their path to make way for higher priority

Authors are affiliated with the Flight Systems and Control Lab, Institute of Aerospace Studies, University of Toronto, North York, Canada. jacky.liao@flight.utias.utoronto.ca, charlesl.liu@mail.utoronto.ca, liu@utias.utoronto.ca

agents [16]. The planning problem can also be formulated as a decoupled iterative sequential convex programming problem. Optimal agent actions are produced with linearized collision constraints at each timestep [17]. Finally, Hönig and Ayanian [18] used Bézier curve interpolation to optimize trajectories generated from graph based planning, and to prevent inter-agent collisions, hard margin SVM is used to generate safe corridors.

The contribution of this paper is to combine cooperative hunting with multi-agent planning using model predictive control (MPC). In obstacle rich environments, the proposed method will better avoid local minima, and in moving obstacle environments, the agent will capture the target faster by minimizing obstacle interaction. The proposed method is tested and compared with ORCA in simulation, and is afterwards validated in experiment with quadrotor UAVs.

II. PROBLEM FORMULATION

This paper focuses on the encirclement of a single target in 2D environments which contain obstacles. The motion and behaviour of obstacles and target is assumed known along with their full state.

A. Agent Dynamics

The agents are quadrotor UAVs where the inputs are desired roll and pitch denoted u_ϕ and u_θ. Actuation and communication delay is modeled using a first order approximation parameterized by α. Modelling such delay allows for the design of more robust controllers compared to using simple double integrator models, which is necessary when dealing with multiple UAVs in cluttered environments.

The dynamics can be represented more conveniently in terms of acceleration and jerk. Let (u_x, u_y) be the desired accelerations in the x-y directions. Linearization at constant height and zero attitude gives rise to the following state space, which is discretized in time afterwards yielding matrices $(\mathbf{F_x}, \mathbf{F_u})$.

$$u_x = g u_\theta \qquad u_y = -g u_\phi \tag{1}$$

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T \qquad \mathbf{u} = \begin{bmatrix} u_x & u_y \end{bmatrix}^T \tag{2}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & -\alpha \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \alpha & 0 \\ 0 & \alpha \end{bmatrix} \mathbf{u} \tag{3}$$

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4}$$

$$\mathbf{x}_{t+1} = \mathbf{F_x}\mathbf{x}_t + \mathbf{F_u}\mathbf{u}_t \tag{5}$$

The parameter α can be determined using system identification. This was done for the Crazyflie 2.1 quadrotor [19] by approximating its attitude controller and actuation delay through experimental data and analysis of onboard gains. It was determined that $\alpha = 5.5759$ and this value will be used for the simulation and experiment sections.

This paper assumes that the linear model applies to any direction in the x-y plane, and that the control delay is equal in all directions of travel. This assumption allows for non-zero yaw in practice, as the actual UAV inputs can be rotated from (u_x, u_y).

B. Encirclement Criteria

The encirclement criteria can be formulated in polar coordinates centered at the target. The target is located at (x_d, y_d), and the polar coordinates of agent i relative to the target is as follows.

$$r_i = \sqrt{(x_i - x_d)^2 + (y_i - y_d)^2} \tag{6}$$

$$\varphi_i = \operatorname{atan2}(y_i - y_d, x_i - x_d) \tag{7}$$

The encirclement criteria is for the agents to reach a desired radial distance r_d from the target as well as reach desired angles Φ_d. This is measured using error tolerances ϵ_r and ϵ_ϕ.

$$|r_i - r_d| \le \epsilon_r \qquad i = 1, 2, \ldots, N \tag{8}$$

$$|\varphi_i - \varphi_{i,d}| \le \epsilon_\varphi \qquad \Phi_d = \{\varphi_{1,d}, \varphi_{2,d}, \ldots, \varphi_{N,d}\} \tag{9}$$

Φ_d contains the desired angles to the target which are equally spaced apart, and is assigned based on the agent locations in order to reduce the total distance traveled. Hence, Φ_d is a permutation of angle assignments where each element $\varphi_{i,d}$ is a multiple of $2\pi/N$. This paper assigns target angles by minimizing the absolute angle difference using the Hungarian algorithm. With this objective formulation, the agents to spread out as they converge to their desired angles and agent interaction is minimized. The alternative of assigning goal locations equally spaced around the target generates linear trajectories, which can lead to many undesirable agent interactions. The capture time $T_{i,c}$ is defined when agent i satisfies both bounds.

III. CONTROL STRATEGY

A. Target Encirclement

The encirclement objective can be solved using optimal control and feedback linearization. Define the axis $\mathbf{n}_{i,r}$ pointing outwards from the target to agent i, and $\mathbf{n}_{i,\varphi}$ pointing in its perpendicular direction. The encirclement error in the $\mathbf{n}_{i,r}$ direction is $e_{i,r}$ which is the difference to r_d. The error in the $\mathbf{n}_{i,\varphi}$ direction $e_{i,\varphi}$ is the arc length to the desired angle $\varphi_{i,d}$. The inputs $(u_{i,r}, u_{i,\varphi})$ are the desired acceleration inputs in the respective directions. Together, the error dynamics follow the same state space dynamics (5) at a local level.

$$\begin{bmatrix} u_{i,r} \\ u_{i,\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi_i & \sin\varphi_i \\ -\sin\varphi_i & \cos\varphi_i \end{bmatrix} \begin{bmatrix} u_{i,x} \\ u_{i,y} \end{bmatrix} \tag{10}$$

$$\mathbf{u}'_i = \mathbf{R}_{\varphi_i}\mathbf{u}_i \tag{11}$$

$$\mathbf{e}_i = \begin{bmatrix} e_{i,r} \\ e_{i,\varphi} \\ \dot{e}_{i,r} \\ \dot{e}_{i,\varphi} \\ \ddot{e}_{i,r} \\ \ddot{e}_{i,\varphi} \end{bmatrix} = \begin{bmatrix} r_i - r_d \\ r_i(\varphi_i - \varphi_{i,d}) \\ \dot{r}_i - \dot{r}_d \\ \dot{r}_i(\varphi_i - \varphi_d) + r_i(\dot{\varphi}_i - \dot{\varphi}_{i,d}) \\ \ddot{r}_i - \ddot{r}_d \\ \ddot{r}_i(\varphi_i - \varphi_{i,d}) + 2\dot{r}_i(\dot{\varphi}_i - \dot{\varphi}_{i,d}) + r_i(\ddot{\varphi}_i - \ddot{\varphi}_{i,d}) \end{bmatrix} \tag{12}$$

$$\mathbf{e}_{i,t+1} = \mathbf{F_x}\mathbf{e}_{i,t} + \mathbf{F_u}\mathbf{u}'_{i,t} \tag{13}$$

As $\mathbf{n}_{i,r}$ and $\mathbf{n}_{i,\varphi}$ change direction while the agent moves, so do the inputs $(u_{i,r}, u_{i,\varphi})$ in Equation (11) which maintains the local linear system. To obtain a feedback controller, a LQR method in discrete time is used, explained by the book *Dynamic Programming and Optimal Control* in [20]. The quadratic cost matrices for the error and input are $(\mathcal{C}_\mathbf{e}, \mathbf{C}_\mathbf{u})$ which define the infinite horizon cost $\mathcal{J}_i$ for agent i. The typeface for the error is different since it is quadratic in the polar coordinate error, but not quadratic in the Cartesian error. On the other hand, it can be shown that the cost is quadratic in the original Cartesian input $\mathbf{u}$ since $\mathbf{u}'$ differs only by rotation and $\mathbf{C}_\mathbf{u}$ is a multiple of the identity matrix and effectively only a scaling term.

$$\mathcal{C}_\mathbf{e} = \mathrm{diag}(k_r, k_\varphi, k_{\dot{r}}, k_{\dot{\varphi}}, k_{\ddot{r}}, k_{\ddot{\varphi}}) \tag{14}$$

$$\mathbf{C}_\mathbf{u} = \mathrm{diag}(k_u, k_u) \tag{15}$$

$$\mathcal{J}_i = \sum_{t=0}^{\infty} \mathbf{e}_{i,t}^T \mathcal{C}_\mathbf{e} \mathbf{e}_{i,t} + \mathbf{u}_{i,t}'^T \mathbf{C}_\mathbf{u} \mathbf{u}_{i,t}' \tag{16}$$

The terms k_r and k_φ scale the cost in the radial distance and arc length distance. The ratio k_r/k_φ was set between 1 to 2 since in the worst case, an agent must travel an arc distance of half the circumference πr. The optimal gain $\mathbf{K}$ to minimize $\mathcal{J}_i$ is obtained by solving the following discrete time algebraic Riccati Equations (17, 18). This yields the optimal input $\mathbf{u}'^*$ which can be transformed back into desired roll and pitch.

$$\mathbf{P} = \mathbf{F_x}^T\mathbf{P}\mathbf{F_x} - (\mathbf{F_x}^T\mathbf{P}\mathbf{F_u})(\mathbf{C_u} + \mathbf{F_u}^T\mathbf{P}\mathbf{F_u})^{-1}(\mathbf{F_u}^T\mathbf{P}\mathbf{F_x}) + \mathcal{C}_\mathbf{e} \tag{17}$$

$$\mathbf{K} = -(\mathbf{C_u} + \mathbf{F_u}^T\mathbf{P}\mathbf{F_u})^{-1}(\mathbf{F_u}^T\mathbf{P}\mathbf{F_x}) \tag{18}$$

$$\mathbf{u}_i'^* = \mathbf{K}\mathbf{e}_i \tag{19}$$

Simultaneous encirclement can be enforced by penalizing the relative radial distance. Suppose N agents surround the target, the error in the radial direction is a joint term $\underline{\mathbf{e}}_r$. The underline notation represents terms which deal with treating the agents jointly.

$$\underline{\mathbf{e}}_r = \begin{bmatrix} r_1 - r_d & \dots & r_N - r_d \end{bmatrix}^T \tag{20}$$

The relative radial distance cost $\mathcal{J}_s$ is quadratic in $\underline{\mathbf{e}}_r$ through matrix $\underline{\mathcal{C}}_s$, which is weighted by k_s and normalized by N.

$$\mathcal{J}_s = \frac{k_s}{N} \sum_{i=0}^{N}\sum_{j=0}^{N} (r_i - r_j)^2 \tag{21}$$

$$= \frac{k_s}{N} \underline{\mathbf{e}}_r^T \begin{bmatrix} N-1 & -1 & \dots & -1 \\ -1 & N-1 & & \vdots \\ \vdots & & \ddots & \\ -1 & \dots & & N-1 \end{bmatrix} \underline{\mathbf{e}}_r \tag{22}$$

$$= \underline{\mathbf{e}}_r^T \underline{\mathcal{C}}_s \underline{\mathbf{e}}_r \tag{23}$$

Including the simultaneous cost centralizes the control problem by coupling the agent states, and the same LQR control can be formulated with the joint agent states and inputs. The effect of the simultaneity cost can be seen in Figure 1, compared with trajectories without in Figure 2. The target is located $(3,2)$ with a desired surrounding radius of 1 m, and an outer circle of radius 1.5 m is drawn as well to visualize the simultaneous encirclement. The capture time difference with additional simultaneous cost is 0.1 s, compared to 6.9 s for the case without simultaneous cost. The agents converge toward the target radius simultaneously for the former case, whereas in the latter case the blue agent lags behind due to traveling a longer distance which is not considered by the others.

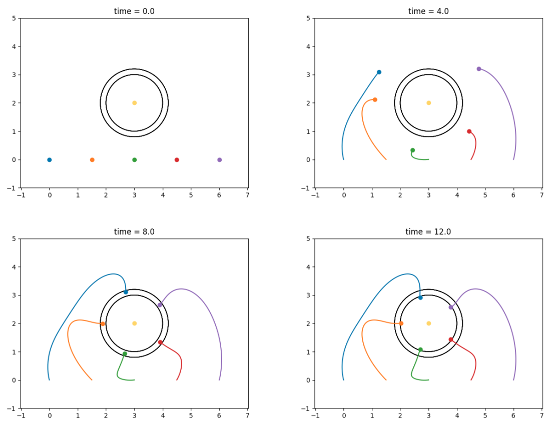

Fig. 1. Trajectories with additional simultaneous cost

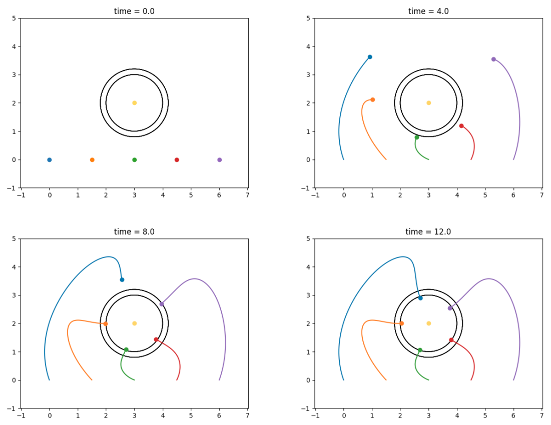

Fig. 2. Trajectories without additional simultaneous cost

B. Planning For Obstacles

A multi-agent path planner generates trajectories which avoid local minima and plan for future obstacle locations. The planner used is based on the windowed hierarchical cooperative A* algorithm (WHCA*), adapted to the cooperative hunting problem [15]. A single agent path planning algorithm is used to generate an optimal trajectory to the target, and its solution is then used as a heuristic to guide the search for multiple agents. The single agent path planning solution is a discretized 3D cost grid $J(t,x,y)$ in spacetime, defined over the following range.

$$t \in [t_s, t_f] \quad x \in [x_{min}, x_{max}] \quad y \in [y_{min}, y_{max}] \tag{24}$$

Each element in J represents the cost to the goal, and the optimal trajectory starting at t_s and ending at t_f is the set of points (t,x,y) which achieves the minimum cost. The agent steps through positions in time with action a assuming 4-connectivity in space. A cost j_a is incurred during state transitions, with a time cost j_t between timesteps and a stepping cost j_s between cells in space.

$$(x,y)_{t_{n+1}} \leftarrow (x,y)_{t_n} + a \tag{25}$$

$$a \in \{(-\Delta x,0),(\Delta x,0),(0,-\Delta y),(0,\Delta y),(0,0)\} \tag{26}$$

$$j_a = \begin{cases} j_t & \text{if } a = (0,0) \\ j_s + j_t & \text{otherwise} \end{cases} \tag{27}$$

Locations occupied by obstacles and target correspond to a cost of infinity and zero respectively. Future locations can be determined if the obstacle motion model and target motion model is known. For simplicity this paper uses a constant velocity model for both, and with this setup the cost is computed for every cell using dynamic programming.

Algorithm 1 3D Cost Grid Computation

```
Input: target and obstacle state at $t = t_s$
Output: 3D cost $J(t,x,y)$
1: Step the obstacle and target over time to $t = t_f$ using their motion models
2: Populate $J$ with 0 at the target and $\infty$ at the obstacles
3: Run wavefront propagation on $J(t = t_f, x, y)$ starting at the target location
4: Populate the remaining layers recursively
       $J(t_n,x,y) = \min_a (J(t_{n+1},x,y) + j_a)$    (28)
5: return $J(t,x,y)$
```

Using cost J, the optimal path for a single agent can be computed recursively by searching forward in time for the minimum cost. Given (t_n,x,y), the optimal next point is (t_{n+1},x^*,y^*).

$$(t_{n+1},x^*,y^*) = \underset{(t_{n+1},x',y')}{\text{argmin}}\; J\big(t_{n+1},x',y' \,|\, (x',y') = (x,y)+a\big) \tag{29}$$

Path planning is done sequentially for multiple agents using a ranking system. The agent with the highest rank plans first and adds its path to J, the next agent does the same and so on. As a result, subsequent agents obtain paths which do not conflict with their higher ranking agents. To reduce computation time, the agents only plan up to a time window t_w. The planning stages are summarized in Algorithm 2.

Algorithm 2 Multiagent Planner

```
Input: Cost $J$, agent positions, agent priority queue $Q$
Output: Set of paths $\mathcal{P}$
1: for agent $i$ in $Q$ do
2:     Search $J$ up to time $t_w$ to obtain path $p_i$ for agent $i$
3:     Add $p_i$ to $\mathcal{P}$
4:     Update $J$ by setting elements occupied by $p_i$ to $\infty$
5: end for
6: Reset $J$ back to its original before adding paths
7: return $\mathcal{P}$
```

Algorithm 2 is further modified by changing the paths added in step 3. The paths generated by the planner focuses on reaching the target rather than surrounding, and does not fully encapsulate the cooperative hunting objective. Therefore, an agent should not use the planner if their path is clear of obstacles and other agents. A path generated by the control strategy should be added in place of the planner path. The encirclement path can be obtained by stepping through the encirclement control strategy with the agent state space dynamics. Determining which path to add integrates the multi-agent planner with the cooperative hunting objective.

C. Collision Avoidance

An emergency collision avoidance rule is required to handle errors caused by modelling and tracking that may lead to unsafe situations during experiments. The flocking algorithm by Olfati-Saber [13] is adopted and customized to handle inter-agent and agent-obstacle collision avoidance. The original algorithm contains three components for agent alignment, obstacle avoidance, and target seeking. However, since alignment is unnecessary and target seeking is handled by the encirclement control, only the collision avoidance term $\mathbf{u}_i^\beta$ is used.

$$\mathbf{u}_i^\beta = \sum_{k \in N_i^\beta} c_1^\beta \phi_\beta \left(\left\| \hat{\mathbf{q}}_{i,k} - \mathbf{q}_i \right\|_\sigma \right) \hat{\mathbf{n}}_{i,k} + c_2^\beta b_{i,k} \left(\hat{\mathbf{p}}_{i,k} - \mathbf{p}_i \right) \tag{30}$$

$$\mathbf{q}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T \qquad \mathbf{p}_i = \begin{bmatrix} \dot{x}_i & \dot{y}_i \end{bmatrix}^T \tag{31}$$

The input $\mathbf{u}_i^\beta$ is the desired acceleration for agent i, and all other agents are treated as obstacles. Hence, N_i^β is agent i's set of neighbouring obstacles and agents. The collision avoidance method creates a virtual agent with state $(\hat{\mathbf{q}}_{i,k}, \hat{\mathbf{p}}_{i,k})$ at the boundary of each obstacle, and then uses repulsive potential function ϕ_β for collision avoidance with the virtual agent. The norm $\|\cdot\|_\sigma$ is a customized norm used for smoothing the potential function. The c_2^β term considers the relative velocity between the agent and obstacles, where $b_{i,k}$ is a heterogeneous adjacency term between agent i and obstacle k.

This collision avoidance approach was chosen over traditional potential field methods based on two key advantages. The velocity term allows for faster reaction with moving obstacles, as well as velocity damping if an agent approaches an obstacle too quickly. The second advantage is that the virtual agent creation allows for collision avoidance with environment boundaries, as the virtual agent generated moves in a straight line along the environment walls.

D. Integration

There are 3 components to the cooperative hunting algorithm: encirclement LQR control, multi-agent path planner, and collision avoidance. Only one component should be active at a time for each agent, as the objectives of each component conflict. LQR does not consider any collision avoidance and the planner does not enforce the surrounding objective. Since collision avoidance is top priority, if the force based term $\mathbf{u}_i^\beta$ from Equation (30) is active then the other two components should be inactive.

A model predictive control approach decides whether an agent performs encirclement or planning based on predicted future obstacle interaction. The agent predicts the future states for a time window t_w (same time as the planner) using the agent state space dynamics and the assumed constant velocity model for the target and obstacles. Agent i does not encounter obstacles if it can follow the encirclement strategy for duration t_w with zero $\mathbf{u}_i^\beta$ in Equation (30). In such cases the agent follows encirclement. Otherwise, the agent follows the trajectory obtained by the planner.

To track this trajectory, a cubic spline interpolation is used to connect the waypoints. Differentiating the spline twice generates reference positions, velocities, and accelerations which is tracked using LQR feedback control with the UAV state space dynamics. Together, the full cooperative hunting algorithm is outlined in Algorithm 3.

Algorithm 3 Cooperative Hunting

```
    Input: Cost $J$, agent, target, and obstacle states
    Output: Set of control inputs $\{\mathbf{u}_1, \mathbf{u}_2, \dots \mathbf{u}_N\}$
 1: Predict future states for time horizon $t_w$
 2: for agent $i$ from 1 to $N$ do
 3:     if collision avoidance active then
 4:         $\mathbf{u}_i \leftarrow$ Equation (30)
 5:     else if predicted obstacle interaction then
 6:         Add agent $i$ to priority queue $Q$
 7:     else
 8:         $\mathbf{u}_i \leftarrow$ Equations (19)
 9:     end if
10: end for
11: Add predicted states of all agents not in $Q$ to $J$
12: Run multiagent planning on agents in $Q$
13: $\mathcal{P} \leftarrow$ Algorithm 2
14: for agent $i$ in $Q$ do
15:     Track path $p_i$ using feedback control and obtain $\mathbf{u}_i$
16: end for
17: return $\{\mathbf{u}_1, \mathbf{u}_2, \dots \mathbf{u}_N\}$
```

IV. SIMULATION RESULTS

The proposed method is compared with optimal reciprocal collision avoidance (ORCA) [14]. ORCA is a well cited algorithm used for coordinating multiple agents to arrive at their desired locations with collision free trajectories. ORCA uses the concept of velocity obstacles, where the agent determines a set of velocities which will cause a collision with an obstacle if maintained through a time horizon. These sets of velocities are used as constraints for planning, and an optimization problem is formulated to generate desired velocities for each agent.

ORCA was selected due to its open source availability and ability to plan for multiple obstacles for which other methods lacked. To adapt ORCA for the cooperative hunting problem, the desired velocities are enforced using a feedback controller based on the UAV dynamics similar to the path following controller. The agents' desired goal locations are equally spaced points around the target.

Simulations were conducted using Python 3 in a 15 m by 15 m environment with 5 circular agents of radius 0.2 m in two different scenarios. The first scenario is a static obstacle rich environment and the second scenario is a moving obstacle environment. The target location is static and random. Circular obstacles are randomly generated in the environment with a radius between 0.3 m to 1 m, with varying obstacle numbers {10, 20, 30, 40, 50}. 100 episodes were conducted for each obstacle number using the proposed approach and ORCA. An example of the simulation environment with 40 static obstacles is shown in Figure 3.

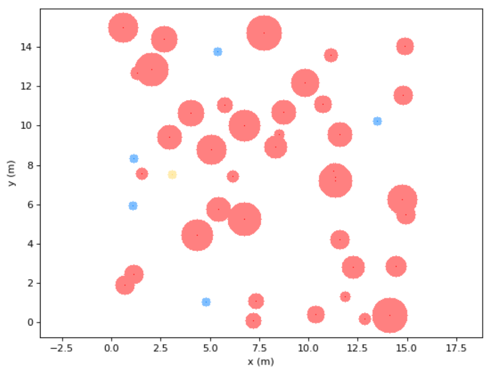

Fig. 3. Example simulation environment with 5 agents and 40 static obstacles. Agents shown in blue and target shown in yellow.

A. Obstacle Rich Environments

The number of successful captures within a time limit of 30 s was recorded, and Table I shows the success percentages. As the number of obstacles increases, the local minima problem becomes more prevalent, and the success rate drops to 37% for cooperative hunting using ORCA. With the proposed approach, most local minima are avoided and the success rate stays above 90%. Since the proposed approach predicts over a finite time window, the algorithm gets stuck in obstacle configurations containing deep local minima. This can be resolved by increasing the time window at the cost of additional computation. The time discretization size for prediction can also be increased, this allows for longer time windows at the cost of model resolution. From the simulations, a time window of 3 s with discretization size 0.2 s balances both accuracy and window length.

TABLE I

Success percentage in static obstacle environments

# of obstacles	10	20	30	40	50
MPC	100	100	98	95	94
ORCA	96	74	65	44	37

B. Moving Obstacle Environments

The obstacles have a random speed of up to 0.5 m/s, and the time limit was extended to 60 s. To measure performance, the capture time and distance traveled by all agents is recorded. The average capture time and path distance over all episodes is shown in Table II. The planner minimizes interaction with obstacles and thus the target is captured faster and with shorter paths compared to ORCA, where the agents constantly react to obstacles rather than following desirable trajectories. This improvement increases as more obstacles are introduced, from an average 0.7 s and 0.3 m improvement with 10 obstacles to a 5 s improvement and 3.1 m with 50 obstacles. Furthermore, the MPC approach has more consistent results compared to ORCA as indicated by the lower standard deviations. For example, ORCA spent up to 50 s in the most difficult environments, whereas the capture time using MPC always remained under 30 s.

TABLE II

Capture time and path distance in moving obstacle environments

# of obstacles	10	20	30	40	50
MPC time (s)	11.7 ± 4.0	13.3 ± 5.1	14.2 ± 4.7	16.1 ± 5.2	17.8 ± 5.4
ORCA time (s)	12.4 ± 6.8	14.5 ± 6.2	15.8 ± 7.9	19.4 ± 11.0	22.8 ± 12.2
MPC dist. (m)	16.3 ± 5.1	17.5 ± 5.4	18.8 ± 5.8	21.7 ± 6.1	22.6 ± 6.9
ORCA dist. (m)	16.6 ± 4.7	18.3 ± 5.9	20.6 ± 8.6	24.5 ± 10.1	25.7 ± 10.6

V. EXPERIMENTAL RESULTS

Two experiments were conducted in a 4 m by 2 m indoor lab environment. The agents are Crazyflie UAVs and the target is an iRobot create ground robot. A set of Optitrack motion capture cameras provide state information streamed through VPRN. Communication was handled by ROS Melodic through the `crazyflie_ros` package [21] and `create_autonomy` package for the iRobot. All experimental components are shown in Figure 4. PID controllers kept the UAVs at constant height and zero yaw to match the 2D setup. The desired target distance r_d was 0.5 m.

Due to downwash effects, the UAVs experience drift and tracking error. The desired radial distance errors were within 0.1 m and the desired angle errors were within 15°. The trajectory plots can be seen in Figure 5.

A. Moving Target

The first experiment verifies the proposed method for tracking the target. Five UAVs surround the ground robot in an empty environment, and the ground robot moves in a circle. The UAVs surround the target at $t = 10$ s and successfully follow its motion afterwards.

B. Static Obstacles

The second experiment verifies the multi-agent planning component of the algorithm. The environment consists of 4

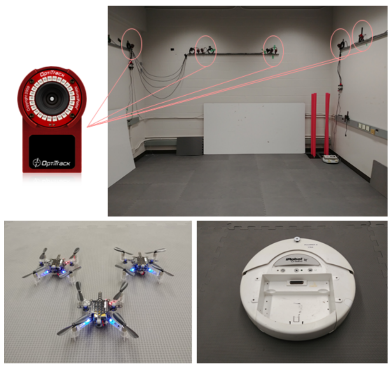

Fig. 4. Experimental Setup: Optitrack cameras in lab environment (top), Crazyflie UAVs (left), iRobot Create (right)

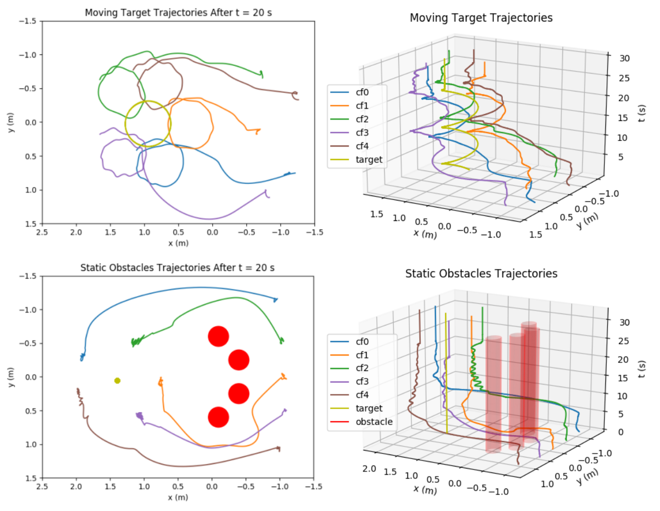

Fig. 5. Trajectory plots for the moving target experiment (top) and the static obstacle experiment (below).

static obstacles and a static target. All UAVs successfully pass the obstacles after $t = 10$ s and surround the target at $t = 15$ s. cf0 and cf4 only followed the encirclement strategy, as the prediction step determined no interactions with obstacles.

VI. CONCLUSION

A model predictive control algorithm was presented for cooperative hunting in obstacle rich environments and moving obstacle environments with consideration for UAV dynamics. The problem is formulated in the context of both control and planning. The proposed approach uses feedback linearized LQR control in polar coordinates for encirclement, dynamic programming based multi-agent planning for obstacle navigation, and flocking based collision avoidance. Integration of all three components is done by selecting the appropriate component to use based on predicted future obstacle interaction. The MPC algorithm was tested in simulation and verified in experiment using Crazyflie UAVs. Results show improved performance over optimal reciprocal collision avoidance (ORCA), where the agents more successfully avoid local minima and capture the target faster with shorter trajectories in the presence of moving obstacles.

References

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87. New York, NY, USA: ACM, 1987, pp. 25–34.

[2] A. Weitzenfeld, A. Vallesa, and H. Flores, "A biologically-inspired wolf pack multiple robot hunting model," in *2006 IEEE 3rd Latin American Robotics Symposium*, Oct 2006, pp. 120–127.

[3] C. Muro, R. Escobedo, L. Spector, and R. Coppinger, "Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations," *Behavioural Processes*, vol. 88, no. 3, pp. 192 – 197, Nov 2011.

[4] J. D. Madden, R. C. Arkin, and D. R. MacNulty, "Multi-robot system based on model of wolf hunting behavior to emulate wolf and elk interactions," in *2010 IEEE International Conference on Robotics and Biomimetics*, Dec 2010, pp. 1043–1050.

[5] A. T. Hafez, A. J. Marasco, S. N. Givigi, M. Iskandarani, S. Yousefi, and C. A. Rabbath, "Solving multi-uav dynamic encirclement via model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2251–2265, Nov 2015.

[6] H. Shen, N. Li, S. Rojas, and L. Zhang, "Multi-robot cooperative hunting," in *2016 International Conference on Collaboration Technologies and Systems (CTS)*, Oct 2016, pp. 349–353.

[7] A. Franchi, P. Stegagno, and G. Oriolo, "Decentralized multi-robot encirclement of a 3d target with guaranteed collision avoidance," *Autonomous Robots*, vol. 40, no. 2, pp. 245–265, Feb 2016.

[8] J. Cao, M. Li, Z. Wang, J. Li, and H. Wang, "Multi-robot target hunting based on dynamic adjustment auction algorithm," in *2016 IEEE International Conference on Mechatronics and Automation*, Aug 2016, pp. 211–216.

[9] J. Ma, W. Yao, W. Dai, H. Lu, J. Xiao, and Z. Zheng, "Cooperative encirclement control for a group of targets by decentralized robots with collision avoidance," in *2018 37th Chinese Control Conference (CCC)*, July 2018, pp. 6848–6853.

[10] Z. Wu, Z. Cao, Y. Yu, L. Pang, C. Zhou, and E. Chen, "A multi-robot cooperative hunting approach based on dynamic prediction of target motion," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2017, pp. 587–592.

[11] W. Yao, Z. Zeng, X. Wang, H. Lu, and Z. Zheng, "Distributed encirclement control with arbitrary spacing for multiple anonymous mobile robots," in *2017 36th Chinese Control Conference (CCC)*, July 2017, pp. 8800–8805.

[12] C. W. Warren, "Multiple robot path coordination using artificial potential fields," in *Proceedings., IEEE International Conference on Robotics and Automation*, May 1990, pp. 500–505 vol.1.

[13] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *Automatic Control, IEEE Transactions on*, vol. 51, pp. 401 – 420, 04 2006.

[14] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 3–19.

[15] D. Silver, "Cooperative pathfinding." in *Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2005*, 01 2005, pp. 117–122.

[16] K.-H. C. Wang and A. Botea, "Mapp: A scalable multi-agent path planning algorithm with tractability and completeness guarantees," *J. Artif. Int. Res.*, vol. 42, no. 1, pp. 55–90, Sep 2011.

[17] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 5954–5961.

[18] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, Aug 2018.

[19] "Crazyflie 2.1 | Bitcraze," https://www.bitcraze.io/crazyflie-2-1/, accessed: 2020-02-22.

[20] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Athena Scientific, 2000.

[21] W. Hönig and N. Ayanian, *Flying Multiple UAVs Using ROS*. Springer International Publishing, 2017, pp. 83–118.