2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

Distributed Model Predictive Formation Control of Robots with Sampled Trajectory Sharing in Cluttered Environments

Sami Satır [1,2], Yasin Furkan Aktaş [1], Simay Atasoy [1], Mert Ankaralı [2,4], Erol Şahin [3,4]

***Abstract*— In this paper, we propose a Model Predictive Control (MPC) based distributed formation control method for a multi-robot system (MRS) that would move them among dynamic obstacles to a desired goal position. Specifically, after formulating the formation control, as a distributed version of MPC, we propose and evaluate three information-sharing schemes within the MRS; namely sharing (i) positions, (ii) complete predicted trajectories, and (iii) exponentially-sampled predicted trajectories. Using a simplified kinematic model for robots, we conducted systematic simulation experiments in (a) scenarios, where the robots are instructed to switch places, as one of the most challenging forms of formation changes, and in (b) scenarios where robots are instructed to reach a goal, within environments containing dynamic obstacles. In a set of systematic experiments conducted in simulation and with mini quadcopters, we have shown that sharing of exponentially-sampled trajectories (as opposed to positions, or complete trajectories) among the robots provides near-optimal paths while decreasing the required computation cost and communication bandwidth. Surprisingly, in the presence of noise, sharing exponentially-sampled trajectories among the robots decreased the variance in the final paths. The proposed method is demonstrated on a group of Crazyflie quadcopters.**

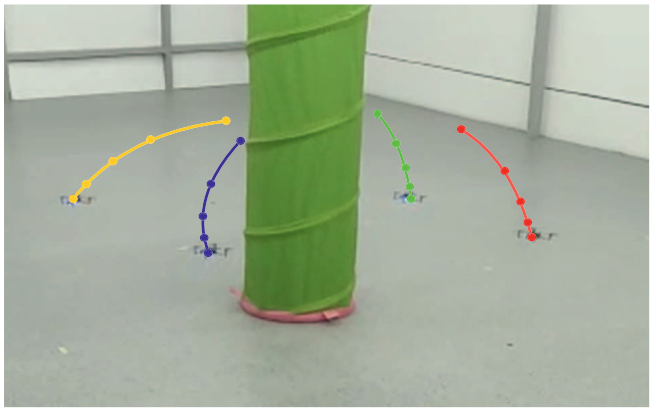

Fig. 1. Illustration of the MPC-based distributed formation control method in which robots share their exponentially-sampled trajectories with each other that would move them among dynamic obstacles to a desired goal position. In a set of systematic experiments conducted in simulation and with mini quadcopters, we have shown that sharing of exponentially-sampled trajectories (as opposed to positions, or complete trajectories) among the robots provides near-optimal paths while decreasing the required computation cost and communication bandwidth. Surprisingly, in the presence of noise, sharing exponentially-sampled trajectories among the robots decreased the variance in the final paths. A supplementary video of our experiments is found at https://youtu.be/rWu8gmLtf3g

I. INTRODUCTION

In MRS, formation control refers to the control of the relative positions and orientations of the robots such that the group maintains a desired set of relative distances among robots [1]. It improves the efficiency, speed, and safety in applications, such as creating aerial images of large areas with high spatial resolution [2], increasing the collective sensing capability of interconnected sensors for target tracking [3], and approximating the behavior of an antenna that is orders of magnitude larger than the size of each robot, thereby improving the sensitivity of the overall system [4].

In this paper, we propose an MPC-based distributed formation control method for a MRS that would move them among dynamic obstacles to a desired goal position. Upon establishing the formation control as a distributed variant of MPC, we proceed to suggest and assess three information-sharing schemes, which involve sharing positions, complete predicted trajectories, and sampled predicted trajectories.

The remainder of this paper is organized as follows. Section II provides a comprehensive survey of related literature. Section III summarizes the proposed methodology regarding the DMPC (Distributed MPC) cost functions and constraints. Section IV contains results obtained from simulations and experiments with Crazyflie 2.1 quadcopters (Bitcraze, Sweden), a discussion on performance. Finally, section V concludes the paper and proposes future directions for further improvements.

II. RELATED WORK

Recent studies indicate that predicting the future behaviors of robots in an iterative approach can increase the stability of aerial swarms [5]–[8]. Several MPC methods are adopted for MRS and swarm robot applications. In [7], linear MPC is implemented for distributed collision avoidance of multi-quadrotors sharing the same workspace. The proposed approach introduces a pipeline for control that consists of a non-linear state controller, an MPC trajectory tracker, and a technique for collision avoidance based on priorities and trajectory prediction to avoid collisions between vehicles. Cheng et al. [9] propose a method that combines MPC with the Optimal Reciprocal Collision Avoidance (ORCA) algorithm for MRS. ORCA algorithm generates a set of collision-free velocities for each robot and these permitted velocities serve as constraints in the MPC problem. In [10], the authors present trajectory tracking and collision avoidance

*This work supported by Aselsan Inc.

[1] ASELSAN Research Center, 06200 Ankara, Turkey
[2] Department of Electrical and Electronics Engineering, Middle East Technical University, 06800 Ankara, Turkey
[3] Department of Computer Engineering, Middle East Technical University, 06800 Ankara, Turkey
[4] Center for Robotics & AI, Middle East Technical University, 06800 Ankara, Turkey

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10341414

as a single optimization problem. In this approach, collision avoidance has been defined as a cost function rather than a constraint. Mehrez et al. [11] proposes a distributed MPC approach for mobile robots by projecting the predicted states onto an occupancy grid to reduce the communication overhead. The proposed approach reduces the communication requirements while still allowing the optimization process to use sensitive information. In [12], the authors focus on the collaborative transportation of a payload with multiple quadrotor vehicles. Both distributed and centralized MPC frameworks are developed and a state-specific linearization is applied to the implemented nonlinear cost function. Also, the research proposes to share predicted control inputs instead of predicted states. In [8], DMPC formulation allows a scalable cohesive flight of aerial swarms in cluttered environments. The presented MPC scheme uses an objective function to incorporate the principles of potential field models in order to enhance the speed, order and safety of the swarm.

To avoid collisions while navigating in a multi-agent structure, the reviewed studies above [7]–[12] make use of information sharing between agents. The shared information mainly constitutes the uniformly sampled future states (velocity and position) of the neighboring robots. Even though predictions regarding the future states of the neighboring robots are necessary to avoid possible collisions, it results in a high communication load. In order to avoid this problem, we propose an exponential sampling approach. With this approach, we aim to decrease the high communication load between robots.

In this work, we propose a framework addressing formation control and collision avoidance problems using distributed and nonlinear MPC methods. Our proposed method examines the cost functions, constraints, and the necessary information to be shared. Additionally, we define a cost function for formation control, enabling robots to perform formation movements using DMPC in a cluttered environment.

III. METHODOLOGY

Consider a group of M mobile robots, whose states are defined as $q_i = [x_i \;\; y_i \;\; \theta_i]^T$ where x_i and y_i is the position and θ_i is the heading of the robot i. The control inputs are defined as $u_i = [v_i \;\; \phi_i \;\; \omega_i]^T$ where v_i is velocity magnitude, ϕ_i is the orientation of the velocity vector from the heading, and ω_i denotes the angular velocity. Fig. 2 illustrates the states and control inputs of the robot kinematic model.

We define the formation control problem as the computation of collision-free paths, for a group of robots with initial states $q^0 = \{q_i^0\}_{i:0,M}$ to a set of desired states $q^g = \{q_i^{goal}\}_{i:0,M}$ in a distributed manner. In our definition, the trajectories are defined as a sequence of waypoints $\tau_i(k) = \{q_i(n|k)\}_{n:0,N}$ for the robot at time k, which can then be fed to the controllers of the robots. We define a sampled version of the trajectory as $\hat{\tau}_i(k) = \{q_i(n|k)\}_{n:0,1,\ldots,2^a<N}$ for the robots, which can be shared with other robots.

At time k, given the state vector $q_i(k)$ and the prediction horizon N, the predicted trajectory for the robot is represented as $q_i(1|k), q_i(2|k), ..., q_i(n|k), ...q_i(N|k)$, where $q_i(n|k)$ is the predicted state at time $k+n$.

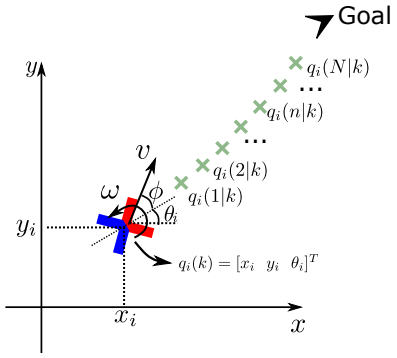

Fig. 2. Simplified kinematic model. The robot is depicted as a half-blue and half-red cross. The red side represents the front of the robot. The green crosses represent the future predictions of MPC. The black arrow shows the goal location and the final heading. The predicted trajectory at time k indicated with a green crosses

We define the objective cost function consisting of three parts: navigation cost, control effort, and formation control as:

$$\min_{q,u} J_{i,\text{total}} = J_{i,\text{nav}} + J_{i,\text{eff}} + J_{i,\text{form}} \tag{1}$$

subject to

$$q_i(n|k) = f(q_i(n-1|k), u_i(n|k)), \qquad n = 1, ..., N \tag{2a}$$
$$b_{min} \leq C q_i(n|k) \leq b_{max}, \qquad n = 0, ..., N \tag{2b}$$
$$u_{min} \leq u_i(n|k) \leq u_{max}, \qquad n = 0, ..., N \tag{2c}$$

where Eqns. 2a, 2b, and 2c define the system dynamics, map margins, and input constraints, respectively. b_{min} and b_{max} are vectors defining the minimum and maximum values of the state constraints, while u_{min} and u_{max} are vectors defining the minimum and maximum values of the control inputs. C is a diagonal matrix that defines which states have constraints.

A. Simplified Kinematic Model

We used a single-integrator model, widely used in several multi-robot coordination control problems such as consensus and formation control [2], [13]–[16], to characterize the motion of the robot as:

$$q_i(k+1) = f(q_i(k), u_i(k)) \quad \text{for} \quad i = 1, 2, \ldots, M$$

$$\begin{bmatrix} x_i(k+1) \\ y_i(k+1) \\ \theta_i(k+1) \end{bmatrix} = \begin{bmatrix} x_i(k) \\ y_i(k) \\ \theta_i(k) \end{bmatrix} + \begin{bmatrix} v_i(k)\cos(\theta_i(k)+\phi_i(k)) \\ v_i(k)\sin(\theta_i(k)+\phi_i(k)) \\ \omega_i(k) \end{bmatrix} \Delta T$$

The single integrator model can generate velocity vector inputs that may be beyond the limits of the robot. In order to avoid this, the control effort cost function is implemented to restrict a drastic change in the velocity vector. The internal controllers of the robot use high-level velocity inputs to drive the actuators of the robot.

B. Navigation of Single Robot with Model Predictive Control

The cost function navigating a single robot from the start to the goal point is given as

$$J_{i,\text{nav}} = \sum_{n=0}^{N-1} \left(\parallel q_i(n|k) - q_i^{\text{goal}} \parallel_Q^2 + \parallel u_i(n|k) \parallel_R^2 \right) + \parallel q_i(N|k) - q_i^{\text{goal}} \parallel_{Q_f}^2, \quad (3)$$

where N is the prediction horizon, Q is the state cost matrix, R is the input cost matrix, and Q_f is the final state cost matrix. q_i^{goal} is the position and heading of the given goal point for robot i, $u_i(n|k)$ is the predicted control input at time $k+n$. For a single robot and an environment with no obstacles, optimizing $J_{i,\text{nav}}$ to compute control actions is sufficient for navigating the robot to the goal point.

The control effort aims to reduce the energy consumption of the control commands and hence minimizes the input effort. To achieve this, it calculates the rate of change of the input commands and penalizes abrupt acceleration and deceleration. It also increases the method's usability as a high-level controller since it prevents drastic changes in the reference control inputs. The control effort is defined as the difference equation of consecutive control inputs $\Delta u_i(n|k) = u_i(n|k) - u_i(n-1|k)$. The control effort cost penalizes the changes in the control commands, which are $[v_i \quad \phi_i \quad \omega_i]$.

The cost function for control effort can be defined as

$$J_{i,\text{eff}} = \sum_{n=1}^{N-1} \parallel \Delta u_i(n|k) \parallel_{R_{\text{eff}}}^2 \quad (4)$$

C. Obstacle Avoidance Constraint

To avoid a potential collision with an obstacle, we introduce additional constraints to the MPC framework. This paper assumes that both the robots and obstacles are circular objects. The actual radius of the robot is denoted as r_{robot}. Additional distance d_{safety} is added to the robot in order to increase safety. For the constraint, r_{safety} is defined as the summation of the robot radius r_{robot} and safety distance d_{safety}. r_{obs} is radius of the obstacle and q_{obs} is position of the obstacle.

The Euclidean distance between the center of the obstacle q_{obs} and the center of the robot i (and its predictions) can be denoted as $||q_i(n|k) - q_{\text{obs}}||$ at time instant k. For all the predictions and current state of each robot i, the Euclidean distance should be larger than $r_{\text{safety}} + r_{\text{obs}}$. The obstacle avoidance constraint is defined as,

$$(r_{\text{safety}} + r_{\text{obs}}) - ||q_i(n|k) - q_{\text{obs}}|| \leq 0 \quad (5)$$

Fig. 3 shows an environment with an obstacle and the defined radii.

For a dynamic obstacle, the center of the obstacle becomes the function of time instant k, and is labeled as $q_{\text{obs}}(k)$. The dynamic obstacles are modelled like as static obstacles whose positions are updated at each time instant k. By updating the q_{obs} in Eqn. 5 for each time instant k dynamic obstacle problem can be solved with the same cost function in Eqn. 1.

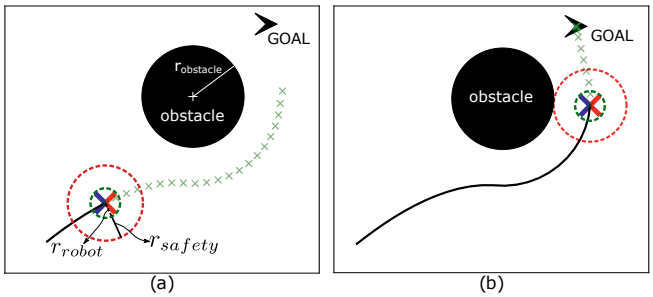

Fig. 3. Robot is approaching goal under the obstacle avoidance constraint. (a) The robot generates its state predictions as indicated with green crosses. (b) The trajectory followed by the robot is given with a solid black line.

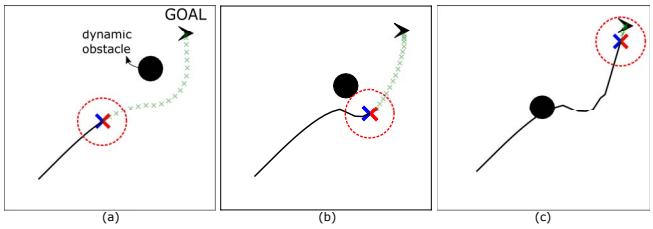

Fig. 4. Robot is approaching the given goal point by avoiding dynamic obstacle coming towards it (a) at time t = 1.0 s (b) t = 2.6 s (c) t = 4.0 s (d) t=6.0 s

D. Robots' Reciprocal Avoidance with Position Sharing Method

In this method, each robot has only the position information of other robots in the environment. As in the case of the dynamic obstacle, each robot predicts its own state by adding the current positions of other robots to its optimization problem as a constraint.

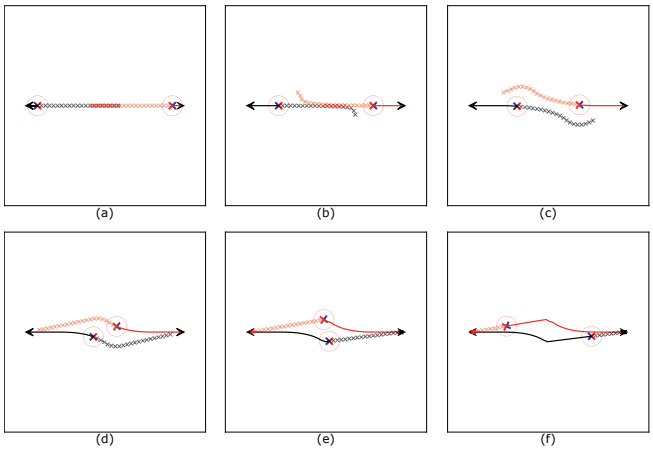

Fig. 5. Reciprocal avoidance of two robots with position sharing method in position swap scenario (a) at time t = 0.8 s (b) t = 1.8 s (c) t = 2.6 s (d) t = 3.6 s (e) t = 4.4 s (f) t = 6.4 s

In Eqn. 6, reciprocal avoidance constraint is defined for robot i. Given constraint should be applied for each robot j in the system. Note that, at instant k, the state vector $q_j(k)$ is available for each robot j with its own measurement. This measurement is published to other robots.

$$(2r_{\text{safety}}) - ||q_i(n|k) - q_j(k)|| \leq 0, \quad n = 0, 1, ..., N \quad (6)$$

E. Robots' Reciprocal Avoidance with Trajectory Sharing Method

If a robot can differentiate between other robots and dynamic obstacles, and also has the ability to recognize that these robots utilize their own predictive models, it is evident that its own path-planning capabilities would improve. However, this method incurs the additional cost of increased information sharing. In the prior approach, each robot solely requires positional information (in the form of two floating-point values representing x and y coordinates) for all other robots ($M-1$ in total), necessitating the sharing of $2 \cdot (M-1)$ floating-point values. In contrast, the current method mandates the sharing of all predicted positional information of other robots, necessitating the sharing of $2 \cdot (M-1) \cdot N$ floating-point values. The reciprocal avoidance constraint should be modified as

$$(2r_{\text{safety}}) - ||q_i(n|k) - q_j(n|k)|| \leq 0, \quad n = 0, 1, ..., N \quad (7)$$

At the instant k, the state vector $q_j(n|k)$ is calculated on each robot j and published to other robots. With the modified constraint, robot i now incorporates the prediction of its own states, ensuring non-collision with the predicted states of other robots.

In addition to the method that shares complete trajectories, a sampling-based method that reduces the computational load and communication load but still improves performance is presented in the following subsection.

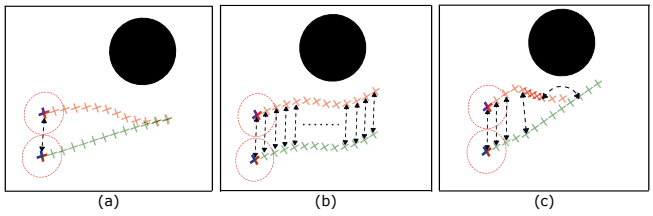

Fig. 6. Reciprocal avoidance of two robots with (a) position sharing (b) trajectory sharing (c) sampled trajectory sharing

1) Sharing Complete Trajectories: In this method, robots share their predicted state sequences, $\tau_i(k) = \{q_i(n|k)\}_{n:0,N}$, with their neighbors to solve the next optimization problem simultaneously. Although sharing all predictions increases the communication load of the system and the total calculation time, it has shortens the traveled path and reduced the control effort.

In Fig. 6(b), it can be observed that by sharing the entire predicted trajectory, the trajectory of the lower robot adjusted based on the received information.

In Fig. 5, it can be observed that the robots plan collision-free trajectories, $\tau_i(k)$, based only on the instantaneous position of the other robot, $q_j(k)$, while in Fig. 7, each robot plans its own trajectory, $\tau_i(k)$, based on the trajectory information shared with it, $\tau_j(k)$, ensuring that its own trajectory does not collide with the trajectory of the other robots.

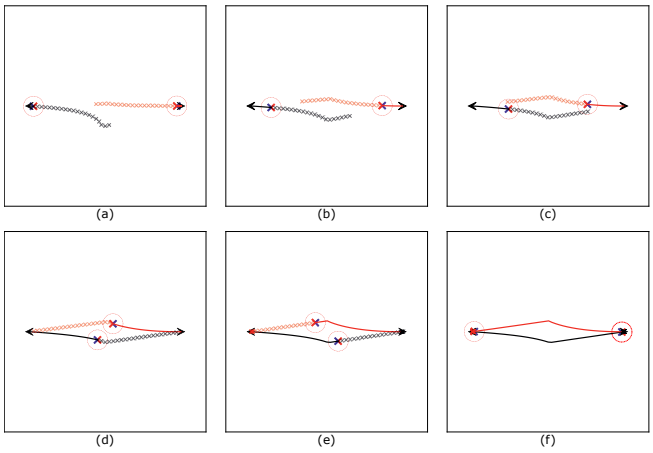

Fig. 7. Reciprocal avoidance of two robots with prediction sharing method in position swap scenario (a) at time t = 0.6 s (b) t = 1.4 s (c) t = 2.2 s (d) t = 3.8 s (e) t = 4.4 s (f) t = 6.8 s

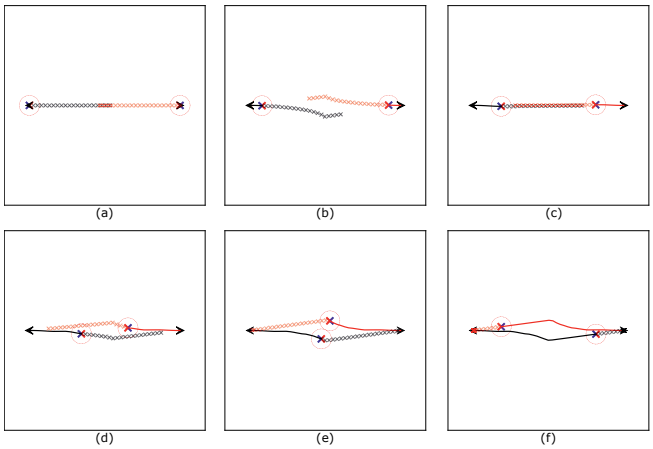

Fig. 8. Reciprocal avoidance of two robots with sampled number of prediction sharing method in position swap scenario at times (a) t = 0.4 s (b) t = 1.0 s (c) t = 1.8 s (d) t = 3.0 s (e) t = 4.0 s (f) t = 6.6 s

2) Sharing Sampled Trajectories: Sharing the robot's predicted trajectory through sampling can significantly reduce both communication and computation loads. While reducing the loads, it is important to perform the sampling in a way that minimizes the impact on collision avoidance and trajectory smoothness. Due to the fact that the closer predictions of the robot contain more critical information regarding collision avoidance and trajectory smoothness, the sampling method should be chosen accordingly. Hence, the sampling was performed by defining an exponential function so that samples are more frequent from close predictions and sparser from distant ones.

The sampled trajectory is defined as $\hat{\tau}_i(k) = \{q_i(n|k)\}_{n:0,1,...,2^a<N}$ for the robot i which can be shared with other robots.

F. Formation Control

We model a formation composed of M robots labeled as $i \in \mathcal{V} = 1, ..., M$ with a undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V}$ represents the robots, and the edge set

$\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ contains the pairs of robots (i, j) for which robot i and robot j are neighbors. We construct the graph and define the neighbors for each robot with the Delaunay triangulation algorithm [17], [18]. Since the number of neighbors for each robot will not increase with the total number of robots but will typically remain limited to a few (typically between 2-4), once the graph is constructed and the neighbors are determined, the subsequent calculations will be independent of the total number of robots. As a result, we can claim that the algorithm is scalable.

To incorporate the formation behavior into the system, it is necessary to add a new cost function to the MPC. This function should ensure that the distance between two neighboring robots namely i and j is equal to the desired formation distance $\bar{d}(i, j)$.

$$J_{i,\text{form}} = \sum_{n=0}^{N-1} \sum_{j=0}^{M_i} \Big(S_0(\| q_i(n|k) - q_j(n|k) \| - \bar{d}(i,j))^2 + S_1(\theta_i - \theta_j)^2 \Big) \tag{8}$$

M_i describes the number of neighbors of the i^{th} robot. $\bar{d}(i, j)$ means the given desired distance between the robots i and j. A quadratic cost is described as the difference between the measured distance and desired distance. In addition, if the heading alignment is required, a quadratic cost is added for the heading difference of robots. S_0 and S_1 are the weights for the formation cost and heading alignment cost respectively.

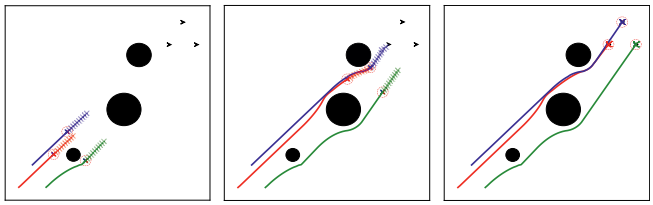

Fig. 9. Three robots moving to goal location without formation at time (a) t = 4.0 s (b) t = 12.0 s (c) t = 20.4 s

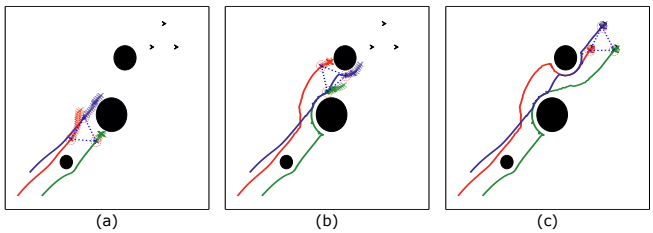

Fig. 10. Three robots moving to goal location with formation at time (a) t = 8.0 s (b) t = 16.0 s (c) t = 28.0 s

Algorithm 1 describes how the DMPC code operates independently on each robot. According to the selected method, each robot receives state or prediction information from its neighbors, updates the constraints of its own Optimal Control Problem (OCP), applies the first step of the optimal control command obtained from the solved OCP, and simultaneously shares its own position or prediction information with its neighbors. This iteration continues for each robot until it reaches its target.

Algorithm 1 Distributed Model Predictive Control

Input: $q_i^0, q_i^{\text{goal}}, M_i, \bar{d}(i,j)$
Output: $q_i(k), q_i(n|k)$

Initialize MPC Parameters: $Q, Q_f, R, R_{\text{eff}}, N, T$
Define the OCP in Eqn. 1 with Eqns. 3,4,8
while q_i^{goal} **not** reached **do**
  for each robot j in M_i **do**
    $constraint \leftarrow q_j(k)$ (position sharing) or
    $constraint \leftarrow \tau_j(k)$ (predicted trajectory) or
    $constraint \leftarrow \hat{\tau}_i(k))$ (sampled predicted traj.)
  end for ▷ Communicate with all neighbors
  Update the constraints as given in Eqns. 6 or 7
  $u_i^* \leftarrow$ Solve OCP with given constraints
  Apply the first step of calculated u^*
  Publish your state $q_i(k)$ or predictions $q_i(n|k)$ or its sampled version $\hat{q}_i(n|k)$
end while

IV. EXPERIMENTAL WORK

In order to evaluate the performance of the proposed model predictive control approach for formation flight, we implemented the proposed control system on mini quadrotors Crazyflie 2.1 and tested the formation control performance in a variety of scenarios, including static and dynamic obstacles, as well as different numbers of quadcopters.

The algorithm was implemented in Python, using ROS to manage the multi-quadrotor system and casADi [19] as the nonlinear optimizer. Simulations and experiments are performed on a laptop with Intel i7 2.4 GHz processor running Ubuntu 20.04.

A. Simulation Results

Fig. 5, 7 and 8 illustrate the paths followed by robots in position swap scenarios with the proposed methods. Figure 12 presents the confidence plots derived from Monte Carlo trials performed under the process noise (SNR $= 1$). It can be observed that the sampled prediction-sharing method reduces the average path length taken and yields more stable results. Two important observations can be made: First, the sampled trajectory method generates paths that are almost as short as the complete trajectory method. This indicates that sampling does not move us away from optimality. Second, the paths generated with sampled trajectory sharing created less variance than the paths generated with complete trajectory sharing, especially under noise. This is an interesting and unexpected result that requires further analysis. One possible explanation is that exponential sampling implicitly moves the problem into the exponential decay of rewards in time within the reinforcement learning paradigm.

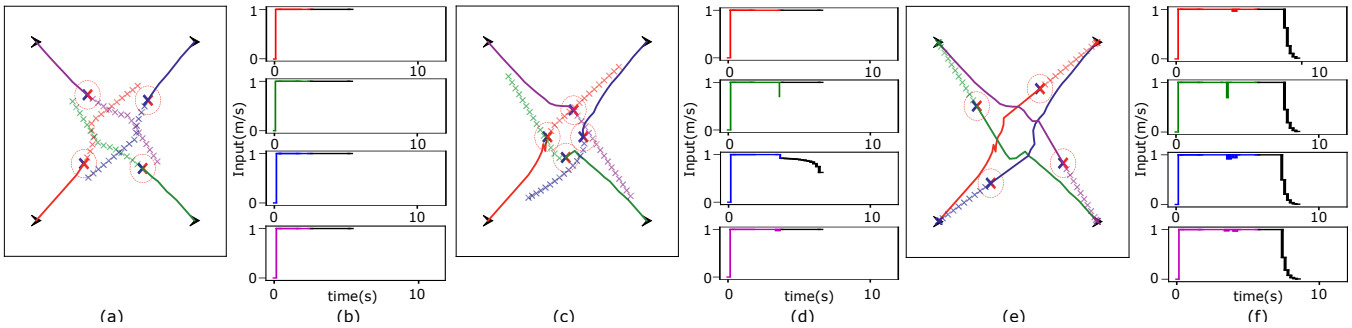

Fig. 11. Robot positions and applied inputs at time (a, b) t = 2.6 s (c, d) t = 3.8 s (e, f) t = 5.8 s. The colored lines in the plots indicate the applied velocities to the robots, whereas the dark lines indicate the calculated future predictions of the MPC.

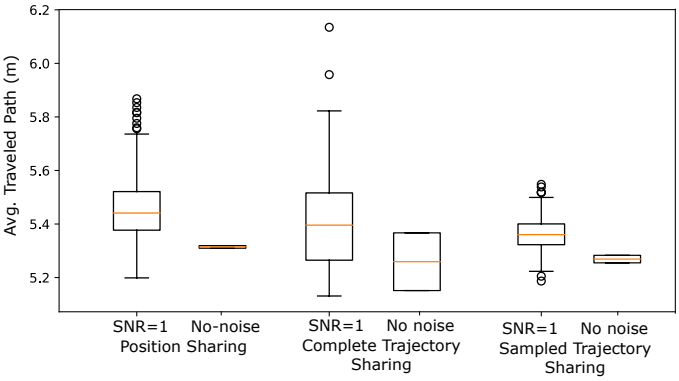

Fig. 12. The average traveled path for each sharing method. The box extends from the lower to upper quartile values of the data, with a line at the median. The whiskers extend from the box to show the range of the data. Outlier points are those past the end of the whiskers.

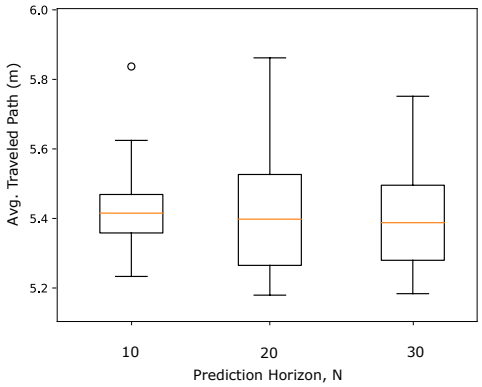

Fig. 13. The average traveled path for prediction horizon values N=10, N=20, N=30

Fig. 13 illustrates the results of the systematic Monte Carlo trials where we analyzed the effects of prediction horizon length on the system performance. The results indicate that the MPC framework's prediction horizon does not significantly impact the traveled path, such that we don't observe either improvement or degeneration. In the MPC literature (in various application domains), it is a well-known result that increasing the prediction horizon almost indeed improves the performance of the algorithms (with the additional cost of computational complexity), but for

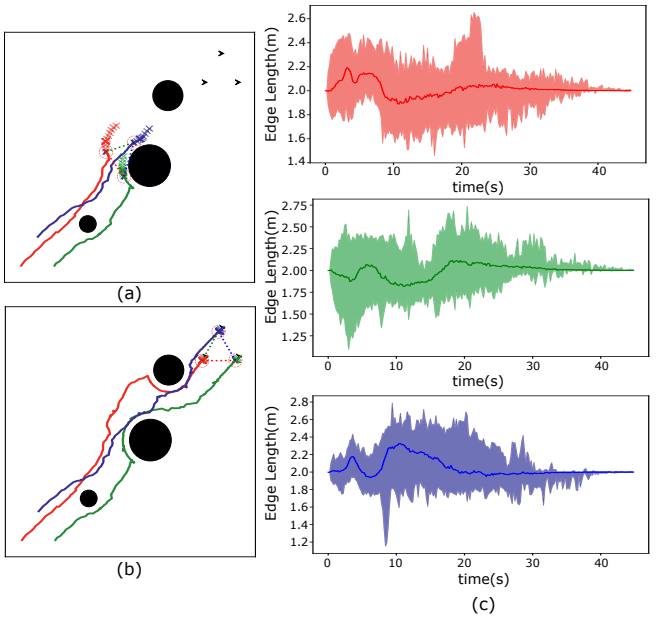

Fig. 14. Three robots navigating to goal locations while keeping the formation under process noise at (a) t = 11.0 s (b) t = 30 s. (c) The plots indicate the inter-agent distance averages (solid lines) and the ranges (shaded region). The colors correspond to the edges indicated with dashed lines.

cases where system dynamics and constraints do not change dynamically. However, in our scenarios, dynamic obstacles exist, but we treat them as static obstacles instantaneously. We believe that due to the highly dynamic behavior of the environment, we did not observe an improvement in the performance with the prediction horizon; however, positively, the results are also not sensitive to the selection of the prediction horizon, which indicates the robustness of the approach.

Fig. 14(a),(b) illustrate the formation motion under the process noise (SNR = 1). Fig. 14(c) presents the variation of distance between the robots over time. The desired distance for each edge of the equilateral triangle formation is defined as 2 meters. The maximum and minimum values obtained from Monte Carlo trials are shown with shaded regions in Fig. 14(c).

Fig. 15 shows the simulation results of the formation

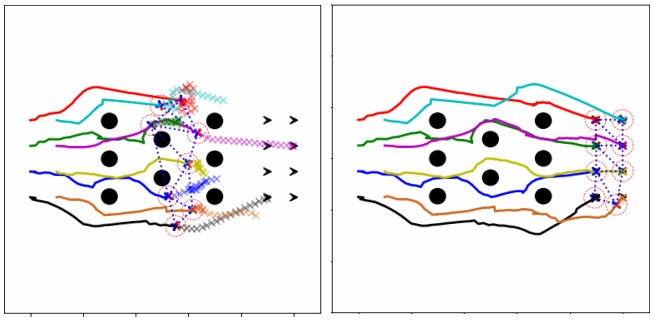

Fig. 15. Eight robots navigating to goal locations while keeping the formation in a forest-like cluttered environment at time (a) t = 15.0 s (b) 25.0 s

movement of 8 robots in a forest-like cluttered environment.

B. Crazyflie Experiments

Crazyswarm [20] offers a centralized system architecture to control and coordinate multiple Crazyflie quadcopters. The framework relies on a motion capture system to track the position and orientation of each quadcopter, and all decisions regarding movement and coordination are made by a central controller. In order to use the architecture provided by Crazyswarm on a different quadcopter platform for real-world applications, the system architecture must be adjusted to a distributed framework.

To convert Crazyswarm architecture from a centralized to a distributed approach, it would be necessary to design a new software architecture that enables each quadcopter to communicate with its neighbors. By using ROS, it would be possible to create a network of nodes that communicate with each other over a wireless network, allowing the quadcopters to exchange information in real-time. To achieve distributed control, each Crazyflie starts as a separate and simultaneous ROS node on the main computer. Crazyflies exchange information about their position and predictions via ROS messages.

V. CONCLUSIONS

In this paper, we propose a Model Predictive Control (MPC) based distributed formation control method for a MRS that would move them among dynamic obstacles to a desired goal position. Specifically, after formulating the formation control, as a distributed version of MPC, we propose and evaluate three information-sharing schemes within the MRS; namely sharing (i) positions, (ii) complete predicted trajectories, and (iii) exponentially-sampled predicted trajectories.

In a set of systematic experiments conducted in simulation and with mini quadcopters, we have shown that sharing of exponentially-sampled trajectories (as opposed to positions, or complete trajectories) among the robots provides near-optimal paths while decreasing the required computation cost and communication bandwidth. Surprisingly, in the presence of noise, sharing exponentially-sampled trajectories among the robots decreased the variance in the final paths. As future work, we plan to conduct detailed analyses to investigate how the average traveled distance and computation time vary with the prediction horizon and noise levels.

REFERENCES

[1] M. A. Kamel, X. Yu, and Y. Zhang, "Formation control and coordination of multiple unmanned ground vehicles in normal and faulty situations: A review," *Annual Reviews in Control*, vol. 49, pp. 128–144, 2020.

[2] M. de Queiroz, X. Cai, and M. Feemster, *Formation Control of Multi-Agent Systems: A Graph Rigidity Approach*. 01 2019.

[3] S. Martínez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, pp. 661–668, 04 2006.

[4] B. Anderson, B. Fidan, C. Yu, and D. Walle, *UAV Formation Control: Theory and Application*, vol. 371, pp. 15–33. 12 2007.

[5] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[6] M. Jacquet and A. Franchi, "Enforcing vision-based localization using perception constrained n-mpc for multi-rotor aerial vehicles," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1818–1824, 2022.

[7] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6753–6760, 2018.

[8] E. Soria, F. Schiano, and D. Floreano, "Predictive control of aerial swarms in cluttered environments," *Nature Machine Intelligence*, vol. 3, 06 2021.

[9] H. Cheng, Q. Zhu, Z. Liu, T. Xu, and L. Lin, "Decentralized navigation of multiple agents based on orca and model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3446–3451, 2017.

[10] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 236–243, 2017.

[11] M. W. Mehrez, T. Sprodowski, K. Worthmann, G. K. Mann, R. G. Gosine, J. K. Sagawa, and J. Pannek, "Occupancy grid based distributed mpc for mobile robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4842–4847, 2017.

[12] J. Wehbeh, S. Rahman, and I. Sharf, "Distributed model predictive control for uavs collaborative payload transport," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11666–11672, 2020.

[13] M. M. Asadi, A. Ajorlou, and A. Aghdam, "Distributed control of a network of single integrators with limited angular fields of view," *Automatica*, vol. 63, pp. 187–197, 01 2016.

[14] Z. Cheng, M.-C. Fan, and H.-T. Zhang, "Distributed mpc based consensus for single-integrator multi-agent systems," *ISA Transactions*, vol. 58, 04 2015.

[15] X. Xu, L. Liu, and G. Feng, "Consensus of single integrator multi-agent systems with distributed infinite transmission delays," pp. 1653–1658, 06 2018.

[16] S. Zhao and Z. Sun, "Defend the practicality of single-integrator models in multi-robot coordination control," 03 2017.

[17] K. Mehlhorn and S. Näher, "Implementation of a sweep line algorithm for the straight line segment intersection problem," 11 1994.

[18] F. Aurenhammer and O. Schwarzkopf, "A simple on-line randomized incremental algorithm for computing higher order voronoi diagrams," pp. 142–151, 1991.

[19] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[20] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," pp. 3299–3304, 2017.