Decentralized Path Planning for Multi-Objective Robot Swarm System

Zhuokai Wu

1 Xi'an Gaixin No.1 Middle School, Xi'an, Shaanxi, 710000, China
*Corresponding author’s e-mail: alinda@wedo365.com

Abstract. The multi-robot path planning aims to explore a set of non-colliding paths with the shortest sum of lengths for multiple robots. The most popular approach is to artificially decompose the map into discrete small grids before applying heuristic algorithms. To solve the path planning in continuous environments, we propose a decentralized two-stage algorithm to solve the path-planning problem, where the obstacle and inter-robot collisions are both considered. In the first stage, an obstacle-avoidance path-planning problem is mathematically developed by minimizing the travel length of each robot. Specifically, the obstacle-avoidance trajectories are generated by approximating the obstacles as convex-concave constraints. In the second stage, with the given trajectories, we formulate a quadratic programming (QP) problem for velocity control using the control barrier and Lyapunov function (CBF-CLF). In this way, the multi-robot collision avoidance as well as time efficiency are satisfied by adapting the velocities of robots. In sharp contrast to the conventional heuristic methods, path length, smoothness and safety are fully considered by mathematically formulating the optimization problems in continuous environments. Extensive experiments as well as computer simulations are conducted to validate the effectiveness of the proposed path-planning algorithm.

1. Introduction

In the early days, mobile robots were used only to help with production in factories. However, with the development of science and technology, such mobile robots are gradually applied to disaster relief, education, medical care, agricultural production, information and communication fields. When performing a locating pathfinder mission, the robot is equipped with many advanced equipments, the main functions of these devices include mapping of the environment, obstructive detection, control the motion and use navigation techniques to avoid obstacles. The most important function of any navigation for a robot is to find and plan a safe route from start position to terminal point.

The most common approach for path planning is to divide the continuous environment into a discrete gridded space. Considerable investigations have been devoted into this scheme. For instance, the genetic algorithm (GA) [1] and particle swarm optimization (PSO) [2] were developed to find the suitable trajectory with the shortest length in a discrete gridded environment. However, the computational complexity as well as time cost may become extremely high due to the NP-hard nature of this approach [3]. Furthermore, the gridded environments may result in unrealistic trajectories, where sharp turns are inevitable.

To address these challenges, an enhanced genetic algorithm (EGA) was developed in [4], where five genetic operators were designed to improved the performance at each iteration in terms of the predefined objective function. In [5], an informative path-planning problem was derived and resolved using an evolutionary strategy. Unfortunately, the aforementioned algorithms are not suitable for dynamic scenarios since the training process is highly depended. In [6], a decentralized method using
velocity control was derived for multi-vehicle traffic networks, where the path-planning problem was only investigated in two-lane scenarios. The control barrier function (CBF) is utilized to guarantee the safe distance among vehicles.

The existing methods for multi-robot path planning can be classified into two categories, namely centralized and decoupled methods. For the centralized approaches, the constraints and trajectory generation for all robots are combined into one optimization problem such that the solutions are near to the global optimum. However, the time complexity may become extremely high due to the combination of multiple constraints and configurations. Therefore, the centralized approaches usually cannot adapt dynamic environments. In contrast, decoupled methods first derive an individual trajectory for each robot and then resolve the possible conflicts of the generated trajectories. A typical decoupled approach was proposed in [7], where the wait-strategy is adopted by predefining the priority of robots. After the trajectories are individually generated, the robot with lower priority is ordered to wait temporarily if two robots are collision-prone. Despite the collisions are avoided, this wait-strategy is not time-efficient, neither feasible in practice.

In this work, we propose a decoupled approach for multi-robot path planning in which two stages are adopted. In the first stage, obstacle-avoidance trajectories are generated for the individual robots. In contrast to the heuristic algorithms based on the gridded environments, such as A* and Dijkstra [8], the proposed trajectory-generation algorithm mathematically formulates an optimization problem by approximating the obstacles into convex-concave constraints. After obtained the trajectories, we control the velocities of the robots to avoid possible inter-robot collision while considering the time efficiency. The proposed approach is decentralized since each robot only computes its own trajectory and velocity according to the surrounding environments.

In summary, the main contributions of this work can be summarized as follows:

1) A two-stage decoupled approach is developed for multi-robot path planning in continuous environments. The proposed algorithm can be implemented in decentralized online operations.

2) As compared to the conventional heuristic algorithms that decompose the spaces into discrete grids, we derive the obstacle-avoidance trajectories by formulating an optimization problem in continuous environments, where the obstacles are circular-approximated.

3) The inter-robot collisions of the planned trajectories are avoided using velocity control by formulating a quadratic programming (QP) problem. In the meanwhile, the travel time of robots is minimized by adopting the control Lyapunov function (CLF). Finally, experiments and simulations are implemented to validate the effectiveness of the proposed approach.

Fig. 1. The diagram of the multi-objective robot swarm system.

The remainder of this paper is organized as follows. Section II provides the brief review of CBF-CLF used in this work. Next, the multi-robot system model and problem statement are presented in Section
III. To solve the path-planning problem, we first generate the obstacle-avoidance trajectories using DCCP in Section IV. Next, the possible collisions of robots are avoided by solving the QP problem derived in Section V. Furthermore, performance of the proposed approach is evaluated in Section VI, and Section VII concludes this work.

Notation: Uppercase boldface and lowercase boldface letters are used to denote matrices and vectors, respectively. $\|A\|$ stands for the $\ell_2$ norm of $A$ while $\hat{A}$ denotes the derivative of $A$. $L_fA$ is the Lie derivative of function $A$ to $f$, $\hat{A}$ the estimator of $\hat{A}$, where $\hat{A}$ can be a matrix, vector, or scalar. $A^T$ stands for the transpose of matrix $A$. Finally, $\nabla f(A)$ represents the gradient of function $f(A)$.

2. A brief review of CBF-CLF problem formulation
Firstly, we provide an elegant introduction of control barrier functions (CBFs). The complete details can be found in [9]. A continuous function $\Gamma(x)$ is defined as a class $K$ function if $\Gamma(x)$ is strictly increasing and $\Gamma(0) = 0$ for $x \leq 0$.

In this paper, we consider a continuous-time affine control system formulated by

$$\dot{x} = f(x) + g(x)u$$  \hspace{1cm} (1)$$

Where $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are both locally Lipschitz. In addition, $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the state and control input, respectively. Barrier functions are used to define the set of all states by

$$C := \{ x \in \mathbb{R}^n : h(x) \geq 0 \}$$  \hspace{1cm} (2)$$

where $h: \mathbb{R}^n \rightarrow \mathbb{R}^n$ is continuously differentiable. If there exist class $\kappa$ functions $\beta_1, \beta_2$ and $\gamma$. A CBF function $B(x)$ for a control system is defined by the following inequalities:

$$\frac{1}{\beta_1(h(x))} \leq B(X) \leq \frac{1}{\beta_2(h(x))}$$ \hspace{1cm} (3)$$

$$\inf_{u \in U} \left[ L_f B(x) + L_g B(x)u - \frac{\gamma}{B(x)} \right] \leq 0$$ \hspace{1cm} (4)$$

for all $x \in C$. $L_f$ and $L_g$ are the Lie derivatives of the function $B(x)$ to $f$ and $g$, respectively. The Lie derivatives are defined

$$L_f B(x) = \nabla f(x)$$ \hspace{1cm} (5)$$

where $\frac{\partial B}{\partial x} \in \mathbb{R}^{n \times 1}$ and $f(x) \in \mathbb{R}^{n \times 1}$. Thus, $L_f B(x)$ is a scalar. For a CBF $B(x)$ and $\gamma > 0$, any Lipschitz continuous controller $u$ with

$$K_{cbf}(x) = \{ u \in U : L_f B(x) + L_g B(x)u - \frac{\gamma}{B(x)} \leq 0 \}$$ \hspace{1cm} (6)$$

renders set $C$ forward invariant [9] for the affine control system in Equation (1).

A continuously differentiable function $V: \mathbb{R}^n \rightarrow \mathbb{R}^n$ is an exponentially stabilizing control Lyapunov function (CLF) if there exist positive constants $c_1, c_2, c_3 > 0$ such that for any $x \in \mathbb{R}^n$, the following inequalities hold:

$$c_1 \|x\|^2 \leq V(X) \leq c_2 \|x\|^2$$ \hspace{1cm} (7)$$

$$\inf_{u \in U} \left[ L_f V(x) + L_g V(x)u + c_3 V(x) \right] \leq 0$$ \hspace{1cm} (8)$$

Given a CLF $V(X)$, any Lipschitz continuous controller $u$ holds

$$K_{clf}(x) = \{ u \in U : L_f V(x) + L_g V(x)u + c_3 V(x) \leq 0 \}$$ \hspace{1cm} (9)$$

The CLF and CBF can be combined into a single controller through a quadratic programming (QP) problem represented as
\[
\begin{align*}
\mathbf{u}(x)^* &= \arg \min_{\mathbf{u} \in \mathcal{U}} \frac{1}{2} \mathbf{u}^T \mathbf{H}_s(x) \mathbf{u} + F(x)^T \mathbf{u} \\
\text{subject to:} & \quad \mathbf{L}_s \mathbf{v}(x) + \mathbf{L}_g \mathbf{v}(x) + \mathbf{c}_s \mathbf{v}(x) - \delta_s \leq 0 \\
& \quad \mathbf{L}_f \mathbf{B}(x) + \mathbf{L}_g \mathbf{B}(x) \mathbf{u} - \mathbf{a}(h(x)) \leq 0
\end{align*}
\] (10)

Where \( C_3 \geq 0 \) is a constant and \( \alpha(h(x)) \) is a class \( K \) function. \( H_s(x) \) is a positive definite and \( F(x) \in \mathbb{R}^{m \times n} \). In addition, \( \delta_0 \) is the relaxation factor on the CLF.

3. System overview and problem statement

3.1 Multi-robot system model

Consider a swarm of \( Z \) robots moving in a continuous environment. All the robots are equipped with LiDARs and RGB-D cameras to detect the surrounding situations. The velocity and position of adjacent robots as well as the locations of \( M \) obstacles can be measured using the equipped sensors. The obstacles in arbitrary shapes are centered \( \{ \mathbf{p}_m \}_{m=1,...,M} \in \mathbb{R}^2 \).

We assume the start and destination locations are given for each robot. The task for each robot is efficiently traveling from the start point to its destination in a continuous environment while avoiding collisions with obstacles and other robots. Furthermore, we assume the robots consist of motion primitives that allow them to deterministically move from \( \mathbf{y}_{z,n} \) to \( \mathbf{y}_{z,n+1} \) with velocity \( \mathbf{v}_{z,n} \), where \( \{ \mathbf{y}_{z,n} \}_{n=1,...,N} \in \mathbb{R}^2 \) are the set of sampled points in the trajectory of the \( z \)-th robot. The maximal velocity and acceleration of a robot are given as \( v_{\text{max}} \) and \( a_{\text{max}} \), respectively.

3.2 Path-Planning Problem Statement

For the multi-objective path planning, the designated robots are assigned to achieve their own destinations. The objectives and constraints are summarized as follows:

- **Path Length**: The sum of path lengths is one of the most important metrics for the performance of the proposed path-planning algorithm. Let \( L_z \) be the length of the \( z \)-th robot that travels from the start point to its destination. The objective function can be modeled as the minimization of \( \sum_{z=1}^{Z} L_z \).

- **Time Efficiency**: The minimal travel time can be achieved if all robots move at \( v_{\text{max}} \) in the planned trajectory. Therefore, the travel time of a robot is lower bounded by \( \frac{L_z}{v_{\text{max}}} \). Obviously, the possible collisions among the robots will increase the max travel time.

- **Safety**: The safety constraints can be divided into two categories, namely static and dynamic collision-avoidance constraints. Firstly, the static collision avoidance refers to the avoidance of static obstacles in the given environment. Next, the inter-robot collisions are considered as dynamic collision avoidance. The safety distance must be satisfied with both of the constraints during the movement of the robots.

- **Smoothness**: A smooth and continuous UGV’s trajectory without sharp turns is desired in practice. We will first characterize the smoothness of the generated trajectories. For any three ordered positions \( \{ \mathbf{y}_{z,n-1}, \mathbf{y}_{z,n-1}, \mathbf{y}_{z,n+1} \} \) with \( n = 2, \cdots, N-1 \), the turning angle is defined as

\[
\cos \beta_{z,n} = \frac{(y_{z,n+1} - y_{z,n}) \cdot (y_{z,n} - y_{z,n-1})}{\| y_{z,n+1} - y_{z,n} \| \cdot \| y_{z,n} - y_{z,n-1} \|}
\] (11)

We propose to minimize the following metric to ensure the smoothness:

\[
S = \sum_{z=1}^{Z} \sum_{n=2}^{N-1} |\beta_{z,n}|
\] (12)

In this work, we aim to achieve the aforementioned objectives subject to the constraints for each robot.
4. Path planning for a single robot based on dccp

4.1 Obstacle-Avoidance Path Planning for A Single Robot

In the sequel, we investigate the path-planning problem for a single robot. The inter-robot collisions are neglected temporarily. The goal of path planning is to find the shortest trajectory connecting initial point $s_{init}$ and destination $s_{des}$ that avoids Mobstacles, which are approximated as $\tilde{M}$ circles using Algorithm 1. Apart from obstacles, the available road denoted by $\Gamma$ is assumed to be known in advance. For instance, the available roads and corridors are usually constrained by the kerbstones or walls.

We assume a continuous trajectory is discretized into $N$ points. The problem is represented as

$$\begin{align}
\text{minimize} & \quad L_z \\
\text{subject to} & \quad C1: y_1 = s_{init}, y_N = s_{des} \\
& \quad C2: \|y_{z,n} - y_{z,n,i}\| \leq \frac{L_z}{N} \\
& \quad C3: \|y_{z,n} - p_m\| \geq r_m, m = 1, ..., \tilde{M} \\
& \quad C4: y_{z,n} \in T
\end{align}$$

where $L_z$ is the path length for $z = 1, ..., Z$. $C1$ provides the start and destination locations while $C2$ ensures the smoothness of the generated trajectory. A larger $N$ will produce a smoother trajectory but the computational complexity is also improved as the increment of variables. In addition, $C3$ guarantees the safety distance between the robot and circular-approximated obstacles.

The constraint $C3$ is a convex-concave constraint, which cannot be solved by generic solvers. To cope with this issue, we linearize the problem in Equation (13) around a given point $x_n^{(i)}$:

$$\begin{align}
\text{minimize} & \quad L_z + \frac{\beta}{2}\|y_{z,n} - y_{z,n}^{(i)}\|^2 \\
\text{subject to} & \quad C1, C2, C4 \text{ in Equation (13)} \\
& \quad F(y_{z,n}) - G(y_{z,n}^{(i)}) - \langle \nabla G(y_{z,n})y_{z,n} - y_{z,n}^{(i)} \rangle \leq 0, N = 1, ..., N
\end{align}$$

where we set $F(y_{z,n}) = r_m$ and $G(y_{z,n}) = \|y_{z,n} - p_m\|$. The subscript $(i)$ denotes the $i$-th iteration for the optimization of Equation (13). Furthermore, $\beta$ is added to the objective function as a regularization term.

Using the sequential convex programming algorithm in [11], the problem in Equation (14) can be iteratively solved via efficient solver, such as DCCP [12]. This problem can be regarded as a difference of two convex (DC) problem, thereby the convergence of Equation (14) is guaranteed. It is worth noting that the obtained trajectories cannot be verified as global optimums due to the non-convex nature.

Once the trajectories for all of the robots are generated, the sum of path lengths is lower bounded by

$$L_{min} = \sum_{z=1}^{Z} L_z^*$$

where $L_{z}^*$ is the obtained path length from Equation (13). We will validate the effectiveness of the proposed algorithm by comparing the path lengths with other algorithms in Section VI.

5. Collision-aware path planning for multi-robot scenarios

With the given trajectory for all of the robots, the collision-avoidance problem is addressed by velocity control in the sequel. Based on CBF-CLF reviewed in Section II, the time-varying velocities of robots are obtained by solving a quadratic programming (QP) problem. For each robot, it only needs to acquire the location and velocity of the nearest robot. Thus, the operation is decentralized and can be implemented online.
5.1 Leader-follower dynamic model

Once the trajectories of robots are generated in Section IV, we now consider the collision-aware path planning by velocity control. In the decentralized dynamic model, we assume one robot is able to obtain the location and velocity of its nearest robot. Since no central coordinator exists, each robot can only control its own velocity. Thus, we propose a leader-follower (LF) dynamic model, where the velocity of the leader robot cannot be changed while the follower robot executes the decentralized algorithm.

Consider a 2-D LF dynamic model that the velocities and locations of the both robots are known at the follower robot. The trajectories are represented by \( L_l \) and \( L_f \). At the \( t \)-th instant, the velocities and locations are given as \( [v_l(t), y_l(t)] \in L_l \) and \( [v_f(t), y_f(t)] \in L_f \) for the follower and leader robot, respectively. As shown in Fig. 2, the velocity components between two robots are denoted by:

\[
\tilde{v}_l = |v_l(t)| \cos(\alpha_l) \tag{16}
\]

\[
\tilde{v}_f = |v_f(t)| \cos(\alpha_f) \tag{17}
\]

Where the \( \alpha_l \) and \( \alpha_f \) are given as follows

\[
\cos(\alpha_l) = \frac{(\hat{y}_l(t+1) - \hat{y}_f(t))^T(\hat{y}_l(t) - \hat{y}_f(t))}{|\hat{y}_l(t+1) - \hat{y}_f(t)| \cdot |\hat{v}_l(t) - \hat{v}_f(t)|} \tag{18}
\]

\[
\cos(\alpha_f) = \frac{(\hat{y}_f(t+1) - \hat{y}_f(t))^T(\hat{y}_l(t) - \hat{y}_f(t))}{|\hat{y}_f(t+1) - \hat{y}_f(t)| \cdot |\hat{v}_l(t) - \hat{v}_f(t)|} \tag{19}
\]

Furthermore, the distance of two robots is denoted by

\[
D(t) = |\hat{y}_l(t) - \hat{y}_f(t)| \tag{20}
\]

Given \( x(t) = (\tilde{v}_r(t), \tilde{v}_f(t), D(t)) \), the dynamics of an LF system can be compactly represented as

\[
x(t) = \begin{bmatrix} -F_r(v_u(t)) / M \\ \alpha_u(t) \\ \hat{v}_l(t) - \hat{v}_f(t) \end{bmatrix} + \begin{bmatrix} 1 / M \\ 0 \\ 0 \end{bmatrix} u(t) \tag{21}
\]

where \( F_r(v_u(t)) = f_0 + f_1 \tilde{v}_r(t) + f_2 \tilde{v}_r(t)^2 \) is the aerodynamic drag; the coefficients \( f_0, f_1, \) and \( f_2 \) are...
depend on the practical environments. \( \alpha g(t) \) is the random noise of the measured velocity of the leader robot respect to a Gaussian distribution with zero mean. Next, we consider solving the decentralized collision-aware path planning by formulating a QP problem at the \( t \)-th instant. For notation simplicity, the state variable \( x(t) \) is written as \( x \) temporarily.

### 5.2 Travel Time Minimization

The minimum travel time can be achieved when the robots move with the maximal velocity. We sort the path lengths from the largest to the smallest, i.e., \( L_z \geq L_{z+1} \). Thus, the lower bound of the travel time to accomplish the tasks is

\[
T_{\text{min}} = \frac{L_z}{v_{\text{max}}}
\]

To minimize the travel time, we should select the robot with a larger path length as a leader robot since the follower robot may reduce its velocity to avoid the inter-robot collisions. In this way, the robot with a larger path length can keep a relatively larger velocity.

Given the dynamic model, we give a “soft constraint” for each robot to achieve a desired velocity:

\[
v_g = \left\lVert v_{\text{max}} \right\rVert \cos(\alpha f)
\]

This constraint only works when the safety distance is ensured. The soft constraint can be written in terms of a candidate CLF:

\[
V(x) = (\tilde{v}(f) - v(d))^2
\]

For any \( c > 0 \), the following inequality verifies that \( V(x) \) is a valid CLF [13]

\[
\inf_{x \in \mathbb{R}} [L_x V(x) + L_y V(x)u + c_v V(x)] \leq 0
\]

### 5.3 CBF-CLF QP formulation

Now we consider the constraints for inter-robot collision avoidances. We express the safety distance as a “hard constraint” given by

\[
D \geq \tilde{v}_f
\]

where \( \tau \) is a constant reaction time for the follower robot. The hard constraint ensures the safety distance of two robots. We consider the function \( h(x) = D - \tilde{v}(f) \) and a candidate CBF

\[
B(x) = -\log\left(\frac{h(x)}{1 + h(x)}\right)
\]

Recalling Equation (6), we set \( \gamma = 1 \)

\[
\inf_{x \in \mathbb{R}} \left[ L_f B(x) + L_g B(x)u - \frac{1}{B(x)} \right] \leq 0
\]

which validates \( B(x) \) is a CBF.

As a result, a CLF-CBF QP is formulated by combining the above constraints [14]:

\[
\begin{align*}
    u^* &= \arg \min \frac{1}{2} u^T H_{\text{acc}}(x)u + F_{\text{acc}}(x)^T u \\
    \text{subject to:} & \quad C1: [L_x V(x), -1]u \leq b_{\text{clf}} \\
    & \quad C2: [L_y B(x), 0]u \leq b_{\text{clf}} \\
    & \quad C3: [1, 0]u \leq b_{\text{clf}}
\end{align*}
\]

where
\[ L_x B(x) = \frac{\tau}{M(1 + D - \tau \dot{v}_x)(D - \tau \dot{v}_x)} \]
\[ L_j B(x) = \frac{-\tau F_x / M + \dot{v}_j - \dot{v}_x}{(1 + D - \tau \dot{v}_j)(D - \tau \dot{v}_j)} \]
\[ L_x V(x) = \frac{2(\dot{v}_j - \dot{v}_x)}{M} \]
\[ L_j V(x) = \frac{2(\dot{v}_j - \dot{v}_x)}{M} \]  

and
\[ b_{xj} = -L_j V(x) - cV(x) \]
\[ b_{xj} = -L_j B(x) - \frac{1}{B(x)} \]
\[ b_{xj} = a_{fc} \]  

The constraint \( C_1 \) is generated by CLF and the additive parameter \( \sigma \) makes the CLF constraint “soft”. The constraint \( C_2 \) guarantees the “hard” safety distance for two robots while \( a_{fc} \) is the acceleration limit of a robot in constraint \( C_3 \). It is worth noting that the combination of CLF and CBF assures the requirements are not in conflict.

In addition, the cost function related to the control yields the following function in terms of \( u \):
\[ C(u) = \left\| F_x - u \right\|^2 / M^2 = \frac{1}{M^2}(F_x^2 + u^2 - 2uF_x) \]  

which can be converted into the matrix form in Equation (28)
\[ H_{acc} = 2 \begin{bmatrix} \frac{1}{M^2} & 0 \\ 0 & p_{ac} \end{bmatrix}, \quad F_{acc} = -2 \frac{F_x}{m} \]  

where \( p_{ac} \) is the relaxation weight of the parameter \( \sigma \).

The QP problem in Equation (28) can be solved by an available solver such as the function `quadprog` in MATLAB.

---

**Algorithm 1: Decentralized Path Planning Algorithm**

**Input:**
- Initial and terminal positions of robots: \( \{y_{z, int}\}_{z=1,Z}, \{y_{z, des}\}_{z=1,Z} \)
- The positions of the circular approximated obstacles: \( \{p_m\}_{m=1,M} \)

**Procedures:**

1. Generate the obstacle-avoidance trajectories by Equation (13): \( \{y_{z,a}\}_{z=1,Z} \)
2. for \( n = 1, \ldots, N \) do
    3. Sort the left travel distances for all the robots;
4. for \( z = 1, \ldots, Z \) do
    5. Select the leader robot and the nearest robot as the follower;
    6. Calculate the controller \( u \) by solving Equation (28);
    7. Update the position and velocity by Equation (33) and Equation (34), respectively;
8. end for
9. end for

**Outputs:** The center coordinates and radius of \( K^* \) circles.

---

### 5.4 Time-varying control systems

The velocities are updated by
\[ \ddot{v}_j(t+1) = \ddot{v}_j(t) + \frac{u(t) - F_x}{M} \Delta T \]  

(33)
where $\Delta T$ is a time slot given by

$$\Delta T = \frac{y_j(t+1) - \tilde{y}_j(t)}{v_j(t)}$$

(34)

Furthermore, we assume the leader and follower robot are the $zl$-th and $zf$-th robot, respectively. The sampled points are given by $\tilde{y}_j(n) = y_{zl,n}$ while the initial position of the follower robot is given by $\tilde{y}_j(1) = y_{zf,\text{ini}}$. Given the time slot according to the leader robot, the sampled trajectories of the follower robot are updated by

$$\tilde{y}_j(t+1) = \tilde{y}_j(t) + v_j(t)\Delta T$$

(35)

As the velocity components of the LF model are derived, we can straightforwardly calculate the velocities of the both robots by:

$$v_j(t+1) = \frac{v_j(t+1)}{\cos(\alpha f)} \frac{y_j(t+1) - \tilde{y}_j(t)}{y_j(t+1) - \tilde{y}_j(t)}$$

(36)

In each instant, the left travel distances of robots are sorted. The robot with a larger left distance is selected as the leader robot. In this way, the travel time can be minimized according to Section V-B. The decentralized path planning algorithm is elaborated in Algorithm 1, where the DCCP and CBF-CLF are implemented. Specifically, the obstacle-avoidance trajectories for each robots are first generated by temporarily ignoring the possible collisions using DCCP. Next, the inter-robot collisions are considered by formulating a QP problem using CBF-CLF. In the time-varying dynamic system, the possible collision is avoided by controlling the velocities of robots.

6. Performance evaluation

To evaluate the performance of the proposed multi-objective path-planning algorithm, the concept-proof experiments are presented in this section. We use multiple RoboMaster produced by Da-Jiang Innovations (DJI) to accomplish the experiments. The RoboMaster can be controlled via computer with the SDK provided by DJI. Each RoboMaster is equipped with a LiDar and a RGB-D camera such that the state of other robots can be detected. We assume the geographical information of the given environment is acknowledged.

| TABLE I PARAMETERS USED IN THE EXPERIMENTS. |
|-----------------|------|------|------|------|
| $M$             | 2    | $f_0$| 0.1  | $f_1$| 5    |
| $f_2$           | 0.25 | $\rho_{sc}$| 10   | $v_{s}(0)$| 1   |
| $v_{w}(0)$      | 0.1  | $v_{s}(0)$| 3    | $r$  | 4    |
| $\delta$        | 0.01 | $c$  | 10   | $\alpha_{fc}$| 20 |

6.1 Velocity Control for LF Dynamic Model

We first investigate the velocity control of two robots in the LF dynamic model. The leader robot is observed by the follower robot. After obtained the position and velocity of the leader robot, the follower robot dynamically control its velocity to avoid possible collisions. If no collision exists, the follower robot is prone to move with the maximum velocity.

In Fig. 3, the velocities of two robots are illustrated by exploiting the CBF-CLF-based algorithm derived in Section V.

Fig. 4 reveals the variation of the states and the control input in the LF dynamic system. The control input is reported in the bottom figure.
Fig. 3. The velocity of the leader and follower robots.

Fig. 4. The distance between the two robots (up), and the controller of two robots (bottom).
6.2 Path-Planning for the Robot Swarm

Next, we consider the multi-objective path planning for the robot swarm. Assume that four RoboMasters are assigned tasks to achieve different destinations in a continuous environment, where the obstacles are randomly deployed.

The left side of Fig. 5 shows the trajectories generated using the circular-approximated obstacles corresponding to the practical environment in the right side.

Fig. 5. The trajectory of two robots.

Next, we consider the multi-objective path planning for the robot swarm. Assume that four RoboMasters are assigned tasks to achieve different destinations in a continuous environment, where the obstacles are randomly deployed.

The left side of Fig. 5 shows the trajectories generated using the circular-approximated obstacles corresponding to the practical environment in the right side.

7. Conclusion and Future Work

In this work, a decentralized path-planning approach is proposed for multi-robot scenarios. Based on DCCP and CBF-CLF, the decoupled problems are separately resolved in two stages. Firstly, the trajectories are generated by neglecting the inter-robot collisions. Next, we formulate a QP problem using CBF-CLF to design the velocity controller for each robot. In this way, the travel time as well as the trajectory length are minimized. The decentralized algorithm operates on the robots by implementing the LF model. Finally, extensive experiments verify the advantages of the proposed decentralized approach, showing the improvements on efficiency and solution quality.
In our future works, we will consider the dynamic path planning problem for the robots. For instance, one of the most important path-planning problems is the existing of the dynamic obstacles. It requires much more computational resource for the computer equipped on the robot. Moreover, the path-planning problem can be also extended to the 3-D scenarios, such as the unmanned aerial vehicles (UAVs).

References
[1] M. Elhoseny, A. Tharwat, and A. E. Hassanien, “Bezier curve based path planning in a dynamic field using modified genetic algorithm,” Journal of Computational Science, vol. 25, pp. 339–350, 2018.
[2] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, “A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization,” Applied Soft Computing, vol. 59, pp. 68–76, 2017.
[3] P. Surynek, “An optimization variant of multi-robot path planning is intractable,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 24, no. 1, 2010.
[4] M. Nazarahari, E. Khanmirza, and S. Doostie, “Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm,” Expert Systems with Applications, vol. 115, pp. 106–120, 2019.
[5] G. Hitz, E. Galceran, M.-E. Garneau, F. Pomerleau, and R. Siegwart, “Adaptive continuous-space informative path planning for online environmental monitoring,” Journal of Field Robotics, vol. 34, no. 8, pp. 1427–1449, 2017.
[6] W. Xiao, C. Belta, and C. G. Cassandras, “Decentralized merging control in traffic networks: A control barrier function approach,” in Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems, 2019, pp. 270–279.
[7] S. Zeng, Y. Zhu, and C. Qi, “HTN-based multi-robot path planning,” in 2016 Chinese Control and Decision Conference (CCDC), Yinchuan, China, May. 2016, pp. 4719–4723.
[8] A. Candra, M. A. Budiman, and K. Hartanto, “Dijkstra’s and a-star in finding the shortest path: a tutorial,” in 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA). IEEE, 2020, pp. 28–32.
[9] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” IEEE Transactions on Automatic Control, vol. 62, no. 8, pp. 3861–3876, 2016.
[10] N.-s. P. Hyun, P. A. Vela, and E. I. Verriest, “A new framework for optimal path planning of rectangular robots using a weighted Lp norm,” IEEE Robotics and Automation Letters, vol. 2, no. 3, pp. 1460–1465, 2017.
[11] F. Debrouwere, W. Van Loock, G. Pipeleers, Q. T. Dinh, M. Diehl, J. De Schutter, and J. Swevers, “Time-optimal path following for robots with convex–concave constraints using sequential convex programming,” IEEE Transactions on Robotics, vol. 29, no. 6, pp. 1485–1495, 2013.
[12] X. Shen, S. Diamond, Y. Gu, and S. Boyd, “Disciplined convex-concave programming,” in 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, USA, Dec. 2016, pp. 1009–1014.
[13] A. D. Ames, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs with application to adaptive cruise control,” in 53rd IEEE Conference on Decision and Control, Los Angeles, USA, Dec. 2014, pp. 6271–6278.
[14] K. Galloway, K. Sreenath, A. D. Ames, and J. W. Grizzle, “Torque saturation in bipedal robotic walking through control lyapunov function-based quadratic programs,” IEEE Access, vol. 3, pp. 323–332, 2015.