3D Decentralized Prioritized Motion Planning and Coordination for High-Density Operations of Micro Aerial Vehicles

Xiaobai Ma, Ziyuan Jiao, Zhenkai Wang and Dimitra Panagou*

***Abstract*—This paper presents a decentralized motion planning method for multiple aerial vehicles moving among 3D polygonal obstacles resembling an urban-like environment. The algorithm combines a prioritized $A^\star$ algorithm for high-level planning, along with a coordination method based on barrier functions for low-level trajectory generation and vehicle control. To this end we extend the barrier functions method developed in our earlier work so that it treats 2D and 3D polygonal obstacles, and generates collision-free trajectories for the multi-agent system. We furthermore augment the low-level trajectory generation and control with a prioritized $A^\star$ path planning algorithm, in order to compute waypoints and paths that force agents of lower priority to avoid the paths of agents of higher priority, reducing thus congestion. This feature enhances further the performance of the barrier-based coordination, and results in shorter paths and time to the goal destinations. We finally extend the proposed control design to agents of constrained double-integrator dynamics, compared to the single-integrator case in our earlier work. We assume that the obstacles are known to the agents, and that each agent knows the state of other agents lying in its sensing area. Simulation results in 2D and 3D polygonal environments, as well as experimental results with micro aerial vehicles (quadrotors) in an indoor lab environment demonstrate the efficacy of the proposed approach.**

I. Introduction

Motion planning is a core, active research topic in the robotics and control systems communities, that becomes particularly challenging for multi-robot systems [1], [2]. Motion planning for a single robot is typically treated with (i) sampling-based methods, including probabilistic roadmaps [3], and rapidly-exploring random trees [4], [5], (ii) Lyapunov-based methods, including either the definition of closed-form feedback motion plans via potential functions or vector fields, or computation of Lyapunov-based feedback motion plans via sum-of-squares programming [6], [7], (iii) graph search and decision theory, see also [8], [9] for an extensive presentation on motion planning methods.

*Corresponding author

Xiaobai Ma is a Graduate Student with the Department of Aeronautics and Astronautics, Stanford University, USA; maxiaoba@stanford.edu.

Ziyuan Jiao is a Graduate Student with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, USA; zyjiao@umich.edu.

Zhenkai Wang is a Graduate Student with the Department of Aeronautics and Astronautics, Stanford University, USA; zackwang@stanford.edu.

Dimitra Panagou is an Assistant Professor with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, USA; dpanagou@umich.edu.

Dimitra Panagou would like to gratefully acknowledge the support of the NASA Grant NNX16AH81A. This work was in part done during the University of Michigan SURE (Summer Undergraduate REsearch) Project of the first and second author.

Each method has its own merits and disadvantages. Sampling-based methods typically produce paths that need to be smoothened prior to execution, while the trajectory generation for agents with complex dynamics is not straightforward; furthermore, the convergence to the desired goal is probabilistic. Potential functions [10] unify path planning and trajectory generation, but they typically exhibit local minima which result in trajectories getting stuck away from the desired goal, while special forms of local-minima free functions typically require tweaking of certain parameters to produce collision-free and convergent trajectories [11], [12]. Reactive methods have been refined over the years to overcome the deficiencies of gradient-based solutions [13]. In [14], the authors introduce the notion of a "reciprocal velocity obstacle" which guarantees oscillation-free motion; however, typically such methods do not consider global path optimization. In addition, the traditional discrete planning methods such as rapidly-exploring random trees and probabilistic roadmaps do not scale well for multi-agent problems as they require extensive computation when directly used in high-dimensional spaces. The Lazy Probabilistic Roadmap Method [15], and MAPP [16] aim to reduce the computational demand of such methods; however, collision avoidance in these approaches is inherently treated in a discrete manner, i.e., agents can only switch between nodes, but do not fully use the space between nodes. In contrast to those solutions, potential function and vector field based methods can provide a continuous avoiding behavior. More recently, safety verification of hybrid systems by means of barrier certificates [17], [18] has motivated and been used in the computation of Lyapunov-based feedback controllers for classes of nonlinear systems by sum-of-squares programming [19], [20]. Here we adopt an approach that utilizes barrier functions and seeks to render certain level sets as positively invariant sets, but instead of computing the Lyapunov barrier (or barrier certificate), we utilize a specific given form for the candidate Lyapunov barrier function.

In addition to the previous methods that have been mostly used in single-agent settings, prioritized planning has been a popular method in the multi-robot motion planning literature, and various approaches on assigning priorities are described, for instance, in [21]–[28]. In a prioritized approach, each robot has its own priority, and computes its path based on the relative order imposed by their priorities. Hence the resulting path for each robot avoids static obstacles on the map as well as the robots with higher priority in its communication region, which are considered as dynamic obstacles.

A. Motivation and Problem Challenges

This work presents a decentralized prioritized algorithm for the motion planning and coordination of multiple agents in 3D obstacle environments, which combines a prioritized $A^\star$ algorithm for the high-level path planning, and a low-level barrier-based control method for multi-agent coordination and collision avoidance.

We are primarily motivated by applications relevant to the future Unmanned Aircraft Systems Traffic Management in low-altitude airspace. Arguably, the problems of sense-and-avoid, separation assurance, and trajectory prediction and generation have been well-studied research topics both in the robotics field, as well as in the aerospace field, see for instance [29]–[41]. Nevertheless, the envisioned autonomous low-altitude flight of small Unmanned Aerial Systems (UAS) in urban environments poses both technological and fundamental challenges, that make the problem different compared to manned flight in the current airspace structure, or compared to 2D robot deployments.

Towards this end, in this paper we build protocols and algorithms for the automated 3D trajectory generation and conflict resolution for small UAS (i) in cluttered polygonal environments that resemble an urban environment under given constraints on the sensing, communication and mobility of the agents, (ii) in making the trajectory generation for each agent anticipatory of the others' actions, so that areas of high congestion are avoided.

The proposed approach combines a priority-based high-level path planning and a barrier-based low-level coordination and control scheme. The low-level reactive coordination scheme offers a computationally inexpensive way to resolve conflicts, however with the caveat that it typically results in unnecessarily longer (i.e., suboptimal) paths, that often suffer from chattering. Thus a prediction or look-ahead capability in the decision-making processes of the agents is beneficial for efficient mission accomplishment. For this reason we augment the low-level coordination with the proposed priority-based high-level planning in order to improve the performance of the system when it comes to high-density operations of autonomous agents, such as small micro aerial robots in cluttered and congested 2D and 3D environments.

B. Contributions

More specifically, we propose a two-level strategy for the decentralized motion planning, trajectory generation and control of micro aerial vehicles in 3D polygonal obstacle environments (Fig. 1), which combines a prioritized $A^\star$ high-level path planner along with a low-level controller that is based on barrier functions for multi-agent coordination. We first build upon the method in [42] and modify it to account for known polygonal obstacles in 2D and 3D environments, as well as for agents of double integrator dynamics. We also define a prioritized $A^\star$ algorithm to plan paths for the agents that result in reduced traffic congestion in confined environments. The prioritized $A^\star$ is triggered whenever agents become connected, and results in paths for the lower priority agents that are anticipatory of the paths of the agents of higher priority. *This way the agents of lower priority avoid the areas-to-be-occupied in a short time horizon ahead by the agents of higher priority, with the effect of reducing traffic congestion.* System safety is taken care of by the low-level control using a semi-cooperative coordination protocol built upon our work in [43]. In effect, the proposed two-level strategy combines the advantages of the utilized methods while overcoming their disadvantages. Namely, the decentralized barrier functions provide safe directions of motion for the agents, and furthermore guide the design of their velocity adjustment algorithms for conflict resolution and collision avoidance. The derived semi-cooperative protocol guarantees the generation of collision-free trajectories without necessarily forcing all agents to participate in conflict resolution (hence the term semi-cooperative). This way, the trajectory generation does not suffer from the computational load of pure discrete path-planning methods in high-dimensional spaces. In addition, the prioritized $A^\star$ algorithm provides waypoints that eliminate the "getting stuck" situation often encountered in pure potential function-based methods, and forces agents of lower priority to avoid the paths of agents of higher priority, resulting thus in shorter paths that favor the reduction of traffic jam. Finally, the low-level control design in this paper is based on double-integrator dynamics with acceleration bounds, a consideration which is different compared to our earlier relevant work [42], [43] that addresses kinematic agent models only. Experimental results with three quadrotors accompany the computer simulations and demonstrate the efficiency of the proposed approach in realistic settings.

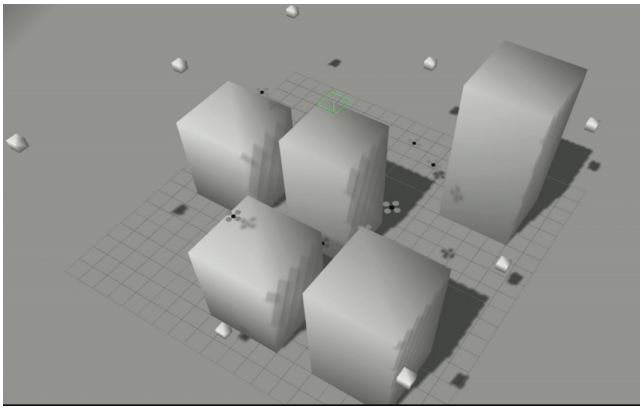


Fig. 1. The considered scenario: micro aerial vehicles flying in an urban area towards goal destinations while avoiding obstacles (other quadrotors, buildings).

Part of this work has been presented in [44]. Compared to the conference version, here we consider agents of constrained double-integrator dynamics, instead of agents governed by single-integrator dynamics, and provide all the theoretical proofs that were omitted in the conference version in the interest of space.

The paper is organized as follows: Section II gives the mathematical modeling and an overview of the proposed approach. The high-level prioritized planning and the low-level coordination are given in Sections III and IV, respectively, while Section V presents their extension to 3D environments.

Simulation results in 2D and 3D environments, including a comparison of the prioritized $A^\star$ algorithm with the standard $A^\star$ algorithm, are given in Section VI, whereas experimental results using quadrotors are presented in Section VII. Section VIII summarizes our findings and thoughts on future research. Finally, the Appendix reviews the standard $A^\star$ planning algorithm.

II. Modeling and Problem Statement

The equations describing the motion of a quadrotor with mass m and inertia matrix $\boldsymbol{J}$ expressed in the body-fixed frame are given as:

$$\dot{\boldsymbol{r}}^n = \boldsymbol{C}(\boldsymbol{\eta})_b^n \, \boldsymbol{v}^b, \tag{1a}$$

$$\dot{\boldsymbol{v}}^b = \boldsymbol{\omega}^b \times \boldsymbol{v}^b + m^{-1}\boldsymbol{F}^b, \tag{1b}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}\,\boldsymbol{\omega}^b, \tag{1c}$$

$$\dot{\boldsymbol{\omega}}^b = \boldsymbol{J}^{-1}(-\boldsymbol{\omega}^b \times \boldsymbol{J}\boldsymbol{\omega}^b + \boldsymbol{M}^b), \tag{1d}$$

where $\boldsymbol{r}^n$ is the position vector of the center of gravity of the quadrotor with respect to (w.r.t.) the navigational (i.e., an inertial) frame of reference, $\boldsymbol{v}^b$ is vector of translational (linear) velocity of the quadrotor expressed in the body frame of reference, $\boldsymbol{\eta} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$ is the attitude representation of the body frame relative to the navigational frame in 3-2-1 Euler angles (yaw (ψ), pitch (θ), roll (ϕ)), $\boldsymbol{C}(\boldsymbol{\eta})_b^n$ is the rotation matrix in 3-2-1 Euler angle representation from the body frame to the navigational frame, $\boldsymbol{\omega}^b$ is the angular velocity of the vehicle expressed in the body frame, $\boldsymbol{F}^b$ is the total force including the motor trust, gravity and aerodynamic forces expressed in the body frame, and $\boldsymbol{M}^b$ is the total torque expressed in the body frame, containing the moments generated by thrust, gravity and aerodynamic forces, and

$$\boldsymbol{R} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}.$$

In the subsequent simulation and experimental trials, the motor regulation as well as the attitude control of the quadrotor are implemented by the simulator and the vehicle's on-board controller, respectively. Hence our planning, coordination and control design focuses on computing and implementing translational (linear) velocity and acceleration commands to the quadrotor.

We furthermore assume that the roll, pitch, and yaw angles remain close to zero during flight, so that the equations of motion can be approximated via double integrator dynamics:

$$\dot{\boldsymbol{r}} = \boldsymbol{v}, \tag{2a}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{a}, \tag{2b}$$

where $\boldsymbol{r}$ is the position vector, $\boldsymbol{v}$ is the linear velocity vector, and $\boldsymbol{a}$ is the linear acceleration vector, all expressed w.r.t. the navigational frame.

We consider N quadrotors, called thereafter agents, under double-integrator dynamics (2), that are deployed in a known 2D workspace $\mathcal{W}$ with M static polygonal obstacles. Each agent $i \in \{1, \ldots, N\}$ is modeled as a circular disk of radius r_a and is assigned a unique, static priority, that is used in the high-level planning mode, as will be explained in the sequel.

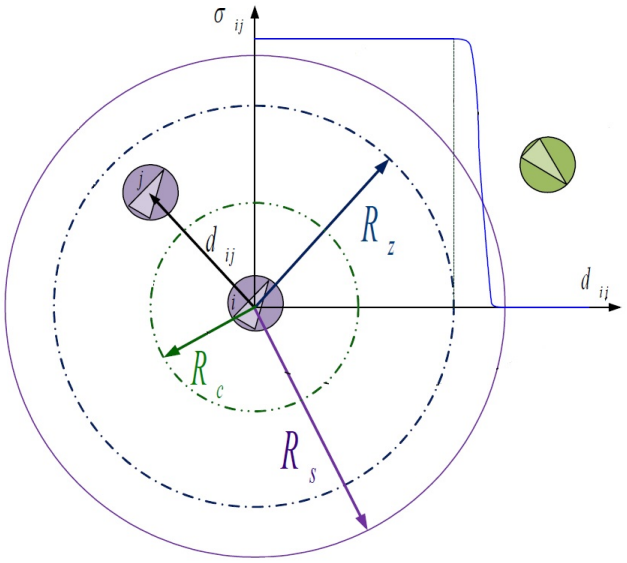


Fig. 2. Each agent i can measure the position $\boldsymbol{r}_j$ of any agent j lying within distance $d_{ij} \leq R_s$, i.e., within its sensing region. Furthermore, each follower agent i receives the linear velocity u_j of any agent j lying within distance $d_{ij} \leq R_c$, i.e., within its communication region. The radius R_z is used in the definition of the blending function $\sigma_{ij}(\cdot)$ in (9).

We assume that each agent: (i) knows its own position $\boldsymbol{r}_i$ w.r.t. the inertial frame, (ii) knows its velocity vector $\boldsymbol{v}_i$ w.r.t. the inertial frame, (iii) measures the position $\boldsymbol{r}_j$ of agents $j \neq i$ lying in its sensing region, realized as a circular disk of radius R_s, and (iv) exchanges information on linear velocities with agents $k \neq i$ lying in its communication region, realized as a circular disk of radius $R_c < R_s$, see also Figure 2. A pair of agents i and j is said to be connected if the inter-agent distance $d_{ij} = \|\boldsymbol{r}_i - \boldsymbol{r}_j\|$ is less or equal than the sensing radius R_s, i.e., $d_{ij} \leq R_s$.

Each agent is assigned a goal location $\boldsymbol{r}_{gi} = \begin{bmatrix}x_{gi} & y_{gi}\end{bmatrix}^T$ in the free space and needs to move there while avoiding static obstacles and other agents.

A. Overview of the Two-Level Approach

The proposed approach comprises a low-level coordination and control method, as well as a high-level prioritized path planning method.

The purpose of the prioritized high-level planning is to make agents of lower static priority avoid the nodes to be occupied by the nearby agents of higher static priority, providing thus a way of anticipating and avoiding traffic congestion. Towards this end, each agent in a connected group computes a path to its goal by executing an $A^\star$ planning algorithm in a sequential spirit: starting from the agent of highest priority in the group, each agent i re-plans a path to its goal location while taking into account the location of the static obstacles and the first n nodes in the waypoint lists of the agents of higher priority, and communicates the first n nodes of its new path to the agents lower priority. Hence, by avoiding the future nodes of nearby agents with higher priorities, traffic congestion is anticipated and avoided by the agents of lower priority. The high-level planning for each agent is triggered at the time instances when

a new agent of higher priority is added to its set of neighboring agents.

The generation of collision-free trajectories for each agent that track the sequence of waypoints computed out of the prioritized $A^\star$ planner is ensured by the low-level control, which is built upon the coordination protocol in [42]. In this paper we extend this coordination protocol to apply (i) in 3D polygonal environments, (ii) under double-integrator dynamics for the agents. The main idea in the low-level coordination is that connected agents are *dynamically* prioritized on-the-fly to resolve conflicts and avoid each other by considering whether their current actions result in jeopardizing safe separation, i.e., whether their control actions reduce the pairwise inter-agent distance, or the distance to static obstacles.[1] Based on this criterion, some of the agents decide to adjust their velocities so that collisions are avoided, while not all agents participate in ensuring safety (hence the term *semi-cooperative* coordination). In other words, the role of semi-cooperative coordination is to generate collision-free trajectories with not all connected agents participating in conflict resolution and deviating from their nominal paths. Imminent conflicts among agents or with static obstacles are modeled through decentralized Lyapunov-like barrier functions for each one of the agents, and the gradient vector fields of these functions dictate safe directions of motion.

The main advantage of combining the prioritized $A^\star$ high-level planning with the low-level barrier-based coordination is that it can effectively handle high-density operations in confined environments, compared to using a stand-alone barrier-based planning and control scheme, or a stand-along $A^\star$ planning method; in addition, it is guaranteed to generate collision-free trajectories. Namely, agents of lower priority essentially anticipate the paths to be followed by the agents of higher priority; hence, the prioritized $A^\star$ planning forces the agents sequentially leave the congested area based on their static priorities, resulting in shorter paths to the goal locations, and shorter travel time for the agents.

III. High-Level Prioritized Path Planning

We build upon the standard $A^\star$ path-planning algorithm, and develop a prioritized $A^\star$ path planner. The high-level planning for each agent i is triggered at initial time $t_0 = 0$ sec, and at every time instance $t_k > 0$, $k = 1, 2, \ldots$ that agent i gets connected with at least one agent j of higher static priority. The proposed high-level planning is executed as follows:

1) Assume that at time $t_k \geq 0$ sec, a group of $L \leq N$ agents becomes connected. Agents exchange information on their sets of neighbor agents and static priorities. This way all L agents eventually get informed about all $L-1$ nearby agents, even if they are not physically connected, as well as of the ordered set of priorities in the group at time t_k. Let us denote the L agents of ordered priorities by the sequence $\{a_{P_{min}}, \ldots, a_P, \ldots, a_{P_{max}}\}$, where $P \in \{1, \ldots, N\}$ and P_{min}, P_{max} the minimum and maximum priorities in the connected group at time t_k.
2) Agent $a_{P_{max}}$ plans a path to its goal using $A^\star$ taking into account the static obstacles only, and transmits the first n nodes of its path to the agent of one priority lower. The second agent computes its own path based on the static obstacles, the path of the agent of one priority higher, and transmits the information received by the higher priority agent and the first n nodes of its path to the agent of one priority lower, and so on. In summary, the agents in a connected group plan their paths in the order of their priorities; each agent treats the first n of the expected cells to be occupied by the agents of higher priority as obstacles.
3) As soon as the waypoint list of each agent a_P is generated by its prioritized $A^\star$ algorithm, the agent follows the waypoints under the low-level controller to be defined in the next section towards its destination, unless the high-level planning is triggered again. The triggering occurs if a new agent becomes connected to the originally connected group; in this case, all agents whose priorities are lower than the priority of the recently joined agent(s) replan their paths as described in step (2).

A pseudocode of the prioritized $A^\star$ algorithm is given below in Algorithm 1.

IV. Low-Level Coordination

The low-level control is based on a class of Lyapunov-like barrier functions that penalize the violation of the minimum pairwise safe separation among agents, and encode convergence to desired destinations. We briefly discuss the construction of the Lyapunov-like barrier V_i per agent i, that encodes convergence to its destination and avoidance of neighbor agents $j \neq i$ lying in its sensing region. More details are available in [42].

Let us denote $\{\boldsymbol{r}_{id}^1, \boldsymbol{r}_{id}^2, \ldots, \boldsymbol{r}_{id}^k\}$ the sequence of waypoints for each agent i computed by the prioritized $A^\star$ planner, with $\boldsymbol{r}_{id}^k = \boldsymbol{r}_{gi}$. The destination of agent i being in cell l is defined as the waypoint $\boldsymbol{r}_{id}^{l+1}$, where $l = 0, 1, 2, \ldots, k-1$. Convergence of agent i to the waypoint $\boldsymbol{r}_{id}^k = \begin{bmatrix} x_{id}^k & y_{id}^k \end{bmatrix}^T$, $k = 1, 2, \ldots$, is encoded through the cost function:

$$V_{i0} = \left\| \boldsymbol{r}_i - \boldsymbol{r}_{id}^k \right\|^2 = \left(x_i - x_{id}^k \right)^2 + \left(y_i - y_{id}^k \right)^2. \quad (3)$$

To encode that agent i keeps a safe separation d_s w.r.t. agent j, we define the constraint function:

$$c_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = (x_i - x_j)^2 + (y_i - y_j)^2 - {d_s}^2 > 0, \quad (4)$$

the barrier function:

$$b_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = -\ln(c_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j)), \quad (5)$$

and the recentered barrier function [45]:

$$r_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = b_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) - b_{ij}(\boldsymbol{r}_{id}^k, \boldsymbol{r}_j) - (\nabla b_{ij}|_{\boldsymbol{r}_{id}^k})^T (\boldsymbol{r}_i - \boldsymbol{r}_{id}^k), \quad (6)$$

[1] It should be stressed out that the semi-cooperative, on-the-fly (dynamic) prioritization is not the same with the one performed in the high-level planning mode, where the priorities of the agents are static and preassigned.

Algorithm 1 Prioritized $A^\star$ Algorithm

Re-plan Decision
if Agent with higher priority joins the group **then**
  if Agents with higher priorities in group have finished path planning **then**
    goto **Path Finding**
  else
    return
else
  return
Path Finding
procedure $A^\star$ PATH FINDING
  find the agent's current node, labeled N_{start}
  $N_{current} \leftarrow N_{start}$
loop:
  search the passable nodes around $N_{current}$:
  for node N_i found:
  if the destination is in N_i **then**
    $N_{i,parent} \leftarrow N_{current}$
    $N_{end} \leftarrow N_i$
    break loop
  else if N_i is UNCHECK **then**
    if $N_{current}$ is not occupied with other agents with higher priority **then**
      set N_i to OPEN
      $N_{i,parent} \leftarrow N_{current}$
      calculate F_i of N_i
  else if N_i is OPEN **then**
    if $G_i(N_{current}) < G_i(N_{i,parent})$ **then**
      $N_{i,parent} \leftarrow N_{current}$
    calculate F_i of N_i
  after searching all nodes around:
  set $N_{current}$ as CLOSE
  $N_{next} \leftarrow$ the node in OPEN with lowest F.
  $N_{current} \leftarrow N_{next}$
end loop
  $N_{current} \leftarrow N_{end}$
  while $N_{current} \neq N_{start}$ **do**
    save $N_{current}$ as the first node in the agent's waypoint list
    $N_{current} \leftarrow N_{current,parent}$
  save the destination as the last node in the agent's waypoint list

where $(\nabla b_{ij}|_{\boldsymbol{r}_{id}^k})^T$ is the transpose of the gradient vector of the barrier function (5), evaluated at $\boldsymbol{r}_{id}^k$. To obtain a positive-definite function, we take:

$$V'_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = r_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j)^2, \tag{7}$$

and finally, in order to encode avoidance only w.r.t. the agents $j \neq i$ that lie in the sensing region, we define:

$$V_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = \sigma_{ij} V'_{ij}, \tag{8}$$

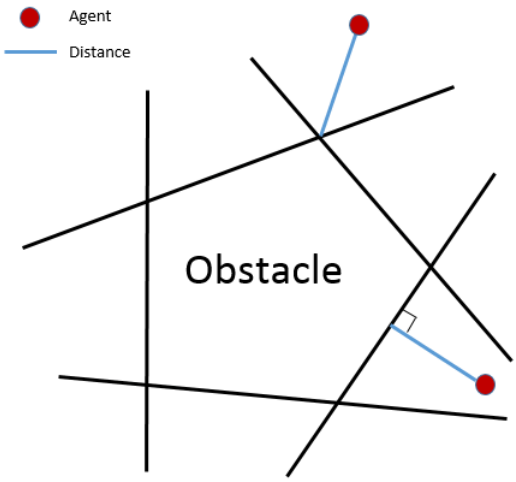


Fig. 3. Area division around a 2D obstacle.

where:

$$\sigma_{ij} = \begin{cases} 1, & d_s \leq d_{ij} \leq R_z, \\ Ad_{ij}^3 + Bd_{ij}^2 + Cd_{ij} + D, & R_z < d_{ij} < R_s, \\ 0, & d_{ij} \geq R_s. \end{cases} \tag{9}$$

The coefficients in (9) have been computed equal to: $A = -\frac{2}{(R_z-R_s)^3}$, $B = \frac{3(R_z+R_s)}{(R_z-R_s)^3}$, $C = -\frac{6R_zR_s}{(R_z-R_s)^3}$, $D = \frac{R_z^2(3R_z-R_s)}{(R_z-R_s)^3}$, so that (9) is a continuously differentiable function.

A. Barriers for Polygonal Environments

We extend the method in [42] to polygonal 2D and 3D environments populated with M static obstacles. We assume that each obstacle o_m, $m \in \{1, \ldots, M\}$ is an arbitrary convex polygon, and that its geometry is known through a map of the environment. Given a polygonal obstacle with p edges, we extend the edges of obstacles and divide the nearby area into $2p$ regions. We define the distance d_{iom} between the agent i and the obstacle o_m according to different regions in which the agent is, see Figure 3, as follows:

- In the regions that include an edge of the obstacle, d_{iom} is defined as the distance between this edge and the agent.
- In the regions that include a vertex but not edges of the obstacle, d_{iom} is defined as the distance between the vertex and the agent.

An example on defining the distance between an agent i and a rectangular obstacle o_j with 4 vertices whose coordinates are $\boldsymbol{r}_{o11} = \begin{bmatrix} x_{oj1} & y_{oj1} \end{bmatrix}^T$, $\boldsymbol{r}_{o12} = \begin{bmatrix} x_{oj1} & y_{oj2} \end{bmatrix}^T$, $\boldsymbol{r}_{o21} = \begin{bmatrix} x_{oj2} & y_{oj1} \end{bmatrix}^T$ and $\boldsymbol{r}_{o22} = \begin{bmatrix} x_{oj2} & y_{oj2} \end{bmatrix}^T$, where $x_{oj1} < x_{oj2}$ and $y_{oj1} < y_{oj2}$, is given as follows:

$$d_{ioj} = \begin{cases} x_{oj1} - x_i, & \text{if} \quad (x_i < x_{oj1}) \wedge (y_{oj1} < y_i < y_{oj2}), \\ x_i - x_{oj2}, & \text{if} \quad (x_i > x_{oj2}) \wedge (y_{oj1} < y_i < y_{oj2}), \\ y_{oj1} - y_i, & \text{if} \quad (y_i < y_{oj1}) \wedge (x_{oj1} < x_i < x_{oj2}), \\ y_i - y_{oj2}, & \text{if} \quad (y_i > y_{oj2}) \wedge (x_{oj1} < x_i < x_{oj2}), \\ \|\boldsymbol{r}_i - \boldsymbol{r}_{o11}\|, & \text{if} \quad (y_i \leq y_{oj1}) \wedge (x_i \leq x_{oj1}), \\ \|\boldsymbol{r}_i - \boldsymbol{r}_{o12}\|, & \text{if} \quad (y_i \geq y_{oj2}) \wedge (x_i \leq x_{oj1}), \\ \|\boldsymbol{r}_i - \boldsymbol{r}_{o21}\|, & \text{if} \quad (y_i \leq y_{oj1}) \wedge (x_i \geq x_{oj2}), \\ \|\boldsymbol{r}_i - \boldsymbol{r}_{o22}\|, & \text{if} \quad (y_i \geq y_{oj2}) \wedge (x_i \geq x_{oj2}), \end{cases}$$

with the resulting function d_{ioj} depicted in Figure 4.

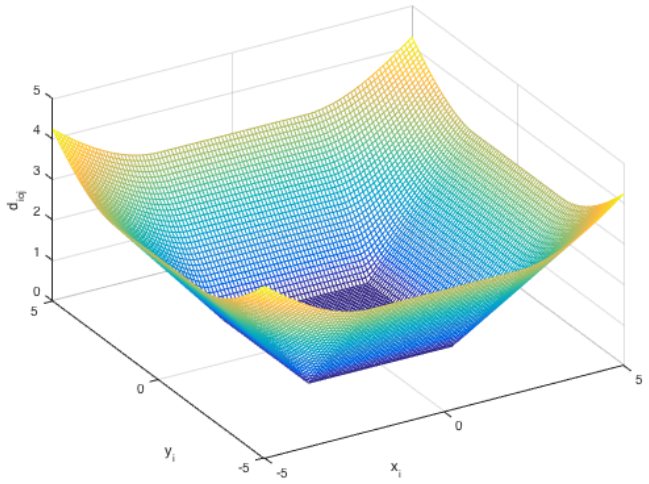


Fig. 4. Distance function d_{ioj} with $x_{oj1} = -2$, $x_{oj2} = 2$, $y_{oj1} = -2$, and $y_{oj2} = 2$

We can then define a barrier function V'_{ioj} encoding agent-obstacle avoidance:

$$V'_{ioj} = (b_{ioj}(\boldsymbol{r}_i, \boldsymbol{r}_{o_j}))^2 = (-\ln(d_{ioj}(\boldsymbol{r}_i, \boldsymbol{r}_{o_j})^2))^2, \quad (10)$$

where $\boldsymbol{r}_{o_j}$ is the vector comprising the position coordinates of the vertices of the obstacle. To suppress the effect of the obstacle barrier (10) to be active only when the obstacle o_j is detected in the sensing range of the agent i, we define:

$$V_{ioj}(\boldsymbol{r}_i, \boldsymbol{r}_{o_j}) = \sigma_{ioj} V'_{ioj}, \quad (11)$$

where σ_{ioj} is a bump function defined same as (9), with d_{ij} being substituted by d_{ioj}.

With all the agent-to-agent and agent-to-obstacle barriers at hand, the function encoding safety for an agent i moving in an N-agent environment with M static obstacles is:

$$\nu_i = \sum_{j=1, j\neq i}^{N} V_{ij} + \sum_{k=1}^{M} V_{iok}. \quad (12)$$

We scale it to vary between 0 and 1 as:

$$V_i = \frac{\nu_i}{1+\nu_i}. \quad (13)$$

Note that the use of the bump functions σ_{ij} and σ_{ioj} in (8) and (11), respectively, results in agent i creating a barrier w.r.t. the detected nearby static obstacles and its neighbor agents $j \in \mathcal{N}_i$ only, where $\mathcal{N}_i$ denotes the set of agents lying in the sensing region of agent i.

Finally, we notice that when $\left\|\boldsymbol{r}^k_{id} - \boldsymbol{r}_j\right\| < d_s$, the terms $b_{ij}(\boldsymbol{r}^k_{id}, \boldsymbol{r}_j)$ and $(\nabla b_{ij}|_{\boldsymbol{r}^k_{id}})^T$ in (6) are undefined. This condition arises if neighbor agents $j \in \mathcal{N}_i$ happen to lie sufficiently close to the current destination $\boldsymbol{r}^k_{id}$ of agent i. To overcome this situation, we consider the unit vector $\hat{\boldsymbol{\eta}}$, whose direction coincides with the line-of-sight from the agent's current node to the next node in the waypoint list. Therefore, if some agent $j \neq i$ is close enough to the current goal location $\boldsymbol{r}^k_{id}$ of agent i so that (6) becomes undefined, then the current waypoint $\boldsymbol{r}^k_{id}$ is replaced with the waypoint $\boldsymbol{r}^n_{id} = \boldsymbol{p}_{N_i} + 3R_s\hat{\boldsymbol{\eta}}$, where $\boldsymbol{p}_{N_i}$ is the agent's current node position. Note that the new waypoint $\boldsymbol{r}^n_{id}$ lies out of the sensing radius R_s of agent i, where no agents j are sensed; this way we avoid both the undesired case in the definition of the barrier function for agent i, while we ensure that the agent keeps moving along its current preferred direction.

B. Design of the Low-Level Controller

The direction of motion of agent i is set to be along the negated gradient vector of (13):

$$\gamma_i = \operatorname{atan2}\left(-\frac{\partial V_i}{\partial y_i}, -\frac{\partial V_i}{\partial x_i}\right). \quad (14)$$

In order to determine control commands that generate collision-free trajectories for each agent i, we consider a standard decoupling between the kinematic subsystem (2a) and the dynamic subsystem (2b), and treat the problem as follows: We first design a desired ("virtual") controller for the magnitude v_{id} of the linear velocity vector $\boldsymbol{v}_{id}$ of each agent i, that ensures the generation of collision-free trajectories that are additionally almost globally convergent to the desired destination of each agent i. We then consider the error $\boldsymbol{v}_{ie}$ between the actual $\boldsymbol{v}_i$ and the desired $\boldsymbol{v}_{id}$ linear velocity of agent i, and design an acceleration controller $\boldsymbol{a}_i$ that exponentially drives this error to zero. Safety guarantees are preserved by considering the maximum position error $\boldsymbol{r}_{ie}$ induced by the discrepancy between the actual velocity vector $\boldsymbol{v}_i$ and the desired velocity vector $\boldsymbol{v}_{ie}$, and adding it to the safety zone of each agent.

1) Velocity (kinematic) control: In order to design the desired velocity command $\boldsymbol{v}_{id}$ that generates collision-free position trajectories for the kinematic subsystem (2a) of each agent i, we build upon the control design in [42].

Theorem 1: Assume N agents $i \in \{1, 2, \ldots, N\}$ moving along the direction (14) dictated by their individual Lyapunov-like barrier functions (13). If the magnitude $v_{id} = \|\boldsymbol{v}_{id}\|$ of the desired linear velocity vector $\boldsymbol{v}_{id}$ of agent i is set as:

$$v_{id} = \begin{cases} -\frac{1}{\mu}\log\left(\sum\limits_{j\in\mathcal{N}_i|J_j<0} e^{-\mu\, v_{i|j}}\right) & d_s \leq d_{ij} \leq R_c, \\ v_{ic} & d_{ij} > R_c, \end{cases} \quad (15)$$

where $v_{i|j}$ denotes the velocity adjustment mechanism of agent i w.r.t. agent j, defined as:

$$v_{i|j} = v_{ic}\frac{d_{ij} - d_s}{R_c - d_s} + v_{is|j}\frac{R_c - d_{ij}}{R_c - d_s}, \quad (16)$$

with the terms in (16) defined as:

$$v_{ic} = k_i \tanh\left(\|\boldsymbol{r}_i - \boldsymbol{r}_{id}\|\right), \quad (17a)$$

$$v_{is|j} = \varepsilon_i\, v_{jd}\, \frac{\boldsymbol{r}_{ji}{}^T \hat{\boldsymbol{v}}_{jd}}{\boldsymbol{r}_{ji}{}^T \hat{\boldsymbol{v}}_{id}}, \quad (17b)$$

$$J_j = \boldsymbol{r}_{ji}{}^T \hat{\boldsymbol{v}}_{id}, \quad (17c)$$

$0 < \varepsilon_i < 1$, $\boldsymbol{r}_{ji} = \boldsymbol{r}_i - \boldsymbol{r}_j$, and $\hat{\boldsymbol{v}}_{id} = \frac{\boldsymbol{v}_{id}}{v_{id}} \triangleq \begin{bmatrix}\cos\gamma_i \\ \sin\gamma_i\end{bmatrix}$, γ_i given out of (14), and where for a given vector $a = \begin{bmatrix}a_1, & \ldots, a_n\end{bmatrix}^T$, the function $g(a) = -\frac{1}{\mu}\log\left(\sum\limits_{i=1}^{n} e^{-\mu a_i}\right)$ is a smooth approximation function of the minimum function $\min\{a_1, \ldots, a_n\}$,

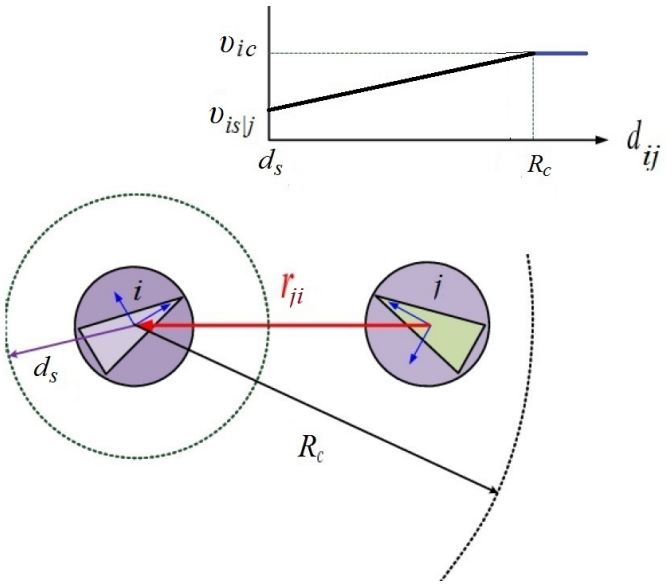


Fig. 5. If $J_j \triangleq {\boldsymbol{r}_{ji}}^T \hat{\boldsymbol{v}}_{\boldsymbol{i}} < 0$, i.e., if agent i moves towards agent j, then agent i adjusts its linear velocity according to the velocity profile shown here, given analytically by (15).

with $\mu \to \infty$, then the motion of agent i is collision-free w.r.t. all neighbor agents $j \in \mathcal{N}_i$.

Remark 1: The set $\{j \in \mathcal{N}_i \mid J_j < 0\}$ denotes the neighbor agents j whom the agent i is approaching on its current direction. This term dictates what we call the "on-the-fly (dynamic) prioritization" among agents: agent i ignores the neighbor agents he/she is moving away from, and decides to adjust its linear velocity (speed) w.r.t. the worst-case neighbor agent, i.e., w.r.t. the neighbor j he/she is more susceptible to collide with.

Proof: Let us assume that at some time $t \geq 0$ the distance $d_{ij}(t)$ between a pair of agents (i, j) is $d_{ij}(t) \leq R_c$. Collision-free motion is realized as ensuring that $\dot{d}_{ij}(t) > d_s$, $\forall t \geq 0$, for any pair of agents (i, j). Consider the time derivative of the inter-agent distance function, which after some calculations reads:

$$\begin{aligned} \dot{d}_{ij} &= \frac{(x_i - x_j)(\dot{x}_i - \dot{x}_j)}{d_{ij}} + \frac{(y_i - y_j)(\dot{y}_i - \dot{y}_j)}{d_{ij}} \\ &= \frac{v_{id}\, {\boldsymbol{r}_{ji}}^T \hat{\boldsymbol{v}}_{id} - v_{jd}\, {\boldsymbol{r}_{ji}}^T \hat{\boldsymbol{v}}_{jd}}{d_{ij}}. \end{aligned} \tag{18}$$

Agent i adjusts its linear velocity v_{id} according to the velocity profile shown in Fig. 5, given analytically out of (15); under this velocity adjustment the distance d_{ij} w.r.t. the worst-case neighbor j remains greater than d_s. First, let us explain the notion of the worst-case neighbor: By the worst-case neighbor we mean the agent $j \in \{\mathcal{N}_i \mid J_j < 0\}$ towards whom the rate of change of the relative distance d_{ij} given by (18), due to the motion of agent i, is maximum. More specifically: The term $J_j < 0$ describes the set of agents $j \in \mathcal{N}_i$ towards whom agent i is moving in its current direction [42]. Agent i computes safe velocities $v_{i|j}$ w.r.t. each neighbor $j \in \{\mathcal{N}_i \mid J_j < 0\}$, and picks the minimum $\min\{v_{i|j}\}$ among the safe velocities so that the first term in (18) is as less negative as possible. Now, the value of the safe velocity $v_{i|j}$ (15) when $d_{ij} = d_s$ is by construction equal to (17b). Plugging (17b) into (18) reads:

$$\dot{d}_{ij} = \frac{(\varepsilon_i - 1) v_{jd}\, {\boldsymbol{r}_{ji}}^T \hat{\boldsymbol{v}}_{jd}}{d_{ij}} \geq 0.$$

To see why this condition is true, recall that $\varepsilon_i - 1 < 0$, $v_{jd} \geq 0$, and ${\boldsymbol{r}_{ji}}^T \hat{\boldsymbol{v}}_{jd} \leq 0$: this is because agent j is either following the gradient of its own Lyapunov-like barrier function V_j that by construction penalizes collision with agent i, or happens to move away from agent i in the first place. This implies that the inter-agent distance d_{ij} can not become less than d_s, hence collisions are avoided. Furthermore, for μ sufficiently large, the minimum function $\min\{v_{i|j}\}$ is approximated from below by the function (15). This completes the proof. ■ ■

2) Acceleration (dynamic) control:

Theorem 2: Assume N agents $i \in \{1, 2, \ldots, N\}$ moving along the direction (14) dictated by their individual Lyapunov-like barrier functions (13) under the acceleration controller

$$\boldsymbol{a}_i = \dot{\boldsymbol{v}}_{id} - \lambda_i (\boldsymbol{v}_i - \boldsymbol{v}_{id}), \tag{19}$$

where $\lambda_i > 0$ and $\boldsymbol{v}_{id}$ the desired linear velocity vector of agent i along the direction (14) with magnitude given out of (15). If the safe separation of each agent i is taken as $d_s + 2\|\boldsymbol{r}_{ieMax}\|$, with $\|\boldsymbol{r}_{ieMax}\|$ given out of (25), then the motion of all agents is collision-free.

Proof: Let us define the vector of velocity error $\boldsymbol{v}_{ie}$ as:

$$\boldsymbol{v}_{ie} = \boldsymbol{v}_i - \boldsymbol{v}_{id}, \tag{20}$$

where $\boldsymbol{v}_{id}$ is the desired velocity vector along the direction of $-\nabla V_i$, with magnitude equal to (15). The time derivative of the velocity error reads:

$$\dot{\boldsymbol{v}}_{ie} = \dot{\boldsymbol{v}}_i - \dot{\boldsymbol{v}}_{id} \overset{(2)}{=} \boldsymbol{a}_i - \dot{\boldsymbol{v}}_{id}. \tag{21}$$

Substituting (19) into (21) yields:

$$\dot{\boldsymbol{v}}_{ie} = -\lambda_i (\boldsymbol{v}_i - \boldsymbol{v}_{id}) = -\lambda_i \boldsymbol{v}_{ie}, \tag{22}$$

where $\lambda_i > 0$, i.e., the velocity error $\boldsymbol{v}_{ie}$ is globally exponentially stable.

Integrating equation (22) furthermore yields:

$$\boldsymbol{v}_{ie}(t) = e^{-\lambda_i t} \boldsymbol{v}_{ie}(0) \tag{23}$$

where $\boldsymbol{v}_{ie}(0)$ is the initial velocity error, while by integrating equation (23) we obtain the position error $\boldsymbol{r}_{ie}(t)$ as:

$$\boldsymbol{r}_{ie}(t) = \frac{1}{\lambda_i} \boldsymbol{v}_{ie}(0)(1 - e^{-\lambda_i t}). \tag{24}$$

The maximum position error $\boldsymbol{r}_{ieMax}$ is reached at $t = \infty$ and has the form of:

$$\boldsymbol{r}_{ieMax} = \frac{1}{\lambda_i} \boldsymbol{v}_{ie}(0). \tag{25}$$

As expected, the larger the gain λ_i, the smaller the induced position error $\boldsymbol{r}_{ieMax}$ during the transient dynamics of the velocity error $\boldsymbol{v}_{ie}(t)$. Thus, if the safe separation of each agent i is enlarged by $2\|\boldsymbol{r}_{ieMax}\|$, the resulting agent trajectories are collision-free.

In order to evaluate $\boldsymbol{r}_{ieMax}$, first we note that the norm $\|\boldsymbol{v}_{ie}(0)\|$ of the maximum possible velocity error $\boldsymbol{v}_{ie}(0)$ is equal to $2k_i$, where k_i is the maximum velocity magnitude of (15); this corresponds to agent i changing instantaneously its direction of motion. Now, the time constant λ_i is related to the acceleration ability of the agent. Let us denote the magnitude of the maximum acceleration that can be achieved by each quadrotor by a_{max}. From Equation (19), we have that the following inequality should hold:

$$\boldsymbol{a}_i{}^T\boldsymbol{a}_i = \dot{\boldsymbol{v}}_{id}{}^T\dot{\boldsymbol{v}}_{id} + \lambda_i{}^2\boldsymbol{v}_{ie}{}^T\boldsymbol{v}_{ie} - 2\lambda_i\dot{\boldsymbol{v}}_{id}{}^T\boldsymbol{v}_{ie} \leq a_{max}{}^2.$$

Noting that $\dot{\boldsymbol{v}}_{id}^T\dot{\boldsymbol{v}}_{id} = \dot{v}_{id}^2$, and $\boldsymbol{v}_{ie}{}^T\boldsymbol{v}_{ie} = v_{ie}{}^2$, we can write:

$$\dot{v}_{id}^2 + \lambda_i{}^2 v_{ie}^2 - 2\lambda_i\dot{v}_{id}v_{ie}\cos\langle\boldsymbol{v}_{id}, \boldsymbol{v}_{ie}\rangle \leq a_{max}{}^2. \qquad (26)$$

The worst-case can be considered for the upper bounds $v_{ie} \leq 2k_i$, $\dot{v}_{id} \leq \frac{2k_i}{\Delta t}$, for some given $\Delta t > 0$, and for $\cos\langle\boldsymbol{v}_{id}, \boldsymbol{v}_{ie}\rangle = -1$, so that (26) reads:

$$\left(\frac{2k_i}{\Delta t}\right)^2 + \lambda_i{}^2(2k_i)^2 + 2\lambda_i\frac{2k_i}{\Delta t}2k_i \leq a_{max}{}^2 \Rightarrow$$

$$4k_i{}^2\left(\frac{1}{\Delta t} + \lambda_i\right)^2 \leq a_{max}{}^2. \qquad (27)$$

Hence for a given bound a_{max} on the agent's acceleration, the condition (27) provides a sufficient way to pick the control gains k_i, λ_i of the velocity and acceleration controllers (15) and (19), respectively. ■

C. Main Algorithm

A pseudocode of our main algorithm comprising the high-level planning and the low-level coordination and control is given below as Algorithm 2.

Algorithm 2 Main Algorithm

procedure MAIN ALGORITHM
for each agent:
    generate node map
loop:
    global path planning:
    Prioritized $A^\star$ Algorithm
    local path planning:
    if other agents in sensing area **then**
        $\hat{\boldsymbol{\eta}} \leftarrow \boldsymbol{r}_{id} - \boldsymbol{p}_{N_i}$, where $\boldsymbol{p}_{N_i}$ is the agent's current node position
        change $\boldsymbol{r}_{id}$ to $\boldsymbol{p}_{N_i} + 3R_s\hat{\boldsymbol{\eta}}$.
    turn on barrier functions and calculate velocity vector
end loop

V. Extension to 3D Environments

In a 3D environment, the agents (aerial robots) are allowed to change altitude. Each agent i is subject to double-integrator dynamics given out of (2), with the position, velocity and acceleration vectors $\boldsymbol{r}_i, \boldsymbol{v}_i, \boldsymbol{a}_i \in \mathbb{R}^3$ of agent i expressed w.r.t. the inertial frame. Denote $\boldsymbol{r}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$. All agents are modeled as spheres of radius r_a, and their sensing and communication regions are modeled as spherical regions as well of the same radii as in the 2D case.

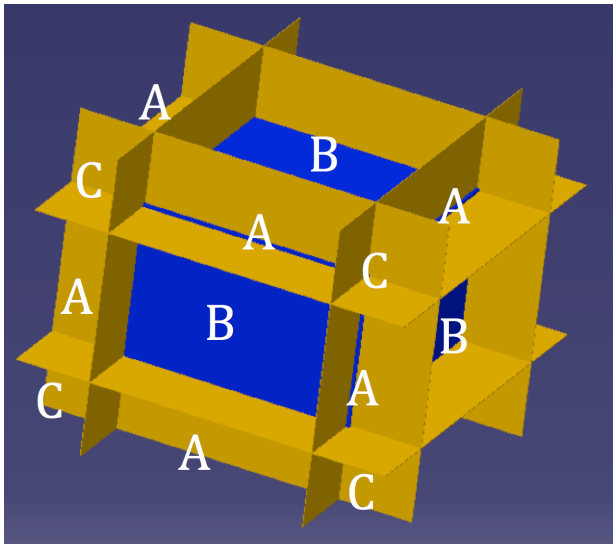


Fig. 6. Space division into regions of type A, B, C around a 3D obstacle.

For the high-level path planning, the environment is divided into uniform cubes, and nodes are located in the center of corresponding cubes. For a given node N_i, its parameters H_i, G_i used in the $A^\star$ planner are rewritten as: $H_i = \sqrt{(x_{N_i} - x_{id})^2 + (y_{N_i} - y_{id})^2 + (z_{N_i} - z_{id})^2}$, $G_i(N_j) = G_j + \sqrt{(x_{N_i} - x_{N_j})^2 + (y_{N_i} - y_{N_j})^2 + (z_{N_i} - z_{N_j})}$.

For the low-level control design, we first extend the construction of the Lyapunov-like barriers in 3D environments. For penalizing the inter-agent distances from becoming less than the safe separation, we consider the constraint function $c_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - d_s{}^2 > 0$, and follow the exact same procedure on the definition of the recentered barrier functions as in the 2D case. Now, given a 3D polyhedron o_m, we extend the obstacle surfaces to divide the surrounding space into regions as shown in Figure 6.

The distance d_{ioj} between the agent i and the obstacle o_j is now defined based on the region in which the agent is, as follows:

- In the regions that include an edge of the obstacle (type A), d_{ioj} is defined as the distance between the position of agent to the line the edge lies on.
- In regions that include a surface of the obstacle (type B), d_{ioj} is defined as the distance between the surface and the agent.
- In the regions that include a vertex of the obstacle (type C), d_{ioj} is defined as the distance between the vertex and the agent.

With the agent-to-agent and agent-to-obstacle barriers at hand, we can define the Lyapunov-like barrier function V_i, and consider the safe direction γ_i for agent i to be along the gradient vector $\nabla V_i = \begin{bmatrix} -\frac{\partial V_i}{\partial x_i} & -\frac{\partial V_i}{\partial y_i} & -\frac{\partial V_i}{\partial z_i} \end{bmatrix}^T$.

VI. Simulation Results

The efficacy of the proposed approach is demonstrated through simulation results in both 2D and 3D environments. The computer simulations were performed with the

Gazebo simulator in ROS (Robot Operating System). The hector_quadrotor package in ROS was used to setup a 2D and 3D simulation environment with quadrotors. This package includes built-in noise, filters, sensors, and velocity control, that make it appropriate to demonstrate the proposed algorithm. All simulations were run with a control loop frequency of 20Hz.

A. 2D case

We consider a known 2D workspace of dimensions 40 m $\times$ 40 m, which is divided into $20 \times 20 = 400$ uniform grids of width $w = 2$ m. The values of the system parameters that were used in the simulation are: $R_s = 1.5$ m, $R_z = 1.4$ m, $R_c = 1.3$ m, $d_s = 1.2$ m, $k_i = 1$. We demonstrate three scenarios involving 20 agents (Fig. 7, 8, 9). The initial locations of the agents are depicted as blue circles, while their goal locations are depicted as green circles. The actual final positions of the agents in each simulated scenario are marked as red circles. Thus a red circle that is not in a green circle denotes the location of an agent that could not reach its desired destination after a sufficiently long simulation time. For each simulated scenario we compare the results generated by the prioritized $A^\star$ algorithm with those generated by the standard $A^\star$ algorithm; recall that the high-level planners are in every case combined with the low-level coordination controller that is based on barriers. The comparison demonstrates the effectiveness of the prioritized $A^\star$ algorithm over the standard $A^\star$ algorithm in terms of improving the quality of the resulting paths, while being successful in finding safe and convergent paths, even in cases when the standard $A^\star$ fails to do so; these cases arise in high-density operations where the agents congest in confined areas. The maximum and average computation time for both the prioritized $A^\star$ and the standard $A^\star$ are also recorded and discussed. Each scenario is described in detail in what follows. Note that for the case of applying the standard $A^\star$ algorithm, all agents plan their paths to their desired goal locations at the beginning of the simulation, and then follow the computed paths under the low-level controller. However, as the simulation proceeds, an agent could encounter other agents; in this situation, the low-level controller ensures the collision avoidance. The side-effect is that the agent might go to a cell that is not on its original planned path in order to avoid collisions. In this situation, the agent replans its path using the standard $A^\star$ algorithm.

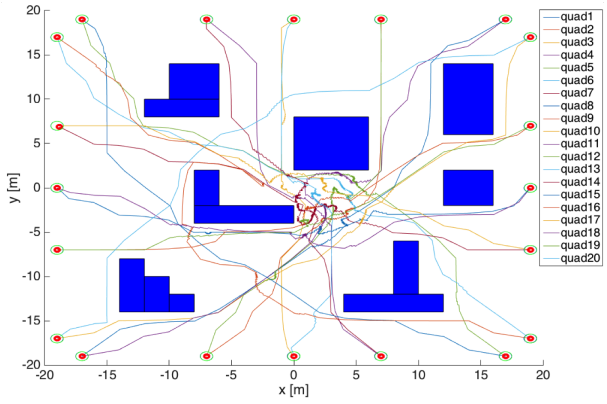
1) Scenario 1: The first scenario illustrates a case where the standard $A^\star$ fails to find a safe and convergent path for each agent when they happen to operate in a highly congested confined area, whereas the prioritized $A^\star$ succeeds in generating both safe and convergent paths. The results are shown in Fig. 7.

The initial locations of the agents are such that the agents are disconnected, while their final locations are such that the straight-line paths intersect in the area around the centroid of the considered workspace. Each agent computes a path to its goal using the proposed prioritized $A^\star$ algorithm and moves towards it under the control laws (15), (19). Obviously, since the agents are initially disconnected, the prioritized $A^\star$ essentially behaves as the standard $A^\star$ in the first run of the algorithm and while there are no connected groups. The $A^\star$ paths guide the agents in the area close to the centroid of the considered workspace. Whenever agents become pairwise connected, the prioritized $A^\star$ is triggered, as described in Section III. The paths resulting from the prioritized $A^\star$ algorithm in combination with the low-level barrier control are shown in Fig. 7(a), while the resulting paths for the case when the standard $A^\star$ is used instead of the prioritized $A^\star$ is shown in Fig. 7(b). The prioritized $A^\star$ forces the connected agents to pick paths in a sequential spirit that are not overlapping for a short horizon ahead, which is dictated by the number n of nodes of the paths of the agents of higher priority that are communicated to, and avoided by, the agents of lower priority. Hence conflicts are resolved along paths that suppress congestion, and the agents are driven to their goal destinations. In comparison, the standard $A^\star$ per connected agent does not take into account the other agents or their nodes, resulting hence in overlapping paths, and defers the conflict resolution to the low-level barrier control. This in turn results in chattering, hence longer, and for some agents non-convergent paths, as shown in Fig. 7(b). Safety under the prioritized $A^\star$ is demonstrated through Fig. 7(c) that depicts the minimum pairwise inter-agent distance among all pairs of connected agents over time, and verifies that it remains greater than the safe separation. Finally the resulting speeds for the agents are shown in Fig. 7(d), and are bounded as expected by the definition of the low-level controller.

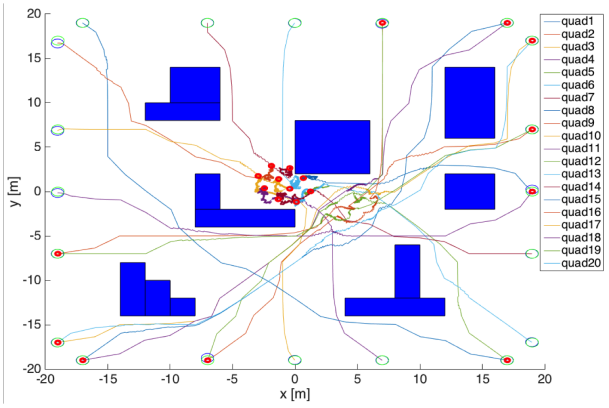
2) Scenario 2: The second scenario illustrates a case where the initial locations of the agents are such that a congested area, and in turn connected groups of agents, are formed in the neighborhood of the centroid of the considered workspace at initial time $t = 0$. These initial conditions trigger the prioritized $A^\star$. As shown in Fig. 8(a), the agents manage to resolve conflicts and converge to their goal destinations. In contrast, the standard $A^\star$ (in combination with the low-level barrier control) fails to find convergent paths for all agents, as illustrated in Fig. 8(b). Furthermore, inter-agent collisions under the prioritized $A^\star$ are avoided as demonstrated by the minimum pairwise distance illustrated in Fig. 8(c), while the resulting speeds for the agents are shown in Fig. 8(d).

3) Scenario 3: The third scenario illustrates a case where both the prioritized $A^\star$ and the standard $A^\star$, combined as always with the barrier control, succeed in finding safe and convergent paths, yet the prioritized $A^\star$ results in shorter traveled paths. The results of this simulation are shown in Fig. 9. The agents' initial and final locations in this case are chosen such that no areas of high congestion are formed; as a result, both the prioritized and the standard $A^\star$ along with the barriers find collision-free and convergent paths. Nevertheless, the total path length in the former case is shorter than in the latter case, see Table I.

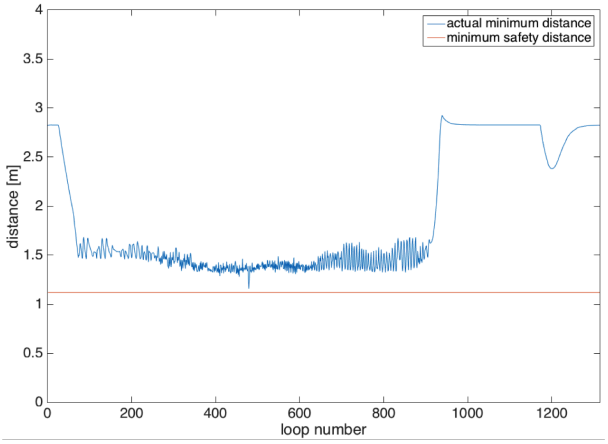
Table I in addition provides the average and maximum loop time for the prioritized and the standard $A^\star$, respectively, as well as the total path length traveled by the agents in the presented three scenarios. Note that the values of the total length marked by "*" refer to the cases where not all agents reached their goals, and the simulation was forced to stop after running for a sufficiently long time.



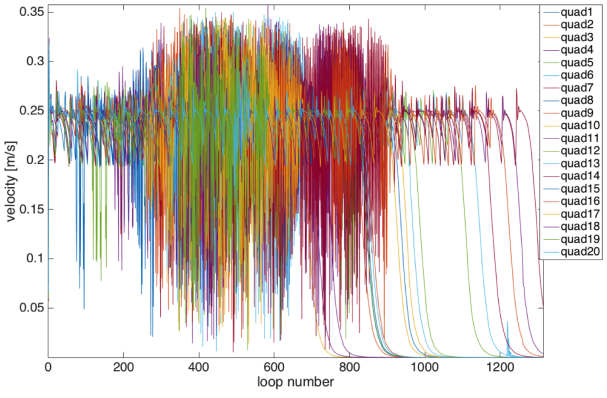
(a) Agents' paths under prioritized $A^\star$ and barriers.



(b) Agents' paths under standard $A^\star$ and barriers.

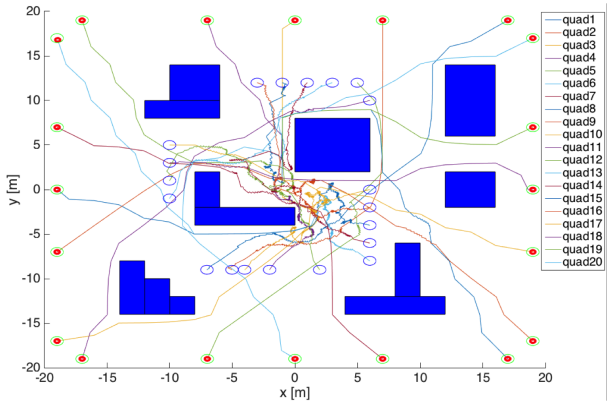


(c) Agents' minimum pairwise distance under prioritized $A^\star$ and barriers.

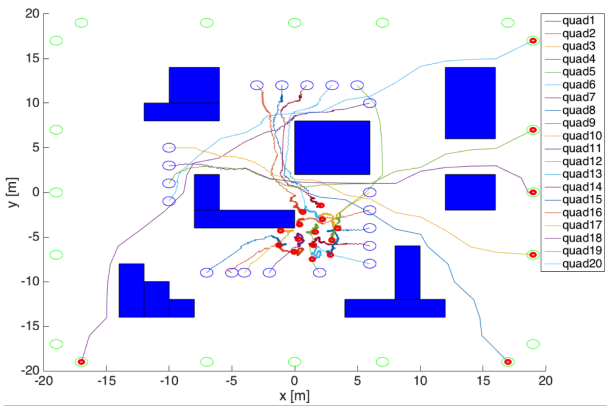


(d) Agents' speeds under prioritized $A^\star$ and barriers.

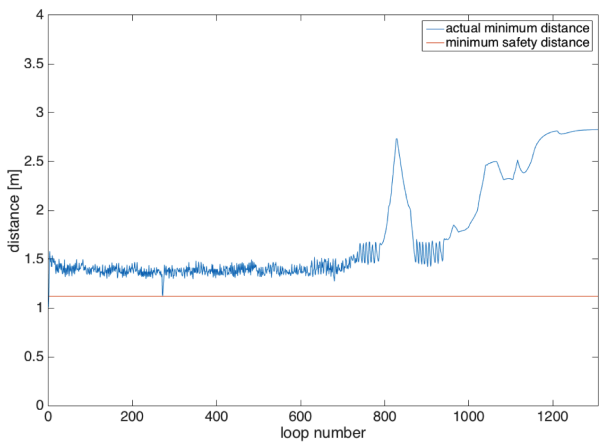
Fig. 7. 2D case - Simulation Results - Scenario 1.



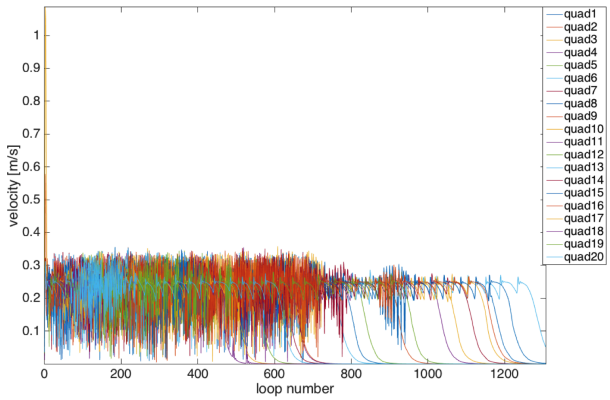
(a) Agents' paths under prioritized $A^\star$ and barriers.



(b) Agents' paths under standard $A^\star$ and barriers.

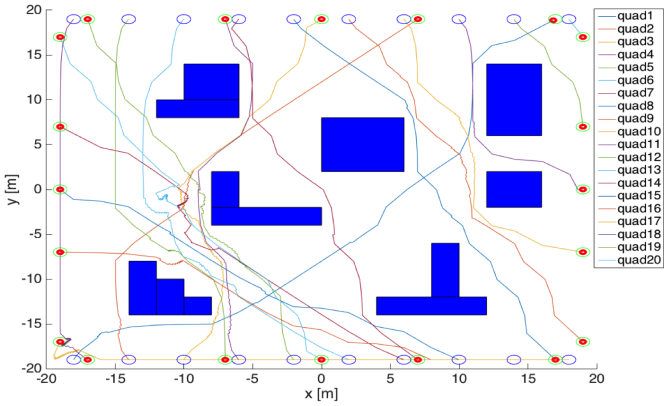


(c) Agents' minimum pairwise distance under prioritized $A^\star$ and barriers.

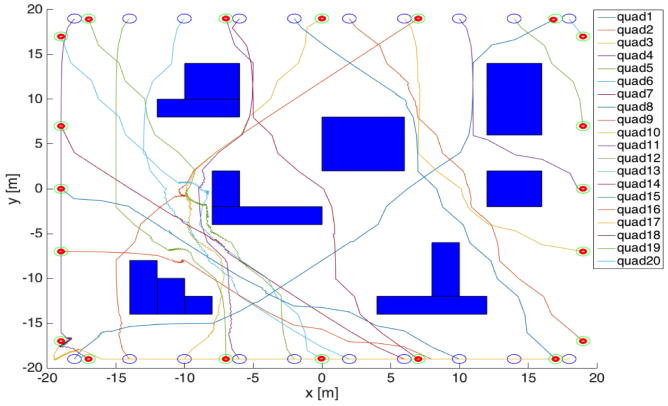


(d) Agents' speeds under prioritized $A^\star$ and barriers.

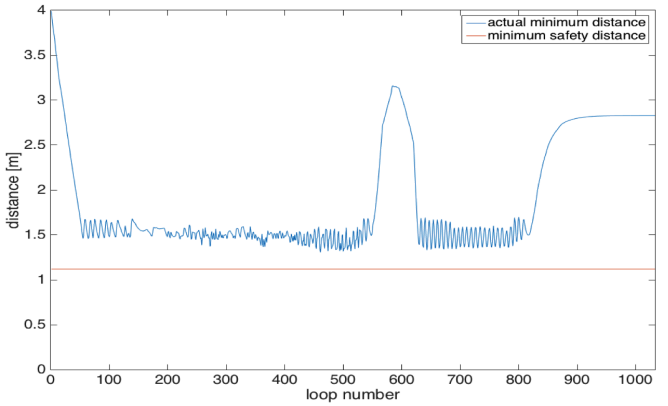
Fig. 8. 2D case - Simulation Results - Scenario 2.



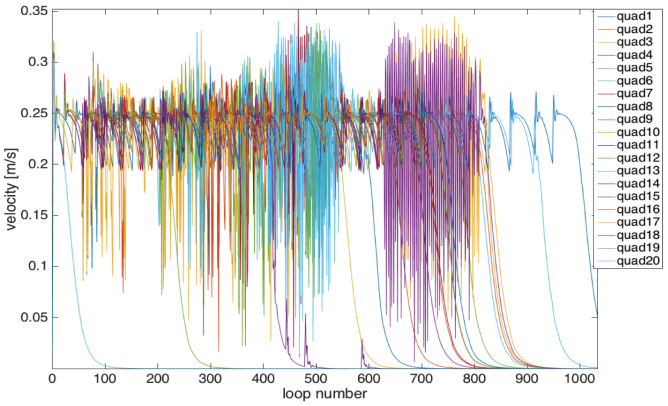
(a) Agents' paths under prioritized $A^\star$ and barriers.



(b) Agents' paths under standard $A^\star$ and barriers.



(c) Agents' minimum pairwise distance under prioritized $A^\star$ and barriers.



(d) Agents' speeds under prioritized $A^\star$ and barriers.

Fig. 9. 2D case - Simulation Results - Scenario 3.

TABLE I
AVERAGE AND MAXIMUM LOOP TIME AND TOTAL PATH LENGTH OF 2D SIMULATION

Scenario	Algorithm	Average Loop Time [s]	Maximum Loop Time [s]	Total Path Length [m]
1	Prioritized $A^\star$	0.0137	0.0404	1249.7
1	Standard $A^\star$	0.0044	0.0377	2481.8*
2	Prioritized $A^\star$	0.0136	0.0385	1115.4
2	Standard $A^\star$	0.0050	0.0362	1501.6*
3	Prioritized $A^\star$	0.0120	0.0289	792.1
3	Standard $A^\star$	0.0051	0.0421	823.5

4) Summary: Summarizing the simulation results: In scenarios 1 and 2, either the standard $A^\star$ (scenario 1) or the initial conditions (scenario 2) resulted in traffic congestion around the centroid of the considered environment; The use of the standard $A^\star$ along with a barrier low-level control did not manage to resolve the congestion, hence some agents were not able to reach their goal destinations, whereas the proposed prioritized algorithm along with the low-level barrier control was successful in finding collision-free and convergent paths. In scenario 3, both algorithms succeeded in goal reaching, yet the prioritized $A^\star$ yielded shorter total path length. The pairwise distance plots also show that the low-level barrier controller successfully maintained the inter-agent distances among all pairs of agents always greater than the safe separation. The speed of the agents was also bounded according to the results. However, since path re-planning is triggered more often in the prioritized $A^\star$ case, and since the planner accounts for both static and dynamic obstacles (other agents), the average computation time for the proposed algorithm is higher than that of the standard $A^\star$ algorithm.

B. 3D case

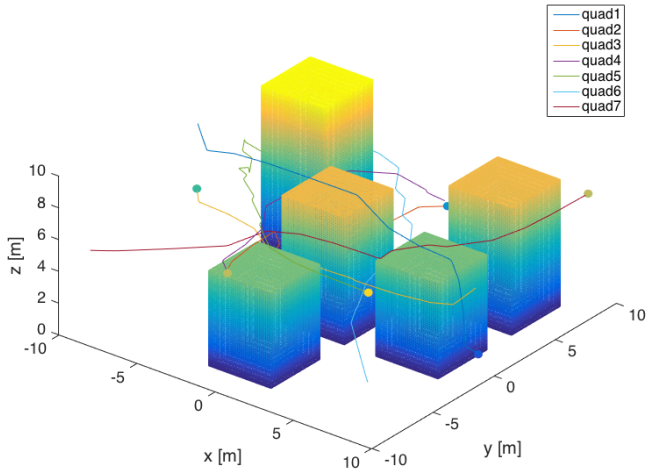
We now consider a 3D environment involving $N = 7$ agents and $M = 5$ cuboid obstacles of known geometries and positions. The base of all obstacles is a square with side of length $w = 4$ m, whereas their heights vary between 6 m and 10 m. The effective workspace of dimensions 20 m $\times$ 20 m $\times$ 10 m is divided into $10 \times 10 \times 5 = 500$ uniform grids of width $w = 2$ m. The parameter values of the simulation are the same as in the previous case. It is also assumed that all agents take off from the ground, fly to their departure points that lie on the same altitude, and wait there until all agents reach their departure points. Then the prioritized algorithm is initiated. The results are presented in Fig. 10.

More specifically, the resulting flight paths are shown in Fig. 10(a), where the spheres denote the destinations of the agents. The maximum loop time needed for the $A^\star$ to compute the path for each agent whenever triggered (for the selected grid) is less than 0.1 msec.

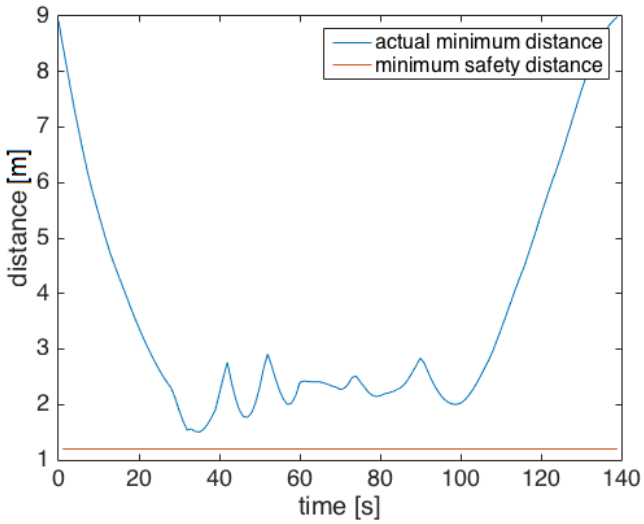
Also, the smallest pairwise inter-agent distance among any pair of agents at each time instant is shown in Fig. 10(b), where the horizontal line depicts the minimum allowable separation, to demonstrate that no collisions ever occur among any pair of agents.

VII. EXPERIMENTAL RESULTS

The efficacy of the proposed strategy is further demonstrated through experimental implementations using $N = 3$ quadro-

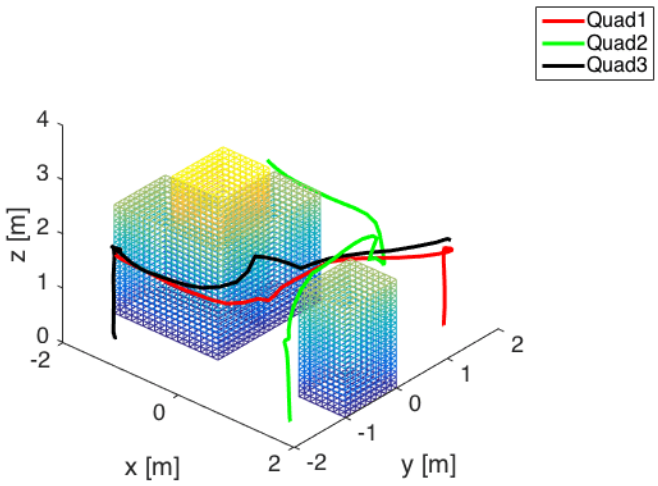


(a) 3D paths for 7 Agents (Simulation).

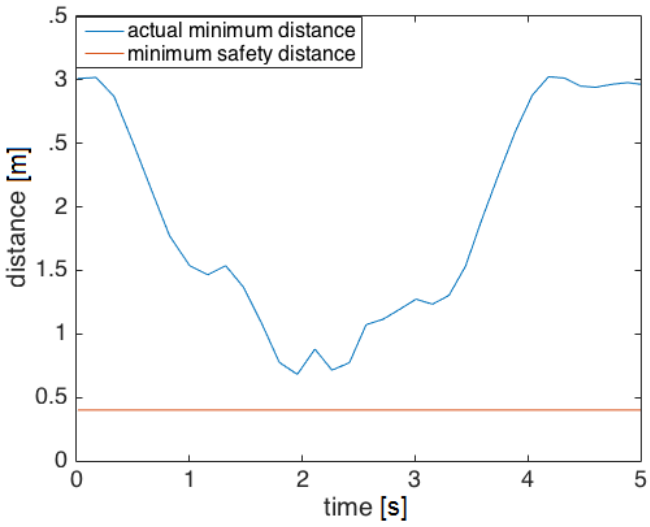


(b) Minimum distance among any pair of agents.

Fig. 10. 3D case - Simulation Results.



(a) 3D paths for 3 Agents (Experiment).



(b) Minimum distance among any pair of agents.

Fig. 11. 3D case - Experimental Results.

tors in an workspace with $M = 4$ obstacles. The experiments were conducted in the Distributed Aerospace Systems and Control Lab at the University of Michigan. The aerial platform used is the Hummingbird Quadrotor from Ascending Technologies, with the original Xbee modules replaced by Xbee WIFI modules for higher bandwidth. The ROS package used for the experiments is asctec_hl_framework, which provides velocity control and wireless communication through Xbee WIFI module between each Hummingbird and the ground PC. The ground PC is running with Intel(R) Core(TM) i7-4710HQ 2.5GHz processor in Linux OS, and the loop rate of control algorithm and communication is 20Hz. A VICON motion capture system provides measurements of high precision and accuracy for the positions of the quadrotors.

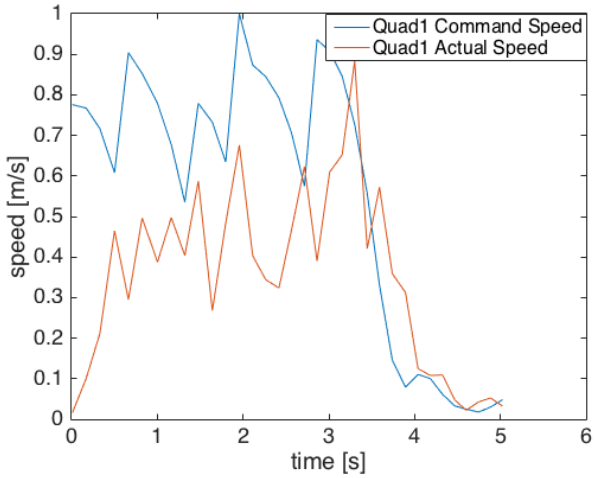
We considered a cubic workspace of dimensions 4 m $\times$ 4 m $\times$ 3 m, which was divided in 48 uniform cubid grid cells of side $w = 1$ m. The parameter values in the simulation are: $R_s = 0.7$ m, $R_z = 0.6$ m, $R_c = 0.5$ m, $d_s = 0.4$ m, $k_i = 3$. The $M = 4$ cuboid obstacles are of either 1 m or 2 m high, and are of the same 1 m $\times$ 1 m base.

The results are shown in Fig. 11, 12. More specifically, Fig. 11(a) shows the resulting paths of each agent. The paths are drawn from the time instant after the agents have reached their starting points, which have been set as: $\boldsymbol{r}_{10} = \begin{bmatrix}-1.5 & -1.5 & 1.5\end{bmatrix}^T$, $\boldsymbol{r}_{20} = \begin{bmatrix}-1.5 & 1.5 & 1.6\end{bmatrix}^T$, $\boldsymbol{r}_{30} = \begin{bmatrix}1.5 & 1.5 & 1.7\end{bmatrix}^T$, whereas their destinations are set as $\boldsymbol{r}_{1d} = \begin{bmatrix}1.5 & 1.5 & 1.5\end{bmatrix}^T$, $\boldsymbol{r}_{2d} = \begin{bmatrix}1.5 & -1.5 & 1.4\end{bmatrix}^T$, $\boldsymbol{r}_{3d} = \begin{bmatrix}-1.5 & -1.5 & 1.7\end{bmatrix}^T$.

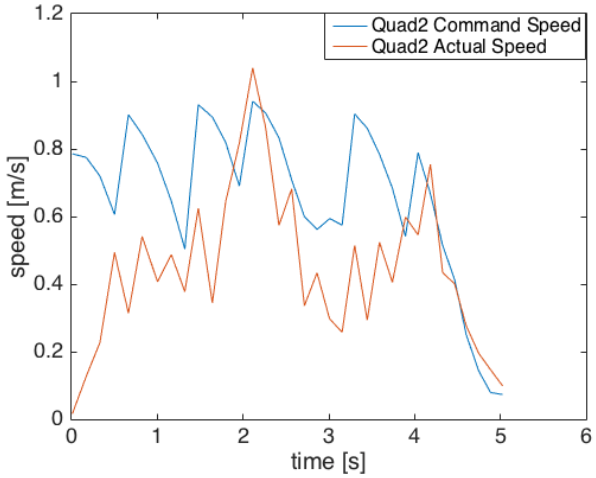
The minimum inter-agent distance among any pair of agents over time is depicted shown in Figure 11(b); since it remains always greater than the safe separation, the quadrotors are flying along collision-free trajectories.

The linear velocities commanded by the proposed algorithm, as well as the actual velocities, calculated as the derivatives of the position data acquired by the motion capture system during the experiment, are shown in Fig. 12(a), 12(b) and 12(c) for each one of the agents, respectively.

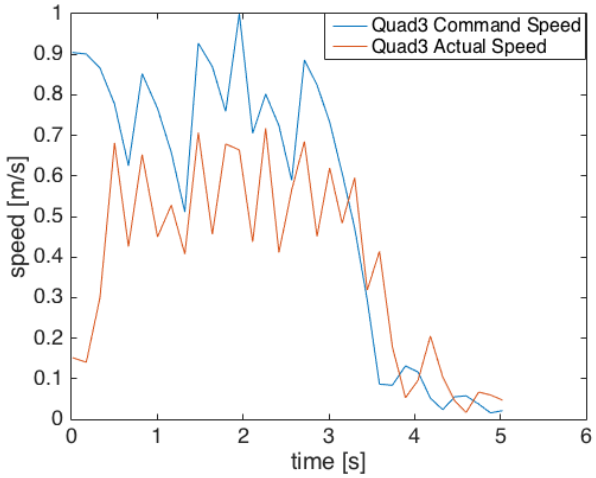
Finally, Fig. 13(a) and 13(b) show snapshots of the experimental procedure. The blue lines on the ground indicate the location of the obstacles.



(a) The commanded linear velocity for agent 1 and its actual linear velocity.

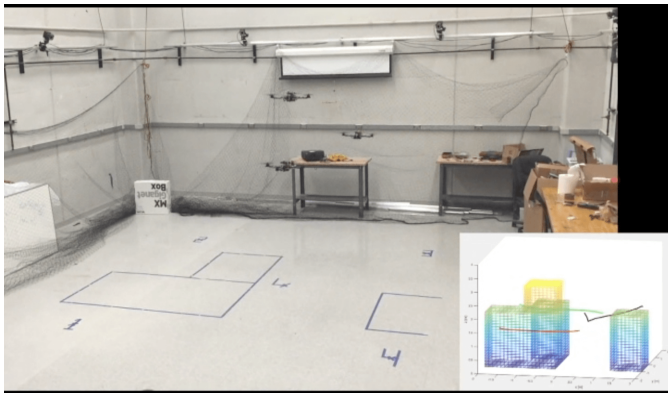


(b) The commanded linear velocity for agent 2 and its actual linear velocity.

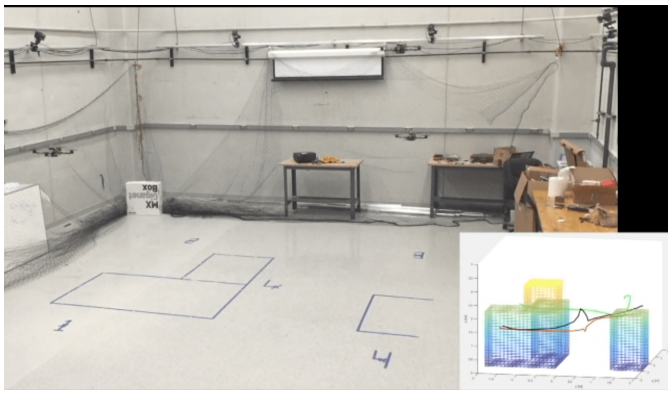


(c) The commanded linear velocity for agent 3 and its actual linear velocity.

Fig. 12. Commanded and actual control inputs during the experiment.



(a) Experimental Procedure with 3 Quadrotors (1).



(b) Experimental Procedure with 3 Quadrotors (2).

Fig. 13. Snapshots from the experimental trial.

VIII. Conclusions

This paper presented a planning, coordination and control strategy for the motion of multiple agents under constrained double-integrator dynamics in 3D polygonal environments. The method combines a prioritized $A^\star$ path planner along with a low-level coordination strategy based on Lyapunov-like barrier functions. Guarantees on the collision-free motion of the agents are provided, while the prioritized paths redefine the waypoints followed by the agents in an anticipatory manner that improves the performance of system in terms of path quality and travel time. Computer simulations as well as experimental results with quadrotors demonstrate the efficacy of the approach. Current and future work focuses on extending the algorithms to agents of more complicated dynamics, e.g., fixed-wing aircraft, and on incorporating uncertain environments and wind models that make relevant assumptions to future Air Traffic Control and Unmanned Aerial Traffic Management scenarios and applications.

Appendix: The $A^\star$ path planning algorithm.

The known map is divided into uniform square grids. Each grid has a corresponding node. When an agent is in one grid, we say that the agent is also in the corresponding node. Node is labeled as "impassable" when there are obstacles inside it, otherwise it is "passable". For a given node N_i, its position $\boldsymbol{p}_{N_i} = \begin{bmatrix} x_{N_i} & y_{N_i} \end{bmatrix}^T$ is the center of the corresponding grid. To describe the prioritized $A^\star$ algorithm we first need to define

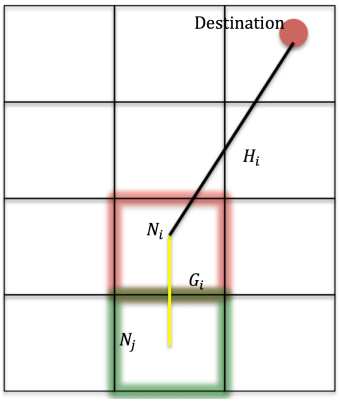


Fig. 14. The node parameters in the $A^\star$ algorithm.

three parameters: H_i, G_i and F_i associated with each node N_i as follows. Figure 14 gives a brief illustration of how the parameters are defined.

The parameter H_i is defined as:

$$H_i = \sqrt{(x_{N_i} - x_{id})^2 + (y_{N_i} - y_{id})^2}, \tag{28}$$

where $\boldsymbol{r}_{id} = \begin{bmatrix} x_{id} & y_{id} \end{bmatrix}^T$ is the destination of agent i. H_i represents the cost-to-go from N_i to the destination $\boldsymbol{r}_{id}$. Furthermore, the parameter:

$$G_i(N_j) = G_j + \sqrt{(x_{N_i} - x_{N_j})^2 + (y_{N_i} - y_{N_j})^2}, \tag{29}$$

where N_j is some other node located at $\boldsymbol{p}_{N_j} = \begin{bmatrix} x_{N_j} & y_{N_j} \end{bmatrix}^T$, represents the potential cost for the agent going from its current node to N_i passing through the node N_j. Finally, the parameter:

$$F_i = H_i + G_i, \tag{30}$$

represents the total potential cost for the agent going from its current node to its destination while passing through N_i.

A node would also have a parent node if it is assigned one, for node N_i, its parent is denoted by $N_{i,parent}$. The final path is determined by first adding the target node to the path list and then recursively tracing the parent node of the last added node, until the start node has been added.

REFERENCES

[1] L. E. Parker, "Path planning and motion coordination in multiple mobile robot teams," in *Encyclopedia of Complexity and System Science*, R. A. Meyers, Ed. Springer, 2009.

[2] W. Ren and Y. Cao, "Overview of recent research in distributed multi-agent coordination," in *Distributed Coordination of Multi-agent Networks*, ser. Communications and Control Engineering. Springer-Verlag, 2011, ch. 2, pp. 23–41.

[3] L. Kavraki, P. Svestka, J.C.-Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–579, Aug. 1996.

[4] S. M. LaValle, J. J. Kuffner, and Jr., "Rapidly-exploring random trees: Progress and prospects," 2000.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal on Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[6] R. Tedrake, I. R. Manchester, M. M. Tobenkin, and J. W. Roberts, "LQR-Trees: Feedback motion planning via sums-of-squares verification," *Int. Journal on Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[7] A. Majumdar, R. Vasudevan, M. M. Tobenkin, and R. Tedrake, "Convex optimization of nonlinear feedback controllers via occupation measures," *Int. Journal on Robotics Research*, vol. 33, no. 9, pp. 1209–1230, 2014.

[8] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[9] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion. Theory, Algorithms and Implementation*. MIT Press, 2005.

[10] E. G. Hernandez-Martinez and E. Aranda-Bricaire, "Convergence and collision avoidance in formation control: A survey of the artificial potential functions approach," in *Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications*, F. Alkhateeb, E. A. Maghayreh, and I. A. Doush, Eds. InTech, 2011, ch. 6, pp. 103–126.

[11] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, Oct. 1992.

[12] D. V. Dimarogonas and K. J. Kyriakopoulos, "Connectedness preserving distributed swarm aggregation for multiple kinematic robots," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1213–1223, Oct. 2008.

[13] N. Ayanian and V. Kumar, "Decentralized feedback controllers for multiagent teams in environments with obstacles," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 878–887, Oct. 2010.

[14] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. of the 2008 IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, USA, May 2008, pp. 1928–1935.

[15] R. Bohlin and L. Kavraki, "Path planning using lazy PRM," in *Proc. of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, 2000, pp. 521 – 528.

[16] K.-H. C. Wang and A. Botea, "MAPP: a scalable multi-agent path planning algorithm with tractability and completeness guarantees," *Journal of Artificial Intelligence Research*, vol. 42, pp. 55–90, 2011.

[17] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, Aug. 2007.

[18] C. Sloth, G. J. Pappas, and R. Wisniewski, "Compositional safety analysis using barrier certificates," in *Proc. of the 15th ACM international conference on Hybrid Systems: Computation and Control (HSCC)*, Beijing, China, Apr. 2012, pp. 15–24.

[19] A. J. Barry, A. Majumdar, and R. Tedrake, "Safety verification of reactive controllers for uav flight in cluttered environments using barrier certificates," in *Proc. of the 2012 IEEE Int. Conf. on Robotics and Automation*, Saint Paul, Minnesota, USA, May 2012, pp. 484–490.

[20] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," https://arxiv.org/abs/1601.04037, 2016.

[21] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 271–276.

[22] P. Velagapudi, K. Sycara, and P. Scerri, "Decentralized prioritized planning in large multirobot teams," in *Proc. of the 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 2010, pp. 4603–4609.

[23] T. Zheng, D. Liu, and P. Wang, "Priority based dynamic multiple robot path planning," in *Proc. of the 2nd Int. Conf. on Autonomous Robots and Agents*, Palmerston North, New Zealand, Dec. 2004, pp. 373–378.

[24] J. van den Berg and M. Overmars, "Prioritized motion planning for multiple robots," in *Proc. of the 2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Aug. 2005, pp. 430–435.

[25] M. Cap, P. Novak, M. Selecky, J. Faigl, and J. Vokrinek, "Asynchronous decentralized prioritized planning for coordination in multi-robot system," in *Proc. of the 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 2013, pp. 3822–3829.

[26] W. Yu, J. Peng, X. Zhang, and K. chi Lin, "A cooperative path planning algorithm for a multiple mobile robot system in a dynamic environment," *International Journal of Advanced Robotic Systems*, vol. 136, no. 11, pp. 1–12, Nov. 2014.

[27] S. Liu, D. Sun, and C. Zhu, "A dynamic priority based path planning for cooperation of multiple mobile robots in formation forming," *Robotics and Computer-Integrated Manufacturing*, vol. 30, pp. 589–596, Nov. 2014.

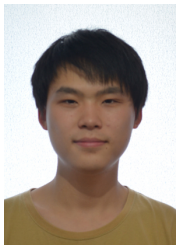
[28] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, "Implan: Scalable incremental motion planning for multi-robot systems," in *Proceedings of the 7th International Conference on Cyber-Physical Systems (ICCPS)*, Apr. 2016, pp. 22–31.

[29] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509–521, Apr. 1998.

[30] K. Zeghal, "A review of different approaches based on force fields for airborne conflict resolution," in *AIAA Guidance, Navigation, and Control Conference*, Boston, Aug. 1998, pp. AIAA Paper 1998–4240.
[31] E. Frazzoli, Z.-H. Mao, J.-H. Oh, and E. Feron, "Resolution of conflicts involving many aircraft via semidefinite programming," *Journal of Guidance, Control and Dynamics*, vol. 24, no. 1, pp. 179–189, Jan. 2001.
[32] M. R. Jardin, "Analytical relationships between conflict counts and air-traffic density," *Journal of Guidance, Control and Dynamics*, vol. 28, no. 6, pp. 1150–1156, Jan. 2005.
[33] G. Dowek, C. Munoz, and V. A. Carreno, "Provably safe coordinated strategy for distributed conflict resolution," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, Aug. 2005, pp. AIAA–2005–6047.
[34] M. A. Christodoulou and S. G. Kodaxakis, "Automatic commercial aircraft-collision avoidance in free flight: The three-dimensional problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 2, pp. 242–249, June 2006.
[35] A. L. Galdino, C. M. noz, and M. Ayala-Rincón, "Formal verification of an optimal air traffic conflict resolution and recovery algorithm," in *Logic, Language, Information and Computation*, ser. Lecture Notes in Computer Science. Berlin Heidelberg: Springer-Verlag, 2007, vol. 4576, pp. 177–188.
[36] M. Gariel and E. Feron, "Graceful degradation of air traffic operations: Airspace sensitivity to degraded surveillance systems," *Proceedings of the IEEE*, vol. 96, no. 12, pp. 2028–2039, Dec. 2008.
[37] A. Platzer, "Air traffic collision avoidance," in *Logical Analysis of Hybrid Systems*. Berlin Heidelberg: Springer-Verlag, 2010, pp. 303–334.
[38] A. Alonso-Ayuso, L. F. Escudero, and F. J. Martín-Campo, "Collision avoidance in air traffic management: A mixed-integer linear optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 47–57, Mar. 2011.
[39] J. Le Ny and G. J. Pappas, "Joint metering and conflict resolution in air traffic control," *Journal of Guidance, Control and Dynamics*, vol. 34, no. 5, pp. 475–482, Sept. 2011.
[40] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-generation airborne collision avoidance system," *Lincoln Laboratory Journal*, vol. 19, no. 1, pp. 17–33, 2012.
[41] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, and A. Platzer, "A formally verified hybrid system for the next-generation airborne collision avoidance system," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2015, pp. 21–36.
[42] D. Panagou, D. M. Stipanović, and P. G. Voulgaris, "Distributed coordination and control of multi-robot networks using Lyapunov-like barrier functions," *IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 617–632, Mar. 2016.
[43] D. Panagou, "A distributed feedback motion planning protocol for multiple unicycle agents of different classes," *IEEE Transactions on Automatic Control*, pp. accepted as Full Paper, to appear., 2016.
[44] X. Ma, Z. Jiao, Z. Wang, and D. Panagou, "Decentralized prioritized motion planning for multiple autonomous UAVs in 3d polygonal obstacle environments," in *Proc. of the 2016 Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, Arlington, VA, USA, June 2016, pp. 292–300.
[45] A. G. Wills and W. P. Heath, "A recentred barrier for constrained receding horizon control," in *Proc. of the American Control Conference*, Anchorage, AK, USA, May 2002, pp. 4177–4182.

Xiaobai Ma was born in Weinan, Shaanxi, China, in 1993. He received the B.S.E degree in Aerospace Engineering from University of Michigan, Michigan, United States, in 2016, and the B.E degree in Electrical and Computer Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2016. He is currently a first year Master student in the department of Aeronautics and Astronautics at Stanford University. His current research interests include deep reinforcement learning and autonomous driving.

Ziyuan Jiao was born in Beijing, China, in 1993. He received the B.S.E degree in Aerospace Engineering from University of Michigan, Michigan, United States, in 2016, and the B.E degree in Mechanical Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2016. He is currently a Master student in the department of Mechanical Engineering, University of Michigan. His current research interests include dynamics and control in robotics systems, mechatronic system design and control in manufacturing.



Zhenkai Wang is a graduate student of Aeronautics and Astronautics at Stanford University. He earned his B.S.E. in Aerospace Engineering from University of Michigan, Ann Arbor, in 2016, and then B.S.E. in Electrical and Computer Engineering at Shanghai Jiao Tong University, in the same year. His research interests include motion planning, computer vision and robotic perception.

Dimitra Panagou received the Diploma and PhD degrees in Mechanical Engineering from the National Technical University of Athens, Greece, in 2006 and 2012, respectively. Since September 2014 she has been an Assistant Professor with the Department of Aerospace Engineering, University of Michigan. Prior to joining the University of Michigan, Dr. Panagou was a postdoctoral research associate with the Coordinated Science Laboratory, University of Illinois, Urbana-Champaign (2012-2014), a visiting research scholar with the GRASP Lab, University of Pennsylvania (June 2013, fall 2010) and a visiting research scholar with the University of Delaware, Mechanical Engineering Department (spring 2009). Her research interests include the fields of planning, coordination and distributed control of complex systems, with applications in unmanned aerial systems, robotic networks and autonomous multi-vehicle systems (ground, marine, aerial, space). She is a recipient of a NASA Early Faculty Career Award, of an AFOSR Young Investigator Award, and member of the IEEE and the AIAA.