Decentralized Prioritized Motion Planning for Multiple Autonomous UAVs in 3D Polygonal Obstacle Environments

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

***Abstract*— This paper presents a decentralized multi-agent motion planning method for aerial robots moving among 3D polygonal obstacles. The algorithm combines a prioritized A^* algorithm for high-level (global) planning along with a barrier functions-based method for low-level (local) coordination and control. We first extend the barriers function method developed in earlier work to treat arbitrary polygonal obstacles. We then combine the prioritized A^* algorithm to compute waypoints and paths that facilitate the performance of the barrier-based coordination and collision avoidance. 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 and experimental results with quadrotors in 2D and 3D environments demonstrate the efficacy of the proposed approach.**

I. Introduction

Path and motion planning is a core, active research topic in the robotics and control systems communities that becomes particularly challenging for multi-robot systems [1]. Motion planning for a single robot operating in an obstacle environment is typically treated with sampling-based methods (probabilistic roadmaps, rapidly-exploring random trees, LQR-trees), Lyapunov-based methods (artificial potential functions and vector field methods), graph search and decision theory, see for instance [2]–[8]. Each of these methods 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 [9] 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 [10]. Re-active methods have been refined over the years to overcome the deficiencies of gradient-based solutions [11]. In [12], 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. Traditional discrete planning methods such as rapidly-exploring random trees and probabilistic roadmaps do not scale well for multi-agent problems as they require much computation when directly used in high-dimensional spaces. The Lazy Probabilistic Roadmap Method [13], and MAPP [14] aim to reduce the computational demand of such methods. However, collision avoidance in these approaches is treated in a discrete manner, i.e., agents can only switch between nodes but not fully use the space between nodes. In contrast to this solution, potential function methods can provide a continuous avoiding behavior.

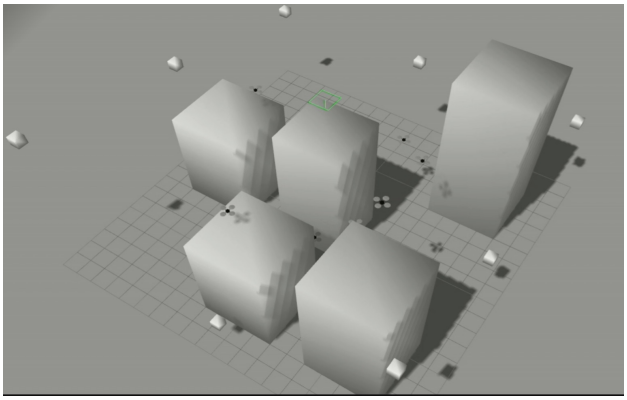

Fig. 1. The considered scenario: Quadrotors flying in an urban area towards goal destinations while avoiding obstacles (other quadrotors, buildings).

Prioritized planning has also been studied in the robotics motion planning literature, and various approaches on assigning priorities are described, for instance, in [15]–[18]. In a prioritized approach, each robot is assigned with a priority, and each robot computes its path based on the order implied by their priority. The resulting path for each agent hence avoids static obstacles on the map and the robots with higher priority in the communication region, which are considered as dynamic obstacles. In this paper, we take advantage of the priority approach to improve the performance of the A^* algorithm when applied to a multi-UAV system.

More specifically, in this paper we propose a two-level strategy for the decentralized multi-agent motion in 3D polygonal obstacle environments, see also Figure 1, that combines prioritized A^* high-level planning with barrier functions for low-level coordination and control. We first build upon the method in [19], and modify it to account for known polygonal obstacles in 3D environments. We then implement a prioritized A^* algorithm for planning paths and computing waypoints for the agents that improve the performance of the low-level barrier-based control. Safe coordination is taken care of by the low-level controller. This way, the proposed two-level strategy combines the

*Corresponding author

The authors are with the Department of Aerospace Engineering, University of Michigan, Ann Arbor. {maxiaoba, zyjiao, zackwang, dpanagou}@umich.edu. This work was in part done during the U. of Michigan SURE (Summer Undergraduate REsearch) project of the first and second author.

advantages of the utilized methods while overcoming their disadvantages. Based on barrier functions we design an velocity coordination algorithm that guarantees the collision-free motion of the agents while reducing the computation load of pure discrete path planning in high-dimensional spaces. By using the prioritized A^* algorithm for path planning, we obtain an optimal path and waypoints that avoids the stucking situation often seen in pure potential function methods. Experimental results with three quadrotors accompany the computer simulations and demonstrate the efficiency of the proposed approach in realistic settings.

The paper is organized as follows: Section II gives the mathematical modeling while Section III provides an overview of the proposed approach. The high-level and the low-level layers of the algorithm are given in Sections V and IV, respectively, while Section VII presents their extension to 3D environments. Simulation and experimental results are given in Sections VIII and IX while Section X summarizes our findings and thoughts on future research.

II. Modeling and Problem Statement

A. Quadrotor Dynamics Model

Modeling of the dynamics of a quadrotor is a topic well stated in the literature, see for instance [20]. Here we simply give a short summary. For a quadrotor with mass m and inertia $\boldsymbol{J}$ expressed in the body-fixed frame, the equations of motion can be written as:

$$\dot{\boldsymbol{r}}^n = \boldsymbol{v}^n, \tag{1a}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \circ \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b \end{bmatrix}, \tag{1b}$$

$$\dot{\boldsymbol{v}}^n = m^{-1}\boldsymbol{C}_b^n \boldsymbol{F}, \tag{1c}$$

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

where $\boldsymbol{r}^n$ and $\boldsymbol{v}^n$ are position and velocity of the center of gravity of the quadrotor with respect to (w.r.t.) the navigational (i.e., a inertial) frame of reference, $\boldsymbol{q}$ is the attitude of the quadrotor in quaternion representation, $\boldsymbol{C}_b^n$ is the rotation matrix from the body-fixed frame to the navigational frame, $\dot{\boldsymbol{w}}^b$ is the angular velocity in the body-fixed frame, and $\boldsymbol{F}$ is the total force including the motor trust, gravity and drag forces. Multiplication $\circ$ in Equation (1b) denotes the Hamilton Product. The total torque $\boldsymbol{M}$ contains the moments generated by thrust and drag forces. 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 linear velocity commands to the quadrotor.

B. Problem Statement

We initially consider N quadrotors which are deployed in a known 2D workspace (environment) $\mathcal{W}$ with M static polygonal obstacles. Each quadrotor $i \in \{1, \ldots, N\}$ is modeled as a circular disk of radius r_a. Each quadrotor has access to its own position and velocity, measures the position of quadrotors lying in its sensing region, realized as a circular disk of radius R_s, and exchanges information on velocity with agents lying in its safety (or communication) region, which is realized as a circular disk of radius $R_c < R_s$. Each quadrotor is assigned a goal location $\boldsymbol{r}_{id} = \begin{bmatrix} x_{id} & y_{id} \end{bmatrix}^T$ in the free space and needs to move there while avoiding static obstacles and other agents. In addition, each quadrotor has its own priority which could be either predefined or calculated dynamically.

III. Overview of the Two-Level Approach

Our method contains two layers: the low-level (or local) coordination and control, and the high-level (or global) path planning.

The low-level layer is built upon the coordination protocols in [19], and extended to multi-agent coordination in 3D polygonal environments. The design is decentralized since the effect of other agents and obstacles is considered only locally, i.e., when other agents and obstacles are close enough. The high-level path planning is done by dividing the map of the environment into uniform grids and performing an prioritized A^* path finding algorithm for each agent. This layer considers the destination and the obstacles for each agent for finding the nominal path as well as the current and expected next grid occupied by nearby agents with higher priority. Once the waypoint list is generated, each agent follows the waypoints under the low-level controller until the high-level re-planning is triggered.

IV. High-Level Path Planning

We use a prioritized A^* algorithm to compute the paths of the agents to their destinations. 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^* algorithm we first need to define three parameters: H_i, G_i and F_i associated with each node N_i as follows. Figure 2 gives a brief illustration of how the parameters are defined.

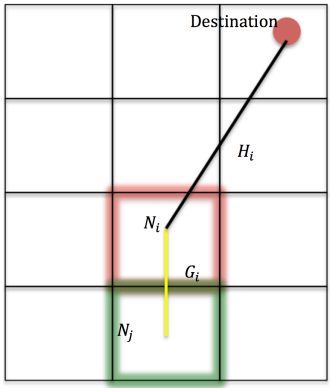

Fig. 2. A^* Algorithm node parameters

The parameter H_i is defined as:

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

where $\boldsymbol{r}_{id} = \begin{bmatrix} x_{id} & y_{id} \end{bmatrix}^T$ is the agent i's destination. H_i represents the cost to go from N_i to the destination $\boldsymbol{r}_{id}$ for agent i. Furthermore, the parameter:

$$G_i(N_j) = G_j + \sqrt{(x_{N_i} - x_{N_j})^2 + (y_{N_i} - y_{N_j})^2}, \quad (3)$$

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:

$$F_i = H_i + G_i, \quad (4)$$

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 $N_{i,parent}$.

Each agent i communicates with other agents inside its sensing area with their priorities, current nodes, and next n nodes in their waypoint lists where $n = \frac{R_c}{gridsidelength}$. The current and the next node of an agent are regarded as occupied by that agent. By avoiding the occupied nodes of nearby quadrotors with higher priorities, traffic jam is effectively detected and avoided.

A pseudocode of the prioritized A^* algorithm is given below in Algorithm 1.

A path re-planning constant $re > 0$ is defined such that the high-level path planning would be triggered once every re times. Increasing re would decrease the overall path re-planning times and therefore decrease the computation cost. However, a larger re also means that it takes larger time for the agent to realize the jam situation around it. Thus, re should be made as high as possible under the given computational limits. During the time intervals when the high-level path planning is not triggered, the mutual avoidance is controlled by the barrier function.

Algorithm 1 Prioritized A^* Algorithm

procedure A^* 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

V. Low Level Control Design

A. Barrier Functions for Multi-Agent Coordination

We briefly discuss the construction of the barrier function per agent i that encodes convergence to its destination $\boldsymbol{r}_{id}$ and avoidance of neighbor agents $j \neq i$. More details are available in [19].

1) Convergence: We encode convergence of agent i to its destination $\boldsymbol{r}_{id}$ through a cost function:

$$V_{i0} = \|\boldsymbol{r}_i - \boldsymbol{r}_{id}\| = \sqrt{(x_i - x_{id})^2 + (y_i - y_{id})^2}. \quad (5)$$

2) Avoiding neighbors: For encoding that agent i keeps a safe separation d_s w.r.t. agent j we define the constraint function: $c_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = (x_i - x_{id})^2 + (y_i - y_{id})^2 - {d_s}^2 > 0$, the barrier function:

$$b_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = -ln(c_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j)), \quad (6)$$

and the recentered barrier function:

$$r_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = b_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) - b_{ij}(\boldsymbol{r}_{id}, \boldsymbol{r}_j) - (\nabla b_{ij}|_{\boldsymbol{r}_{id}})^T(\boldsymbol{r}_i - \boldsymbol{r}_{id}), \quad (7)$$

where $(\nabla b_{ij}|_{\boldsymbol{r}_{id}})^T$ is the transpose of the gradient vector of the barrier function $b_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j)$ (6), evaluated at $\boldsymbol{r}_{id}$. To obtain a positive-definite function we take:

$$V'_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = r_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j)^2, \quad (8)$$

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

$$V_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) = \sigma_{ij} V'_{ij}, \quad (9)$$

where:

$$\sigma_{ij} = \begin{cases} 1, & d_s \leq d_{ij} \leq R_z \\ Ad_{ij}^3 + Bd_{ij}^2 + Cd_{ij} + D, & R_z < d_{ij} < R_s \\ 0, & d_{ij} \geq R_s \end{cases} \quad (10)$$

with the coefficients being computed as $A = -\frac{2}{(R_z - R_s)^3}$, $B = \frac{3(R_z+R_s)}{(R_z=R_s)^3}$, $C = -\frac{6R_zR_s}{(R_z=R_s)^3}$, $D = \frac{R_s^2(3R_z-R_s)}{(R_z-R_s)^3}$, so that (10) is a continuously differentiable function.

B. Extension to Polygonal Environments

We made the necessary amendments to [19] to address 2D (and in the sequel, 3D) polygonal environments.

We first notice that when $\|\boldsymbol{r}_{id} - \boldsymbol{r}_j\|^2 < {d_s}^2$, then the terms $b_{ij}(\boldsymbol{r}_{id}, \boldsymbol{r}_j)$ and $(\nabla b_{ij}|_{\boldsymbol{r}_{id}})^T$ in (7) are undefined. To overcome this, we introduce a new vector called $\boldsymbol{v}$ which indicates the preferred direction of agent i: $\boldsymbol{v}$ is a unit vector whose direction is the relative direction from the agent's current node to the next node in the agent's waypoint list. When agents and goal locations are close enough so that (7) becomes undefined, then the current waypoint $\boldsymbol{r}_{id} = \begin{bmatrix} x_{id} & y_{id} \end{bmatrix}^T$ is replaced by the waypoint $\boldsymbol{r}_{id}^n = \boldsymbol{p}_{N_i} + d\,\boldsymbol{v}$, where $\boldsymbol{p}_{N_i}$ is the agent's current node position and $d = 2 \times R_s$. In this way, we avoid the undefined situation while we make the agent move in its preferred direction.

We next treat the case of polygonal obstacles. We assume that each obstacle is an arbitrary convex polygon, randomly positioned in the workspace, and that its geometry is known through a map of the environment. For a polygonal obstacle with n edges, we extend the edges of obstacles and divide the nearby area into $2n$ regions. We define the distance d between the agent and the obstacle according to different regions in which the agent is, see Figure 3 :

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

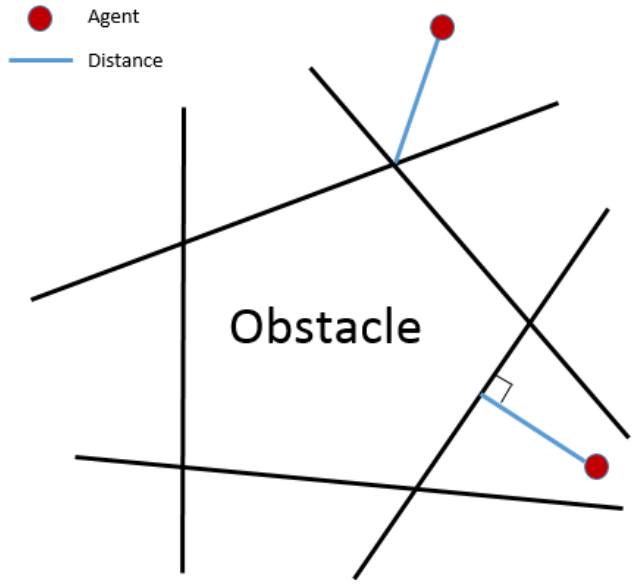

Fig. 3. Area division for 2D obstacle

An example on defining the distance between and agent and a rectangular obstacle o_j with 4 vertices whose coordinates are (x_{oj1}, y_{oj1}), (x_{oj1}, y_{oj2}), (x_{oj2}, y_{oj1}) and (x_{oj2}, y_{oj2}), where $x_{oj1} < x_{oj2}$ and $y_{oj1} < y_{oj2}$, is given as follows. Let the position of the agent be (x, y), then the distance of the agent to the obstacle is defined as:

$$d_{ioj} = \begin{cases} x_{oj1} - x, & \text{if } (x < x_{oj1}) \wedge (y_{oj1} < y < y_{oj2}), \\ x - x_{oj2}, & \text{if } (x > x_{oj2}) \wedge (y_{oj1} < y < y_{oj2}), \\ y_{oj1} - y, & \text{if } (y < y_{oj1}) \wedge (x_{oj1} < x < x_{oj2}), \\ y - y_{oj2}, & \text{if } (y > y_{oj2}) \wedge (x_{oj1} < x < x_{oj2}), \\ \sqrt{(x - x_{oj1})^2 + (y - y_{oj1})^2}, & \\ \quad \text{if } (y \le y_{oj1}) \wedge (x \le x_{oj1}), & \\ \sqrt{(x - x_{oj1})^2 + (y - y_{oj2})^2}, & \\ \quad \text{if } (y \ge y_{oj2}) \wedge (x \le x_{oj1}), & \\ \sqrt{(x - x_{oj2})^2 + (y - y_{oj1})^2}, & \\ \quad \text{if } (y \le y_{oj1}) \wedge (x \ge x_{oj2}), & \\ \sqrt{(x - x_{oj2})^2 + (y - y_{oj2})^2}, & \\ \quad \text{if } (y \ge y_{oj2}) \wedge (x \ge x_{oj2}). & \end{cases}$$

We then define the barrier function for the agent-obstacle case in a similar spirit to the agent-agent case:

$$V_{ioj} = (b_{ioj}(x, y))^2 = (-ln(d(x, y)^2))^2. \tag{11}$$

Finally, we constrain the effect of the obstacles in a zone around them that is determined by a parameter called $fence$; more specifically, when $d_{ioj} > fence$, then b_{ioj} and V_{ioj} are set equal to zero.

C. Design of the Low-Level Controller

For an agent i moving in an N-agent environment with M static obstacles we define the function:

$$v_i = \sum_{j=0}^{N} V_{ij} + \sum_{k=1}^{M} V_{iok}, \tag{12}$$

and scale it to vary between 0 and 1 as:

$$V_i = \frac{v_i}{1 + v_i}. \tag{13}$$

The direction of motion of agent i is then controlled to be along the negated gradient vector: $\phi_i = \text{atan2}(-\frac{\partial V_i}{\partial x_i}, -\frac{\partial V_i}{\partial y_i})$.

To obtain safe (collision-free) position trajectories for agent i we modify the controller in [19] and set the magnitude of the linear velocity of agent i as:

$$u_i = \begin{cases} \min_{j \in I | J_j < 0} u_{i|j} & d_s \le d_{ij} \le R_c \\ u_{ic} & d_{ij} > R_c. \end{cases} \tag{14}$$

The velocity $u_{i|j}$ denotes a safe (collision-free) velocity of agent i w.r.t. agent j, given as:

$$u_{i|j} = u_{ic}\frac{d_{ij} - d_s}{R_c - d_s} + u_{is|j}\frac{R_c - d_{ij}}{R_c - d_s}, \tag{15}$$

where the terms in (15) are defined as:

$$u_{ic} = k_i \tanh(\|\boldsymbol{r}_i - \boldsymbol{r}_{id}\|), \tag{16a}$$

$$u_{is|j} = u_j \frac{\boldsymbol{r}_{ji} \cdot \boldsymbol{v}_j}{\boldsymbol{r}_{ji} \cdot \boldsymbol{v}_i} \tag{16b}$$

where I is the set of agents in the safety region of i, $J_j = \boldsymbol{r}_{ji} \cdot \boldsymbol{v}_i$, $\boldsymbol{r}_{ji} = \boldsymbol{r}_i - \boldsymbol{r}_j$ and $\boldsymbol{v}_i$ is the normalized velocity vector of agent i. The set $J_j < 0$ denotes the neighbor agents of agent i whom the agent i is approaching on its current direction. In this way agent i slows down w.r.t. the neighbor agents with whom it is more susceptible to collide.

Algorithm 2 Main Algorithm

procedure MAIN ALGORITHM
for each agent:
    generate node map
    loop number $\leftarrow 0$
loop:
    global path planning:
    if loop number = re (path re-planing constant) **then**
        loop number $\leftarrow 0$
        re-plan path with Prioritized A^* Algorithm
    else
        loop number $\leftarrow$ loop number+1
    local path planning:
    if obstacle approaching **then**
        calculate $\boldsymbol{v}$, change $\boldsymbol{r}_{id}$ to $\boldsymbol{p}_{N_i} + d\boldsymbol{v}$ where $\boldsymbol{p}_{N_i}$ is the agent's current node position, $d = 2 \times Rs$.
    else
        set $\boldsymbol{r}_{id}$ to the second node
    turn on barrier functions and calculate velocity vector
end loop

VI. Main Algorithm

A pseudocode of our main algorithm is given as Algorithm 2 (above).

VII. Extension to 3D Environments

In 3D environment, the quadrotors are allowed to change altitude instead of keeping altitude-hold mode in 2D case.

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 are rewritten as: $H_i = \sqrt{(x_{N_i} - x_{id})^2 + (y_{N_i} - y_{id})^2 + (z_{N_i} - z_{id})^2}$, $G_i(N_j) = G_j + \sqrt{(x_{N_i} - x_{N_j})^2 + (y_{N_i} - y_{N_j})^2 + (z_{N_i} - z_{N_j})^2}$.

In low level control, we extend the obstacle surfaces to divide the surrounding space into several regions, as shown in Figure 4. The distance d between the agent and the obstacle is now defined based on the region in which the agent is, see also Figure 4, as follows:

- In the regions that include an edge of the obstacle (type A), d 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 is defined as the distance between the surface and the agent.
- In the regions that include a vertex of the obstacle (type C), d is defined as the distance between the vertex and the agent.

Finally, the direction of agent i is now considered to be along the vector $(-\frac{\partial V_i}{\partial x_i}, -\frac{\partial V_i}{\partial y_i}, -\frac{\partial V_i}{\partial z_i})$.

VIII. Simulation Results

The efficacy of the proposed approach is demonstrated through simulation results in 2D and 3D environments.

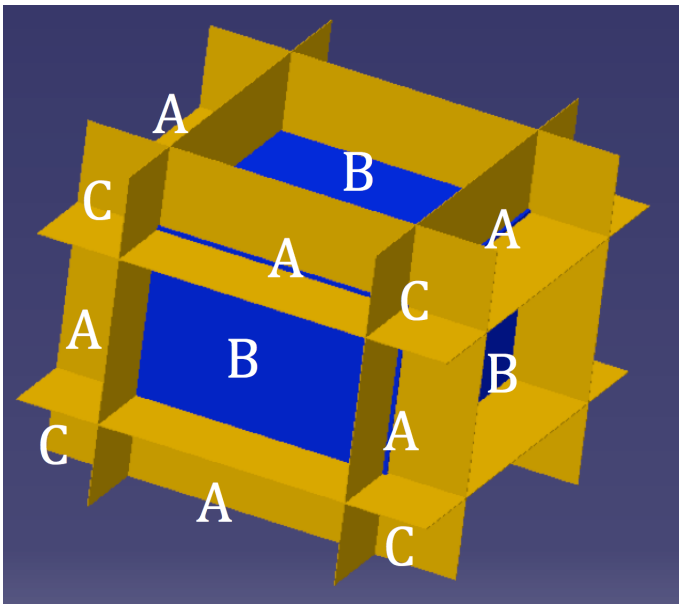

Fig. 4. Space division into regions of type A, B, C around a 3D obstacle.

A. 2D case

Computer simulations were performed by Gazebo with ROS. We use the hector_quadrotor package in ROS to setup a simulation environment with quadrotors. This package has build-in noise, filter, sensor, and velocity control which is suitable to demonstrate our algorithm. We ran two simulations to show 2D and 3D cases with a control loop frequency of 20Hz.

The first simulation demonstrates the motion of 20 agents in a 2D known polygonal environment. Each agent computes a path to its goal using $A^\star$ and moves towards it under the barrier control law (14). Once it gets connected with other agents, the imminent conflicts are modeled in the barrier construction based on the assigned priorities.

The assigned priorities are only used in the high-level path planning which are triggered once re loops for each agent. While the high-level path planning is not triggered, the mutual avoidance is controlled by the barrier function.

In the sequel we consider that all units are in SI. The map is 40x40 and is divided into 20x20=400 uniform grids of width $w_g = 2$. The parameter values in the simulation are: $R_s = 1.5$, $R_z = 1.4$, $R_c = 1.3$, $d_s = 1.2$, $k_1 = 1$, $fence = 0.3$.

Table I gives the departure and destination for the 20 agents. Figure 5 depicts the resulting paths of the agents under the proposed control strategy. The initial positions of the agents are depicted with red circles. The linear velocities of the agents are shown in Figure 6. The minimum pairwise distance over time is illustrated in Figure 7, where the horizontal line stands for the minimum allowable separation d_s. The time needed for A^* algorithm to compute a path for each agent (for the selected grid) is less than 1.5 msec.

For comparison, we also ran the simulation with A^* algorithm without priorities. The simulation setup and the

TABLE I

DEPARTURE AND DESTINATION FOR 2D SIMULATION

agent	departure	destination	agent	departure	destination
1	(-17,-19)	(17,19)	2	(-7,-19)	(7,19)
3	(0,-19)	(0,19)	4	(7,-19)	(-7,19)
5	(17,-19)	(-17,19)	6	(19,-17)	(-19,17)
7	(19,-7)	(-19,7)	8	(19,0)	(-19,0)
9	(19,7)	(-19,-7)	10	(19,17)	(-19,-17)
11	(17,19)	(-17,-19)	12	(7,19)	(-7,-19)
13	(0,19)	(0,-19)	14	(-7,19)	(7,-19)
15	(-17,19)	(17,-19)	16	(-19,17)	(19,-17)
17	(-19,7)	(19,-7)	18	(-19,0)	(19,0)
19	(-19,-7)	(19,7)	20	(-19,-17)	(19,17)

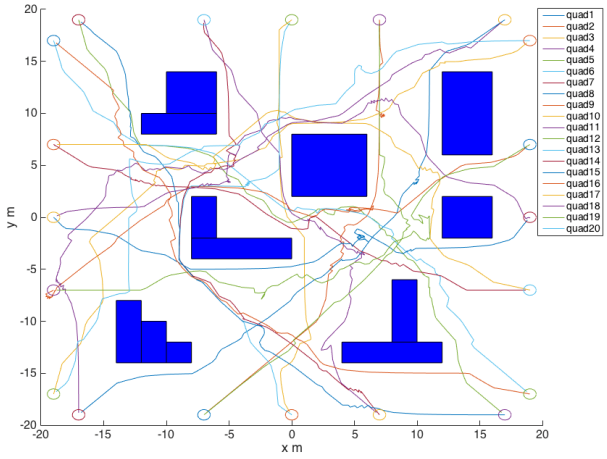

Fig. 5. Paths of 20 Agents in 2D Case

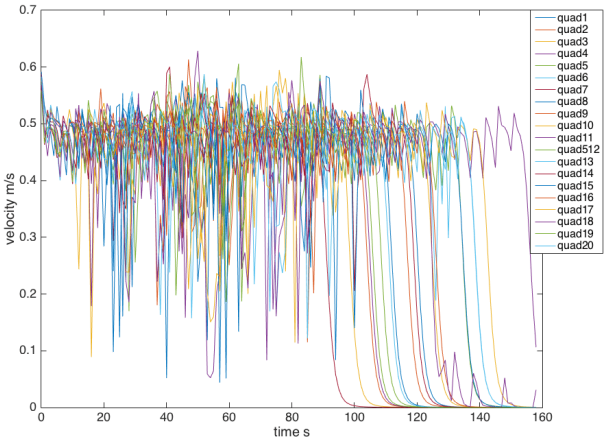

Fig. 6. Velocities of 20 Agents in 2D Case

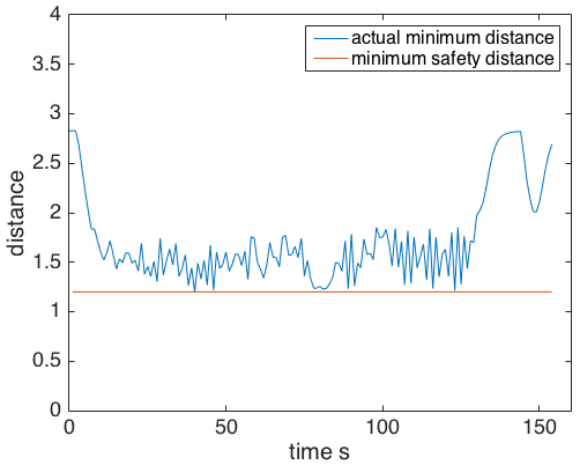

Fig. 7. Minimum pairwise inter-agent distances.

low-level controller remained the same. The simulated path was shown in Figure 8. Obviously, traffic jam caused close

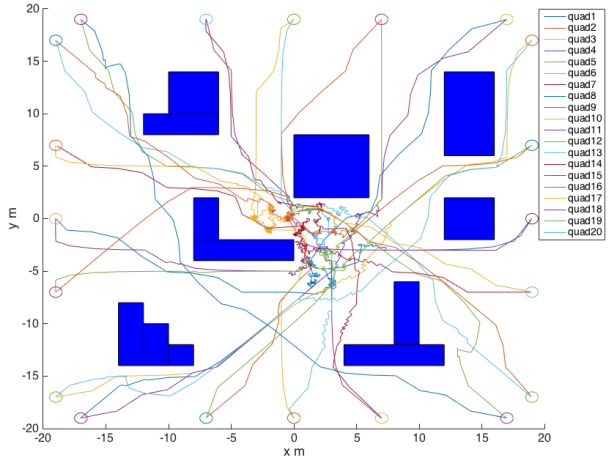

Fig. 8. Agents' Paths with the Un-prioritized A^* Algorithm

to the center of the map while agents move to their destinations results in chattering trajectories when no priorities are considered; in constrast, the use of priorities produces smoother paths for the robots. The total time consumed by the unprioritized algorithm was 206.3s, and the total path length traveled was 1168. For the prioritized algorithm, the the total time consumed was 175.5s, and the total path length traveled was 1078, which gave a 15% reduction in time and a 8% in path length.

B. 3D case

The second simulation involves $N = 8$ agents flying in a 3D environment populated with $M = 5$ cuboid obstacles of known to the agents geometries and positions. The base of all obstacles is a square with side of length $w = 4$, whereas their heights are between 6 and 10. The effective workspace of dimensions 20 m x 20 m x 10 m is divided into 10x10x5=500 uniform grids of width $w_g = 2$. The parameter values of the simulation are the same as in the previous case.

All agents start from the ground, fly to the departure point and wait until all agents reach the departure point. Then the algorithm starts to work. Table II gives the departure and destination for the 8 agents.

TABLE II

DEPARTURE AND DESTINATION FOR 3D SIMULATION

agent	departure	destination	agent	departure	destination
1	(-9,0,9)	(9,0,1)	2	(0,-9,7)	(0,9,3)
3	(9,0,5)	(-9,0,5)	4	(0,9,3)	(0,-9,7)
5	(-9,9,1)	(9,-9,9)	6	(9,-9,3)	(-9,9,5)
7	(-9,-9,5)	(9,9,7)	8	(9,9,7)	(-9,-9,5)

The paths of the agents are shown in Figure 9, where the spheres denote the destinations of the agents.

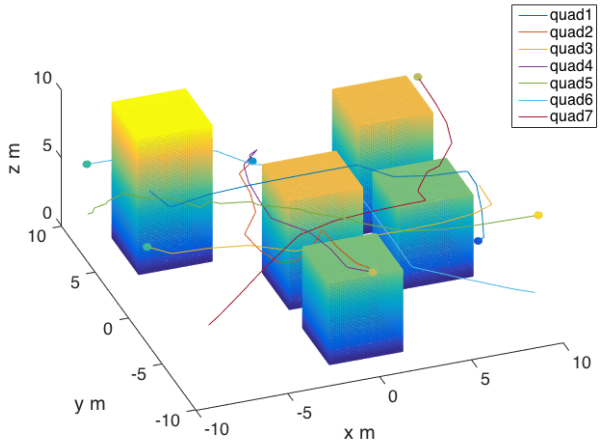

Fig. 9. Paths of 8 Agents in 3D Case (1)

Figure 10 illustrates the resulting linear velocities of the agents under the proposed control strategy.[1]

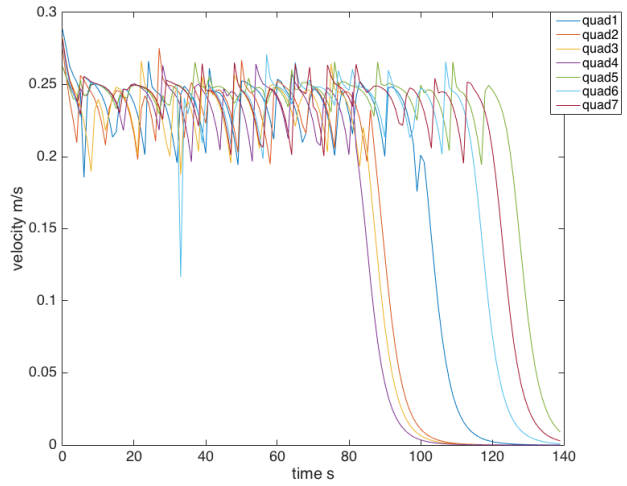

Fig. 10. Velocities of 8 Agents in 3D Case

The smallest pairwise distance among all pairs of agents at each time instant is shown in Figure 11, where the horizontal line depicts the minimum allowable separation. The time needed for A^* to compute the path for each agent (for the selected grid) is less than 0.1 msec.

[1]The initial speed is equal to 1 and indicates the climbing velocity for the agent to reach its departure point. The zero speed after then indicates that the agent is waiting until the other agents reach their departure points.

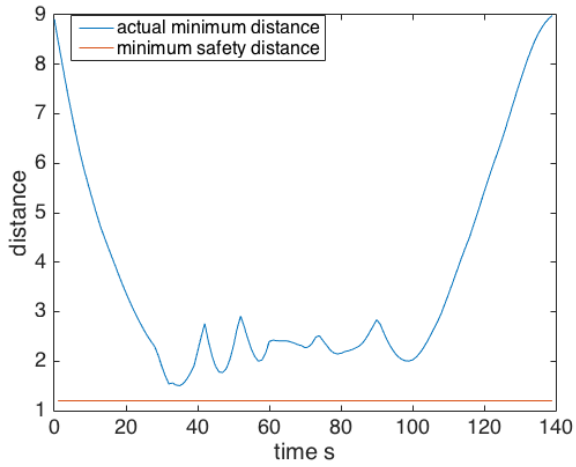

Fig. 11. Smallest inter-agent distance in the scenario of 8 Agents in 3D Case

IX. EXPERIMENTAL RESULTS

The efficacy of the proposed hybrid algorithm is further demonstrated through experimental implementations on a motion planning scenario involving $N = 3$ quadrotors 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 x 4 x 3, which was divided in 48 uniform cubid grid cells of side $w_g = 1$. The parameter values in the simulation are: $R_s = 0.7$, $R_z = 0.6$, $R_c = 0.5$, $d_s = 0.4$, $k_1 = 3$,$fence = 0.3$. The $M = 4$ cuboid obstacles are 1 or 2 m high and have the same 1x1 base.

Figure 12 shows the paths of each agents. The paths are drawn from the time instant after the agents have reached their starting points, which have been set as: $\boldsymbol{r}_{10} = \begin{bmatrix}-1.5 & -1.5 & 1.5\end{bmatrix}^T$, $\boldsymbol{r}_{20} = \begin{bmatrix}-1.5 & 1.5 & 1.6\end{bmatrix}^T$, $\boldsymbol{r}_{30} = \begin{bmatrix}1.5 & 1.5 & 1.7\end{bmatrix}^T$,and their destinations were 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 inter-agent distances are shown in Figure 13 and remain always greater than the safe separation d_s, hence the quadrotors are flying along collision-free trajectories.

Finally, the linear velocities commanded by the proposed algorithm, and the actual velocities, calculated as the derivatives of the position data acquired by VICON during the

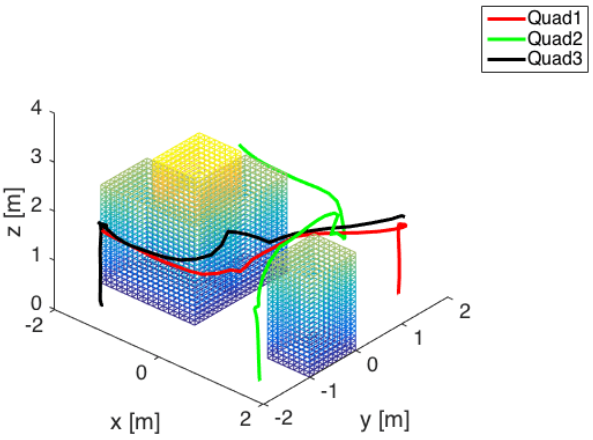

Fig. 12. Paths of 3 Agents in 3D Case

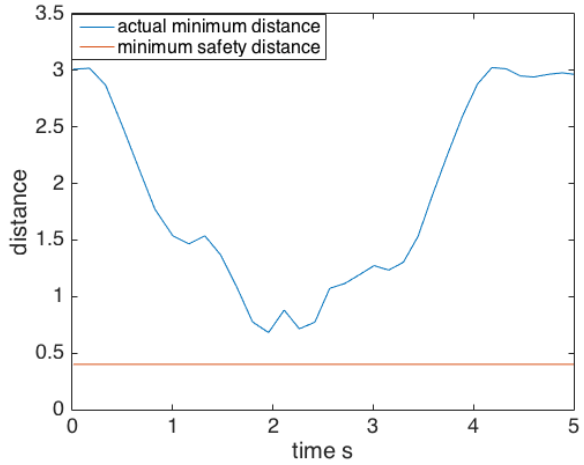

Fig. 13. Pairwise Inter-Agent Distance in the 3 Agent Experiment

experiment, are shown in Figures 14, 15 and 16 for each one of the agents, respectively.

Figures 17, 18 show snapshots of the experimental procedure. The blue lines on the ground indicate the location of the obstacles.

X. Conclusions

This paper presented a two-layer approach for decentralized multi-agent motion planning in 3D polygonal environments which combines a prioitized A^* algorithm along with barrier functions-based coordination and low-level control. The algorithm preserves the guarantees of the collision-free motion of the agents [19] while computing paths through redefining waypoints when appropriate that improve the performance of a stand-alone gradient-based solution in a polygonal obstacle environment. Simulation and experimental results with groups of quadrotors demonstrated the efficacy of the algorithm. Current and future work focuses on extending the algorithm to agents of more complicated dynamics in uncertain environments, that would be relevant in future Unmanned Traffic Management scenarios and applications.

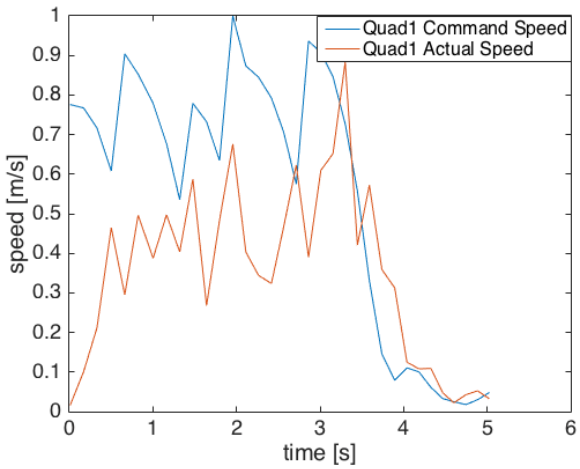

Fig. 14. The control input for agent 1 and its actual linear velocity.

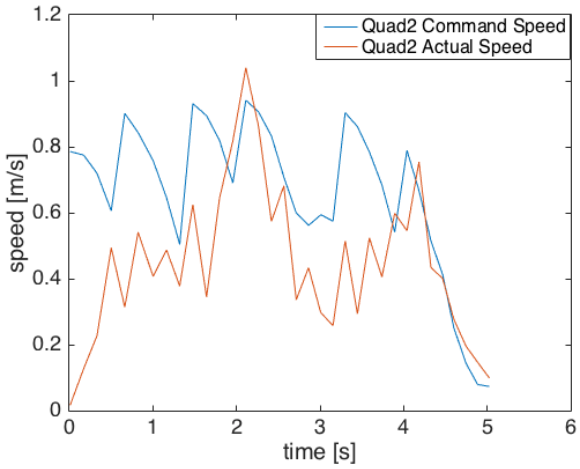

Fig. 15. The control input for agent 2 and its actual linear velocity.

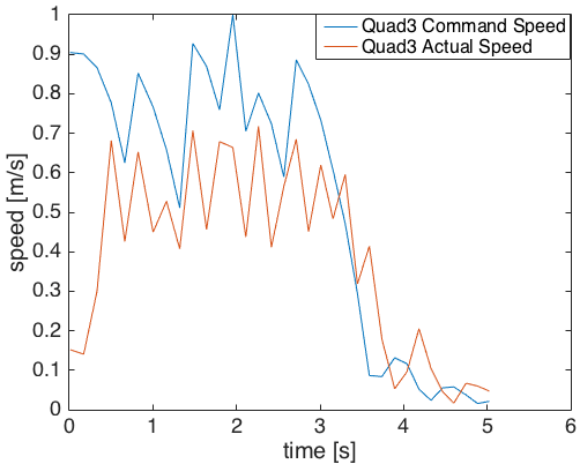

Fig. 16. The control input for agent 3 and its actual linear velocity.

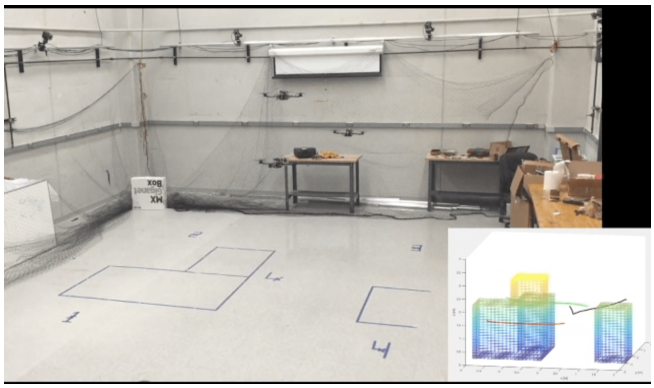

Fig. 17. Experiment Procedure (1)

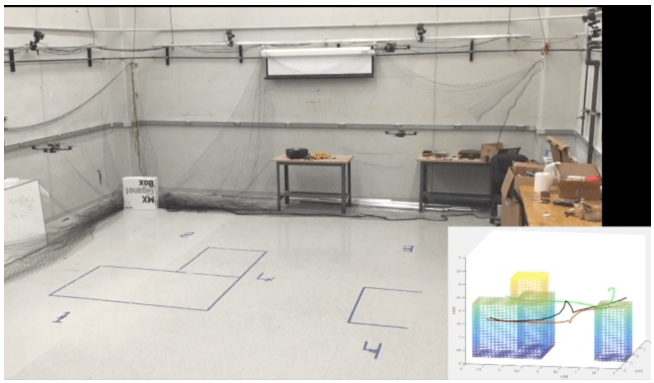

Fig. 18. Experiment Procedure (2)

REFERENCES

[1] 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.

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

[3] 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.

[4] A. M. Ladd and L. E. Kavraki, "Measure theoretic analysis of probabilistic path planning," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 229–242, 2004.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, 2001, pp. 271–276 vol.1.

[16] P. Velagapudi, K. Sycara, and P. Scerri, "Decentralized prioritized planning in large multirobot teams," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, pp. 4603–4609.

[17] T. Zheng, D. Liu, and P. Wang, "Priority based dynamic multiple robot path planning," in *Proc. 2nd Int. Conf. on Autonomous Robots and Agents*, 2004.

[18] J. van den Berg and M. Overmars, "Prioritized motion planning for multiple robots," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug 2005, pp. 430–435.

[19] 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*, pp. accepted as Paper, in press, 2015.

[20] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Stryk, *Simulation, Modeling, and Programming for Autonomous Robots: Third International Conference, SIMPAR 2012, Tsukuba, Japan, November 5-8, 2012. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo, pp. 400–411. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34327-8$_3$6