RRT-based 3D Path Planning for Formation Landing of Quadrotor UAVs

Yiqun Dong[1,2], Changhong Fu[1,2] and Erdal Kayacan[1]
[1]School of Mechanical and Aerospace Engineering
[2] ST Engineering-NTU Corporate Laboratory
Nanyang Technological University, 50 Nanyang Avenue, 639798 Singapore
Email: {dongyq, changhongfu, erdal}@ntu.edu.sg

***Abstract*—This paper discusses the formation landing problem of quadrotor UAVs, which is considered as a UAV leader-follower problem, avoiding static obstacles. Rapidly-exploring random tree algorithm is used to generate the path for the leader UAV firstly. In particular, specifics of tree-grow including nodes selection, parent node connection, feasible and optimal path generation are explained. Given the leader UAV position, path finding for the follower UAV is conducted to avoid both static obstacles and the leader quadrotor. Based on the intensive simulations, which are conducted in ROS-Gazebo environment, the proposed framework is considered to be applicable in real-time formation landing of quadrotor UAVs.**

I. Introduction

Quadrotor UAVs are relatively simple, affordable and easy-to-manipulate aerial robotic systems. Even if they have turned out to be reliable and efficient testbeds in both civilian and commercial missions in the past two decades, e.g., wildlife conservation [1], midair collision surveillance [2] and unknown environment exploration [3], the space for mounting different onboard sensors, payload capacity, flight time and area of a single quadrotor is still low. Therefore, cooperation of quadrotors can achieve better performances in those UAV missions.

In literature, formation control of quadrotor UAVs have been introduced in many works, e.g., [4] [5], however, these works mainly focused on the formation control of the UAVs in midair instead of landing process, and no or few obstacles have been considered in their applications. Although landing of quadrotor UAV on different objects has been also presented, e.g., [6] [7], these works have only discussed the landing with one single quadrotor UAV without the presence of obstacles. For the UAV landing in outdoor environments, a number of obstacles, e.g., trees and buildings, may appear, the implementation of path planning or re-planning is critical for the quadrotor UAVs to find effective trajectories to follow, thereby achieving the landing task successfully.

In this paper, the formation landing problem of quadrotor UAVs has been discussed, path planning for avoiding obstacles has also been involved. In literature, many algorithms can be used for the path planning or re-planning purpose, two well-known of which are artificial potential field [8] and A* [9]. For the formation landing problem, flight environments of the quadrotor UAVs are significantly different for variant missions. For the artificial potential field approach, it is hard to build up a potential field that can deal with all these cases. Additionally, the A* approach normally employs only kinematics model of the vehicle in path finding, the dynamics of the vehicle can be complicated, and the path exported from the A* can be infeasible for the vehicle to track in real applications.

A more effective solution to address the formation-landing issue is the rapidly-exploring random tree (RRT) algorithm [10]–[12]. Basically, the only interaction with external environments in the RRT is the collision detection of vehicle navigaiton, which is transferable to different situations. The RRT is usually considered as a versatile algorithm which can deal with varaint configurations. Moreover, only feasible nodes are added on the tree in the tree-grow stage of the RRT, and the finally delivered path is inherently feasible. Therefore, this paper has utilized the RRT for the UAV formation landing based on above two advantages. In addition, the robot operating system (ROS) and Gazebo tool have been employed to test and analyze our presented UAV formation landing algorithm.

The structure of this paper is organized as follows: Section II introduces general configuration and dynamics of the quadrotor UAV. Section III presents the RRT-based path planning for the formation landing of quadrotor UAVs. The path optimization and trajectory tracking have been proposed in Section IV. The intensive experiments are discussed in Section V. Finally, the conclusion is presented in Section VI.

II. Quadrotor Dynamics and Control

General configuration and body-axis definition of the quadrotor is shown in Fig. 1. Given the thrusts generated by four electric-driven motors (T_i) and the torque induced from

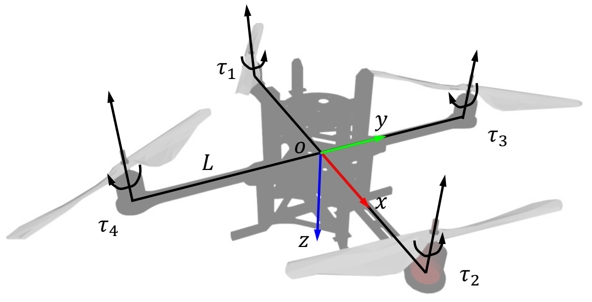

Fig. 1. General configuration and reference definition of quadrotor UAV

thrust:

$$\tau_i = K_\psi T_i, \tag{1}$$

total thrust and rotational torque along each axis of the vehicle body coordinates system is:

$$\begin{bmatrix} u_z \\ u_\phi \\ u_\theta \\ u_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & L & -L \\ L & -L & 0 & 0 \\ K_\psi & K_\psi & -K_\psi & K_\psi \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}, \tag{2}$$

Thrust delivered from each motor is controlled by the input voltage:

$$T_i = K\frac{\omega}{s+\omega} u_{i(i=1,2,3,4)}, \tag{3}$$

note that a first-order system is used to capture motor dynamics, also an input threshold of $u_i \in [0\ 0.05]$ exists.

Equations of the quadrotor motion in world-axis yields [13]:

$$\begin{cases} m\ddot{x} = u_z(cos\phi sin\theta cos\psi + sin\phi sin\psi) \\ m\ddot{y} = u_z(cos\phi sin\theta sin\psi - sin\phi cos\psi) \\ m\ddot{z} = u_z cos\phi cos\theta - mg \end{cases}, \tag{4}$$

rotational dynamic equations are:

$$\begin{cases} I_x\dot{p} + (I_z - I_y)qr = u_\phi \\ I_y\dot{q} + (I_x - I_z)rp = u_\theta \\ I_z\dot{r} + (I_y - I_x)pq = u_\psi \end{cases}, \tag{5}$$

and rotational geometrics are:

$$\begin{cases} \dot{\phi} = p \\ \dot{\theta} = q \\ \dot{\psi} = r \end{cases}, \tag{6}$$

In Equation (4-6) note that nonlinear terms of the quadrotor motion equations originates from position and rotational dynamics coupling; for (6) a premise is involved that the quadrotor is already trimmed at initial steady hovering state, nonlinear terms can be excluded from the equations.

In linearization of quadrotor dynamics, the vehicle is trimmed at steady hovering state. Due to geometrical symmetries of the quadrotor, thrust from each motor is $T_i = mg/4$; linearization of Equations (1-6) can be performed using small-perturbation approach; by specifying the state vector

$$\mathbf{x} = [u\ v\ w\ p\ q\ r\ \phi\ \theta\ \psi]^T,$$

and control input:

$$\mathbf{u} = [T_1\ T_2\ T_3\ T_4]^T,$$

linearized dynamics yields:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \tag{7}$$

wherein non-zero components of state-space matrix (A) and the input matrix (B) are:

$$\begin{cases} A(1,8) = g; A(2,7) = -g; A(7,4) = 1 \\ A(8,5) = 1; A(9,6) = 1 \\ B(3,:) = [1/m\ 1/m\ 1/m\ 1/m] \\ B(4,3:4) = [L/I_x\ \ -L/I_x] \\ B(5,1:2) = [L/I_y\ \ -L/I_y] \\ B(6,:) = [K_\psi/I_z\ K_\psi/I_z\ \ -K_\psi/I_z\ \ -K_\psi/I_z]. \end{cases} \tag{8}$$

For above equations note that specifics of quadrotor parameters (mass, inertia moments, etc.,) are detailed in [13], and will not be listed here.

Given the linearized form of quadrotor dynamics in 8, closed-loop controller of the quadrotor can be organized. Linear quadratic regulator (LQR) based frameworks, which are more robust and produce very low steady state errors, are used. For vehicle heading and height control, associated commands are converted to thrust inputs directly. And for vehicle forward/lateral position control, the signals are transformed into the form of pitch/roll angles first, and then delivered for the thrust execution. A general plot of the quadrotor closed-loop controller is illustrated in Fig. 2.

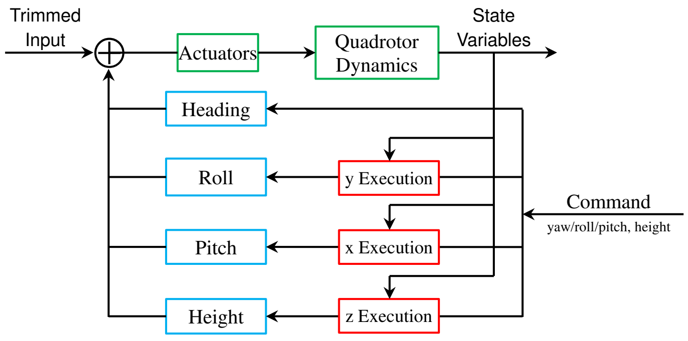

Fig. 2. Flow Chart of LQR-based Controller

III. RRT-based Path Planning

Given start and goal (landing target) position of the quadrotor, the RRT algorithm is implemented for the path finding purpose. the RRT is a sample-based path expansion algorithm. A tree is grown from the start point (tree root), and is strictly biased to the unexplored area of the working space till a certain node/branch on the tree reaches the goal point. Literature review indicates that roughly two categories of RRT have appeared: for the original RRT, the RRT tree grows by sampling input or configuration of the vehicle directly. For configuration sampling, input into the system needs to be calculated reversely. In [14] a so-called closed-loop RRT (CL-RRT) is proposed. This method expands the RRT tree by sampling goal points, and the vehicle is driven to the goal point using a bottom-level controller. Although CL-RRT is computationally more intensive to realize (as an iteration loop needs to be calculated to navigate the vehicle to the goal seed), it provides interface to involve nonholonomic constraints of the vehicle dynamics (minimum/maximum pitch/roll angles, acceleration, pitch/roll rate, etc.). In this paper, the path planning algorithm follows the CL-RRT idea.

A pseudo plot of the path planning algorithm used in this paper can be found in Fig. 3. In the beginning, the start point is added onto the tree as root (line 0). The algorithm detects whether or not the newly added node can see the goal point (line 1). If yes, this node is connected to the goal discretely via a maximum-step propagation. If not, a random seed within the working space is generated (line 2), nodes already-existed on the tree are inspected, and the node to-be-expanded is selected (lines 3-7). In this paper for the RRT tree grow stage, for simplicity purposes quadrotor dynamics are considered as mass point; navigation of the quadrotor is conducted by connecting the node to goal seed via straight line directly. When certain stop conditions are met, this connection will be stopped (lines 8-13). The connection-stop point is designated as the new node to be added onto the tree (line 14); certain information pertaining to node number, distance to the tree root, and node connection is also stored in the newly-added node (line 15). In following we detail each part of the algorithm separately.

```
Algorithm Tree_Grow()
0:  Add start point; designate this point as root node of the tree.
Repeat
1:  If the newly-added node can "see" the goal point,
    connect this node and goal using a maximum-step
    propagation, break Repeat.
2:  Generate a random seed within the working space.
3:  for nodes already-existed on the tree:
4:      Calculate the Euclieаn distance between node and seed;
5:      Add this distance to the Euclieаn distance stored in each
        node as total;
6:      Find the parent node with a minimum total distance.
7:  end for
8:  Navigate the tree from parent node to seed until:
9:      The navigation goes out of the working space; or
10:     Navigation collides with obstacles; or
11:     Maximum length is met; or
12:     Navigation falls into certain neighboring region of the
        random seed.
13: end Navigate
14: Add the navigation-stop point to tree as new node.
15: Store:
        Node distance between this new node and tree root;
        Traveling distance between new node and tree root.
        Parent node number.
    into the new node.
end Repeat
16: Trace back from finally-added node to the tree root to
    generate the feasible path.
```

Fig. 3. General RRT algorithm structure

A. See the goal point and maximum-step propagation

In original RRT algorithm the tree nodes are expanded randomly and strictly biased to the unexplored area of working space. This expansion will not stop till a certain node on the tree branch pokes into a predefined neighboring-hood of the goal point. In this paper, a different algorithm-stop mechanism is used (line 1). For each newly-added node, a (straight line) connection from this node to the goal point is generated. If this connection is collision-free, this node is defined as being able to see the goal point. It will be connected to the goal point directly, and tree grow will stop, see the plot on Fig. 4 for reference.

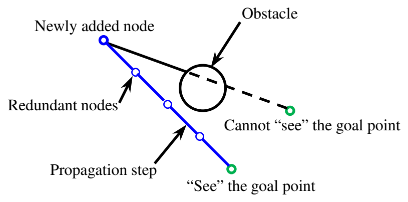

Fig. 4. General plot of final "see-to-connect" propagation

In this see-to-connect stage, to leave redundant nodes on the tree branch for afterwards optimization (to be discussed later), step for node propagation does not exceed a predefined level (maximum propagation step) in each iteration. Since a feasible path has already been found, the definition of maximum-step affects overall path-finding time only, and a smaller step corresponds to a longer computational time. In this work this step is set to the same as maximum step for the vehicle navigation (to be discussed later).

B. Seed generation and parent node selection

If the newly-added node cannot see the goal point, the RRT tree will continue growing. In order to guide the direction of tree-grow, a seed is generated (line 2). In literature for rapid and optimal tree-grow purpose, some efforts have been devoted to manipulating generation of this seed, to name a small portion of which readers may refer to [15] [16]. These works, however, are shaped for some specific working spaces or environments only, and transferability of them to distinct working space is doubtable. In this paper the seed generation follows the original manner as described in [12]–[15]. Generally around 90% of the overall seeds are spread randomly in the working space, while final goal point is designated as seed for the remaining cases directly (10%).

For the to-be-expanded node (parent) selection, again for rapid and optimal purpose some manipulations were proposed. Based on authors trial tests, these works takes effects only when nodes number for the RRT tree grow is high enough. For the work in this paper, a small number of nodes is needed, and the claimed effects from [15] [16] are not obvious, not mentioning that rapidness of the tree grow will be jeopardized (some computational power has to be spared for the manipulations). Parent node selection in this paper, therefore, also follows the original RRT approach illustrated in [10]–[12]. Generally the Euclidean distance between the seed and nodes on the tree are calculated; by adding this distance to the stored distance in each tree node (distance the node has to cover before reaching the tree root), total distance between the seed and tree root can be obtained; the node via which the random seed could approach tree root at a lowest distance is selected as the parent node (lines 3-7).

C. *Navigation and collision detection*

Given the generated seed and selected parent node, this node is navigated to the seed till certain stop conditions are met (line 8-13). As has been mentioned for tree-grow stage, quadrotor dynamics are considered as mass point; while in real test real response (simulation) of quadrotor will be involved, in this paper the quadrotor navigation is modeled as a straight-line connection.

The core of conducting navigation lies in the stop conditions defining (lines 9-12). For line 12 the outcome is obvious: since the navigation has reached goal seed, it can stop. For line 11 similar to the final see-to-connect stage, a maximum length of navigation is defined. Via smaller step the RRT tree can explore the working space more thoroughly, although some longer computational time is needed to fully grow the tree. In this paper a maximum step of 0.5 (meter) is considered a balance between working space exploration and computation.

Lines 9-10 belong to the collision detection work. For line 10 the result is straightforward: the quadrotor is not allowed to go out of the working space, and therefore navigation of mass point is not allowed either. For line 9, the detection of collision with obstacles is needed. Certain information pertaining to obstacles positions and dimensions need to be determined. While in real tests indoor positioning system (OptiTrack/Vicon) or/and image processing-based simultaneous localization and mapping (SLAM) algorithms [3], [17] can be used, in this paper for simulation tests, this information is pre-known.

As will be discussed in following test results, generally obstacles are modeled as round pillars. In authors previous work some geometrical-intersection examinations were used to decide 2D rigid body collisions [18]; this work can be extended to the 3D configurations (although more computational resources needs to be spared). In this paper for simplicity purposes, geometrical dimension of the quadrotor is also reduced as mass point. To compensate for real size of the quadrotor, the round pillars used in path planning is slightly larger than real size of the obstacles. When the navigated point falls into the obstacle pillars, a collision is detected, and the navigation process will stop.

Given that quadrotor dynamics is modeled as mass point in this work, an intuitive approach for conducting the navigation process is to connect parent node to seed via a line segment of length 0.5m (maximum step) directly, see the blue circles on Fig. 5. In some circumstances, a stride as long as 0.5m can be too large, end point of this line-segment may poke into obstacles or go outside the working space as shown on Fig. 5; the tree-grow opportunities using such rule will simply be lost. In this work to take full advantage of each tree-grow iteration, a propagation method is used. Generally end point of the vehicle navigation line-segment is shifted to the direction of generated random seed gradually, a smaller shift step of 0.1m is used; see Fig. 5.

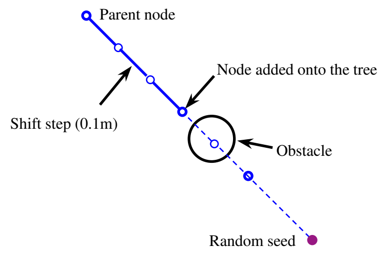

Fig. 5. Node propagation using smaller shift step

D. *Node store and feasible path tracing*

In lines 15 of the algorithm, some information pertaining to node number distance, Euclidean distance, and parent node number are stored in each newly added node. The node number distance indicates overall number of the nodes that has been grown on the tree; in future work some tweaked RRT algorithm may be used (RRT*, for instance), the including of overall node number provides an interface for such purpose (in RRT* some manipulation based on overall node number will be used).

Euclidean distance is stored for the selecting of parent node. As has been mentioned the nodes already-existed on the RRT tree will be inspected, and the one via which the seed can connect to the tree root at a lowest distance will be selected as parent node. After each tree-grow iteration is completed, navigation distance between newly added node and its parent is added onto the distance stored in parent (distance stored in the first parent of the tree root is 0), and will be stored in the newly added node.

For each node on the tree, while multiple children nodes can exist, only one parent is mapped to each node. In future work simulation using quadrotor dynamics and closed-loop controller will be involved for the tree expansion, and 3D collision detection using real size of quadrotor/obstacles will be implemented; nodes added onto the RRT tree are all feasible nodes (non-collision subject to nonholonomic constraints). In this work some simpler navigation process using straight line connection is used; all the nodes added onto the tree are connected via collision-free line segment. Given the final goal point, its parent node is the last node added onto the tree. Based on the parent node information stored in line 15, the feasible path can be generated by tracing back from goal point to its parent till the first parent of vehicle start position (tree root). Given that each node connection of the RRT tree is feasible, this traced path is inherently feasible too. Certainly due to the random manner involved in tree-grow, this path could be zigzagging; some optimization approach will be adopted to remove redundant nodes from this path for optimization, see following.

IV. Path Optimization and Trajectory Tracking

For purpose of rapid and optimal path planning, in literature some tweak of the original RRT algorithm haven been proposed [15] [16]. As has been mentioned, these methods take effect only when nodes number on the RRT tree is high enough. In this work based on trial tests, the overall optimization effects are not obvious, not mentioning that speed of the algorithm will be damaged; frameworks in [15] [16] will not be used.

In this paper an off-line optimization approach similar to [18] is used. Generally, since the feasible path has been generated from the RRT-grown tree, and it consists of a chain of nodes connections, the optimization process is conducted by randomly picking up two nodes from the feasible path, and trying to connect them directly. In real test work quadrotor dynamics and closed-loop controller will be involved, this connection will be performed using quadrotor dynamics simulation. In this work quadrotor dynamics is considered as mass point, a straight line is used. If the connected line segment is collision free, and since length of this straight-line segment will be shorter than original (zigzagging) path, the original path between these two randomly-picked points will be replaced using the new connection; or if the connected line segment is not collision-free, an empty-trial will be returned. Given that we are always trying to prune the original path using a new shorter path, overall length of the original path evolves at a non-increasing manner, and when algorithm-stop conditions are met (overall optimization iterations/empty-trials exceeding a pre-defined level), the returned path is considered an optimized one.

After optimization, the path is delivered to the bottom-level controller for execution. Nodes connected on the path are designated as sub-goals for the LQR-based closed-loop controller, and the vehicle is attracted to the final goal point gradually. When the vehicle falls into a predefined neighboring-hood of the goal, overall trajectory tracking mission is completed.

V. Simulation Test Results

In this paper simulation tests in two different scenarios were involved. For briefness issue, only one test is presented in here. Obstacles are modeled as round pillars, and two quadrotors are involved to perform the formation landing mission. Final tests are performed and presented in the ROS[1]-Gazebo[2] environment, this environment provides a seamless interface with hardware; algorithms and control laws validated in this environment (programmed in Matlab, C/C++, etc.) can be applied to further field tests using real UAVs directly.

Test scenario plotted in Gazebo can be found in Fig. 6; the general goal is to land the two UAVs on far side to the ground vehicles located close to the origin (yellow). Or as in Fig. 7, leader and follower quadrotors start from blue crosses; multiple (black round pillar) obstacles are placed in the configuration space. The leader and follower are expected to fly though the obstacles, and to land on red circles. Given start/goal locations of the vehicle and obstacles positions/dimensions, path planning for leader quardrotor using RRT algorithm is conducted, see Fig. 8; nodes are added onto the RRT tree (cyan), and a feasible path can be connected (magenta). Redundant points on the path are removed, and optimized path is returned (blue circle lines).

For follower quadrotor path planning, the general framework (RRT algorithm) remains the same, only difference being that leader quadrotor is also taken into account as a (static) obstacle. Given the blue circle path of leader, path finding for follower is performed to avoid both obstacles and the leader moving along this path, see the nodes grown on the tree (cyan points); after feasible path is connected (magenta), optimized path (blue circle line) can be delivered, see Fig. 8.

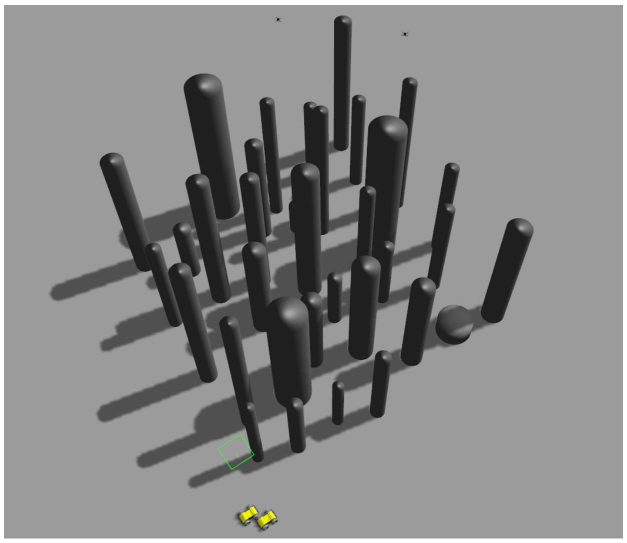

Fig. 6. Plot of test configuration in ROS-Gazebo environment

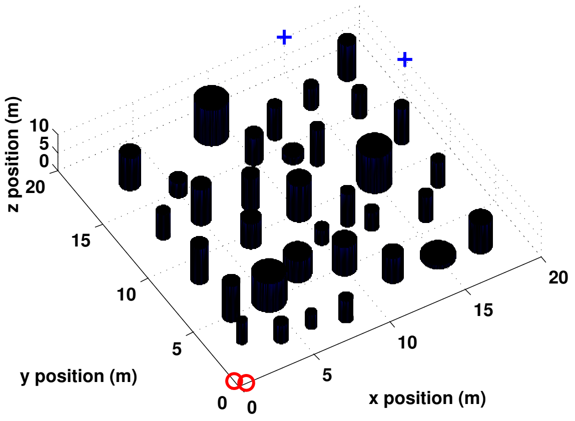

Fig. 7. Plot of test configuration in Matlab environment

Trajectory tracking of the quadrotors is performed in Gazebo using the LQR-based controller; see vehicles response history in Fig. 9. Note that in tracking, the simulation adopts

[1]See more at: http://www.ros.org/.

[2]See more at: http://gazebosim.org/.

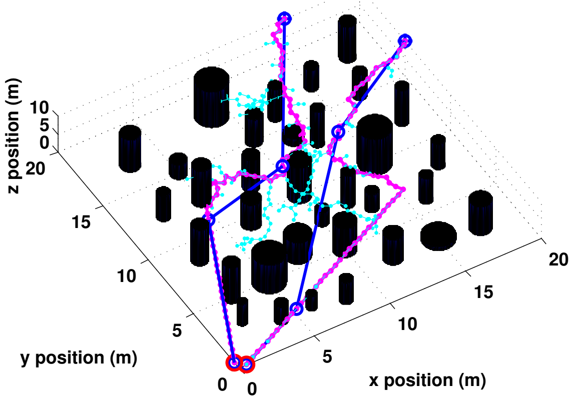

Fig. 8. Path finding for leader and follower quadrotors

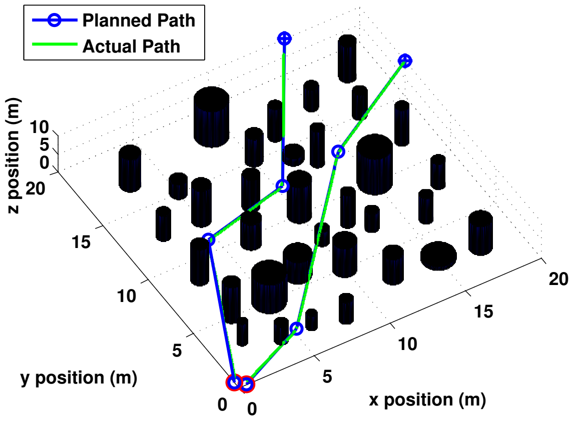

Fig. 9. Trajectory tracking of leader and follower quadrotors

rigid-body dynamics of the vehicle (not mass point). As in Fig. 9 both vehicles take off from the start (blue crosses), and fly to land on the target (red circles) simultaneously. Given the plot in Fig. 9, trajectory-tracking of the vehicle is considered to be success; framework proposed in this paper is applicable.

VI. Conclusion

This paper discusses the unmanned quadrotor formation landing problem in face of static obstacles. As preliminary exploration, quadrotor dynamics/dimension are modeled as mass point in this paper; RRT algorithm is used for the leader to find a feasible path first, and path finding for follower is conducted by taking into account both static obstacles and the leader quadrotor. Tree nodes selection, random seeds generation, and path optimization are presented. Final test is conducted in ROS-Gazebo environment. Given the planned path from RRT, the LQR-based bottom-level controller is used to track this path. Experimental results shows that the proposed framework can be applicable in real-time formation landing of quadrotor UAVs.

For future work, the authors plan to run field tests using real quadrotors. In the first step, indoor positioning systems will be involved, while for later development some image processing-based localization and mapping approaches may be used.

Acknowledgment

This research was partially supported by the ST Engineering-NTU Corporate Lab through the NRF corporate lab@university scheme. The authors are also indebted to Mr. Ran Duan with ST Engineering-NTU Corporate Lab for his illustrating ideas of computer geometrics for the collision detection work.

References

[1] M. A. Olivares-Mendez, C. Fu, P. Ludivig, T. F. Bissyande, S. Kannan, M. Zurad, A. Annaiyan, H. Voos, and P. Campoy, "Towards an Autonomous Vision-Based Unmanned Aerial System against Wildlife Poachers," *Sensors*, vol. 15, no. 12, pp. 31 362–31 391, 2015.

[2] C. Fu, A. Carrio, M. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Robust real-time vision-based aircraft tracking from Unmanned Aerial Vehicles," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 5441–5446.

[3] C. Fu, A. Carrio, and P. Campoy, "Efficient visual odometry and mapping for Unmanned Aerial Vehicle using ARM-based stereo vision pre-processing system," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, 2015, pp. 957–962.

[4] M. Turpin, N. Michael, and V. Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Autonomous Robots*, vol. 33, no. 1, pp. 143–156, 2012.

[5] T. Nageli, C. Conte, A. Domahidi, M. Morari, and O. Hilliges, "Environment-independent formation flight for micro aerial vehicles," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1141–1146.

[6] M. A. K. K. A. Ghamry, Y. Dong and Y. Zhang, "Real-time autonomous take-off, tracking, and landing of uav on a moving ugv platform," in *Control and Automation (MED 2016), 24th Mediterranean Conference on*, 2016.

[7] K. Mohta, V. Kumar, and K. Daniilidis, "Vision-based control of a quadrotor for perching on lines," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3130–3136.

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

[9] B. Stout, *The basics of A* for Path Planning*. Game Programming Gems, Charles River Media, 2000, pp. 254–263.

[10] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[11] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, 2000.

[12] P. Cheng and S. M. LaValle, "Reducing metric sensitivity in randomized trajectory design," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, 2001, pp. 43–48.

[13] D. Yiqun, F. Jun, Y. Bin, Z. Youmin, and A. Jianliang, "Position and heading angle control of an unmanned quadrotor helicopter using lqr method," in *Control Conference (CCC), 2015 34th Chinese*, 2015, pp. 5566–5571.

[14] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[15] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004.

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

[17] C. Fu, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Monocular visual-inertial slam-based collision avoidance strategy for fail-safe uav using fuzzy logic controllers," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 513–533, 2014.

[18] Y. Dong and Y. Zhang, "Application of rrt algorithm to unmanned ground vehicle motion planning and obstacle avoidance," in *11th International Conference on Intelligent Unmanned Systems (ICIUS 2015)*, 2015.