RTDP-based algorithm for real-time obstacle avoidance and distributed motion planning

Abdullah Mohiuddin[1] Tarek Taha[2] Yahya Zweiri[1,3] Dongming Gan[1]

***Abstract*— We propose a cascaded real time dynamic programming based algorithm. The first block searches obstacle-free least distance path and the second block performs velocity optimization on that path to either achieve minimum energy or minimum time motion while generating dynamically feasible trajectories. The algorithm can be deployed to multiple UAVs, in a distributed fashion where UAVs have to cooperatively work towards a common goal. The cascaded motion planner was first tested in simulink based model, and after that software in the loop simulations were performed. The simulation tests confirmed the effectiveness and applicability of the method.**

I. INTRODUCTION

UAVs are being considered for their potential of payload delivery in an efficient and quick manner. In 2016 Amazon prime air performed the first drone-based delivery in Cambridge England [1]. In cases where UAVs have to reach the destination location as early as possible such as while delivering medical samples or organs, the most important factor is the reduction of journey time. In cases where time factor is not important, we would like to save energy consumption for the journey [2]. In all cases, we need to avoid obstacles and select the shortest euclidean distance. Selection of shortest obstacle-free path in real-time is a challenging task. Modern computational platforms have improved in efficiency and are reduced in size to be able to be attached on-board of such UAVs. The improved capability of the computational platforms made it possible to use algorithms such as Real-time dynamic programming for the optimization purposes in real-time.

II. RELEVANT WORK

Researchers have focused on multi-rotor motion in cluttered environments. In [3] developed a method to find time-based obstacle free polynomials in real-time. Study in [4] focused on a real-time trajectory re-planning approach to deal with dynamic environment while keeping the multi-rotor on a global trajectory. In [5] first safe flight corridors were calculated and then the safe flight corridors were used to continuously plan the motion of the quad-rotor. In [6] the trajectories are generated, not only considering the obstacle free path, but also taking into account for the time and control efforts. In [7], the focus was on dealing with motion uncertainty, field of view constraints in dynamic obstacles. In [8], the path searching method finds kinodynamically feasible, minimum time trajectories. In [9], authors presented collision avoidance algorithm for multiple UAVs in the communication range. In this method, altitude difference was used to ensure collision avoidance. In [10], MDP based method is used to find obstacle free trajectory for a fixed winged UAV. The trajectories were modified to make them dynamically feasible using differential flatness property. Similar to [10], we also propose a Dynamic Programming (DP) based approach for searching the feasible path, but differently to [10], our grid based search approach is valid for holonomic systems. In our method we discretize the map into grid points and then use Dynamic programming based method to find the shortest obstacle free path. This path is then used to generate dynamically feasible minimum-time or minimum energy trajectories, using our previously developed multi-criteria optimization approach [2].

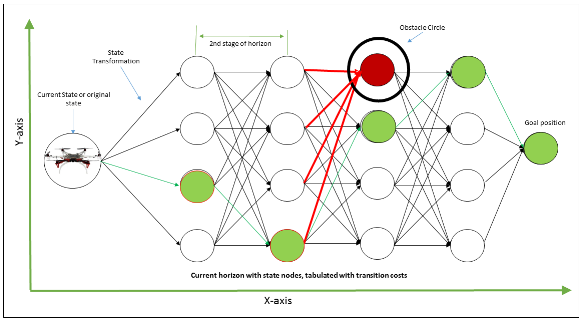

Y-axis
X-axis
Current State or original state
State Transformation
2nd stage of horizon
Obstacle Circle
Goal position
Current horizon with state nodes, tabulated with transition costs

Fig. 1. Process flow diagram showing the DP sweep. Current position of the drone is taken by algorithm and a grid of points is created. Transition costs are calculated for each possible decision. The transition costs towards the node which is within the obstacle circle gets more cost value. Finally the minimum cost path is traced back to the drone and the selected path is implemented

III. OPTIMIZATION ALGORITHM

This paper therefore extends our previous work where the UAV was moving from point A to point B and we were using a multi-criteria approach to optimize either journey time or energy via using RTDP algorithm. The newly developed RTDP path planner generates the obstacle-free shortest path. The first straight path is selected from the generated optimum path and cascaded algorithm as developed in [2] is then used on the first straight path. The contribution of this paper is a distributed control for real-time obstacle-free energy optimized motion of multiple UAVs. The process includes two (Dynamic programming) DP sweeps, one with the cost function J_o as defined in section III-A, the other one J_k discussed in [2]. The overall algorithm is based on minimization of the cost functions defined as follows:

*This work was supported by KUCARS

[1] Khalifa University abdullah.mohiuddin@ku.ac.ae
[2] Algorythma lab
[3] Kingston University

$$\min_{d} J_{PN} = \sum_{N-1}^{k=0} J_o(d_t, O_f) \tag{1}$$

$$\min_{V} J_{VN} = \sum_{N-1}^{k=0} J_k(J_e, J_t, \lambda, d) \tag{2}$$

To avoid repetition we will not explain the cost function J_{VN}, which is already explained in our previous work in [2]. The DP algorithm requires that the system should be described in a discrete model. We describe the system model as follows:

$$X_{k+1} = f_k(X_k, u_k, z_k), k = 0, 1, 2, ...N-1 \tag{3}$$

Where X_k represents the current state of the drone,

$$X_k = [x_k, y_k] \tag{4}$$

Where u_k represents the next decision,

$$u_k = [u_x, u_y] \tag{5}$$

where u_x and u_y represent the next grid point in the search space. Where z_k represents obstacle matrix,

$$z_k = [z_x, z_y, z_r] \tag{6}$$

where z_x, z_y and z_r represent arrays containing the obstacle x,y coordinates and the radius of the obstacle.

The first part of the cost function is calculated in first DP sweep, as done in Fig. 1. The plane of transportation is divided into a grid. Each point on this grid represents a node or state. The obstacles are represented as circles. First transition costs from each node to the other node of consecutive states are calculated and stored. The transition costs are calculated using the cost function defined in III-A. After calculating all transition costs from first to last stage, the minimum cost path is traced back to the first node. After tracing the minimum cost path, the path is used as an input for the algorithm as defined in [2] to obtain the optimized velocity command.

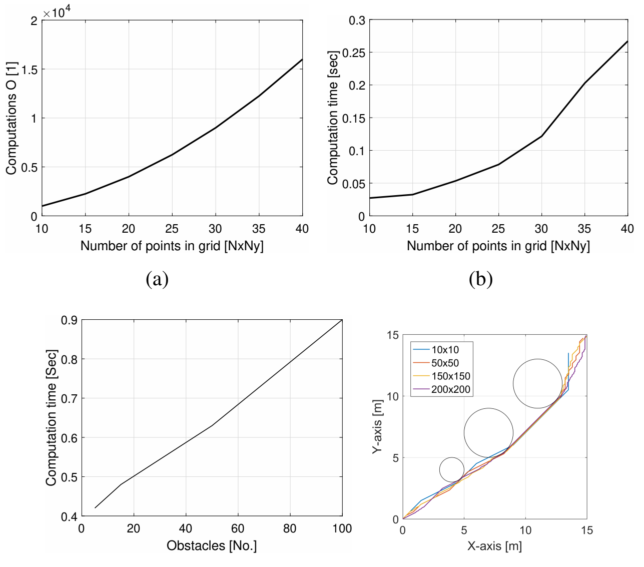
A. Definition of cost function

The cost function is defined as a multiple of Euclidean distance and an obstacle factor as $J_o = d_t \dot{O}_f$. Where $d_t = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$, where $i + 1$ represents the coordinates of the next state and i represents the coordinates of the current state. $\dot{O}_f$ is the length of the whole journey which allows for maximum cost for any state which lies inside the obstacle circle. The obstacle factor penalizes the path decision that leads into the circle defining the obstacle region.

B. Grid size selection

The grid composition and size can effect the computational costs and execution time. The resolution of the grid is exponentially linked with the computational cost. However, the computational costs vary linearly with number of obstacles.

The complexity of algorithm $O(N_s N_o N_p^2)$, depends on; number of distance intervals N_s between the start and goal position, number of grid point intervals N_p and the number of obstacles N_o.



(a)

(b)

(c)

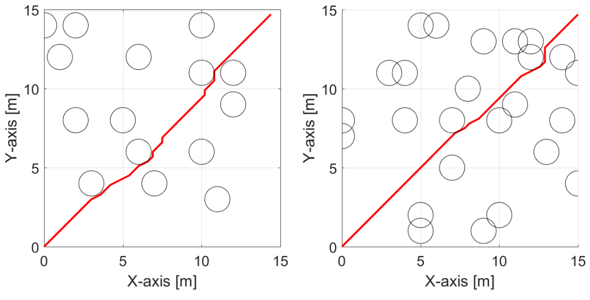
(d)

Fig. 2. The effects of a decrease in grid resolution that results in an exponential increase in (**a**) number of computations with varying grid-points and (**b**) computation time with varying grid-points (**c**) computation time with varying number of obstacles (**d**) Various grid resolutions and resulting paths

Sample computation times and number of computations were plotted in Fig. 2(a) and Fig. 2(b) for varying grid points, which shows that the computational time increases exponentially with the number of grid-points. The analysis in Fig. 2(b) is based on 10 obstacles, where as number of stages were varying with N_p. The number of grid points ranged from $10 - 40$ for this analysis. The computational time, with increasing number of obstacles was plotted in Fig. 2(c). The number of obstacles ranged from 5-100 for this analysis, while the grid size was fixed to $[50 \times 50]$. Grid resolutions ranging from $[10 \times 10]$ to $[200 \times 200]$ were applied on three obstacles and the results were plotted in Fig. 2(d). At lower resolution the path has some intersection with the obstacle circle, and with higher resolutions the path is avoiding the circles properly, however with further increasing the grid resolution does not provide any special benefit. The computation is done using a $2.4GHz$ computer with $8GB$ RAM.

C. Sample results

Ideally a smaller grid size favors the energy savings and perfect obstacle avoidance. However the limitations are due to the computational costs which can effect the real-time application of the algorithm. Several tests were conducted with 15 and 25 obstacles to test the DP algorithms ability to find the least distance obstacle-free path. These tests are shown in Fig. 3



(a)

(b)

Fig. 3. Three sample DP sweep results with 50x50 grid and 20, 25 obstacles, start point is [0,0] and goal position is [15,15] (**a**) 15 obstacles (**b**) 25 obstacles

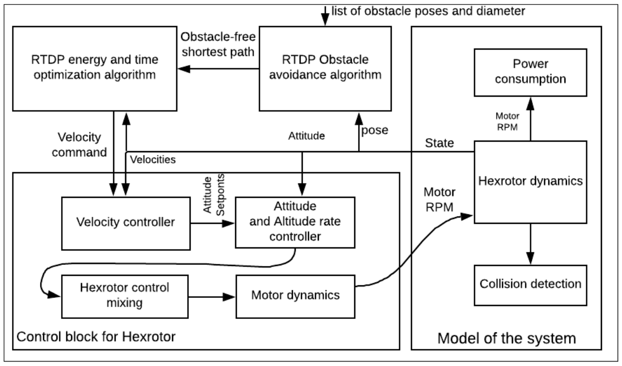


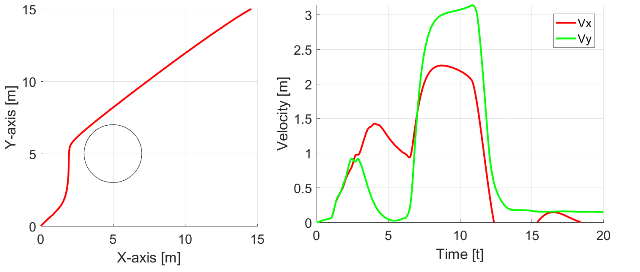
Fig. 4. Description of the complete system for simulation tests of the algorithm, including the controller block, the hex-rotor block, and the optimization algorithm block, collision detection block

D. Assumptions

We assume that the location and size of the obstacles are known to the algorithm or are updated regularly. The obstacle circles are defined to include the drone radius, so that if the drone passes tangentially with the circumference of the obstacle circle, it will not create collision. Since the cascaded DP sweeps involve the DP algorithm defined in our previous work [2], all assumptions of that work are in place here too.

IV. Numerical testing in Simulink based model

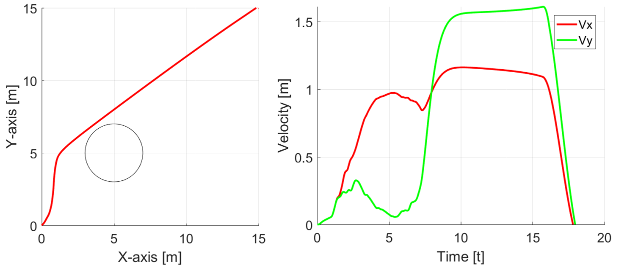
The algorithm is tested using the numerical model as described in [2] The complete description of the energy optimized obstacle avoidance RTDP algorithm can be seen in Fig. 4. The notable difference here from [2] is the presence of two RTDP algorithm blocks, and the collision detection block. Assuming prior information of the obstacle locations, and a fixed goal position $[15m, 15m]$, several tests were performed to verify the effectiveness of the algorithm. The UAV model and all its parameters are defined in our previous work [2]. An obstacle with a diameter $2m$ was considered to be located at $[5, 5]$ during the simulation tests. The factor $\lambda = 0, 0.5, 1$ were used to select the priority between energy or time saving during the tests. Fig. 6(a), Fig. 5(a) and Fig. 7(a) show the results of the obstacle-free path following of the UAV, where the circle shows the obstacle, when $\lambda = 0, 0.5, 1$. Fig. 6(b), Fig. 5b and Fig. 7(b) show the generated velocity profiles using the RTDP algorithm when $\lambda = 0, 0.5, 1$.



(a)

(b)

Fig. 5. The flight plots showing the path and the obstacle and the velocity profiles for $\lambda = 0.5$ (**a**) Obstacle-free path following (**b**) Velocity profile



(a)

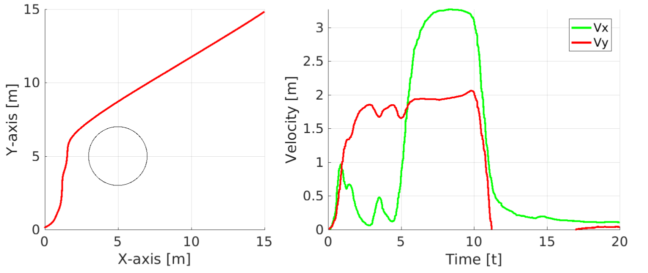
(b)

Fig. 6. The flight plots showing the path and the obstacle and the velocity profiles for $\lambda = 1$ (**a**) Obstacle-free path following (**b**) Velocity profile

The results obtained via numerical simulation show that by changing a factor λ we can alter the journey time while the UAV is continuously generating and implementing obstacle-free path.

V. Software in the loop testing

After performing the numerical simulations, the tests were extended to software-in-the-loop simulations. Special consideration was required to ensure that the publish rates of the



(a)

(b)

Fig. 7. The flight plots showing the path and the obstacle and the velocity profiles for $\lambda = 0$ (**a**) Obstacle-free path following (**b**) Velocity profile

velocity commands do not fall below the $2Hz$ limit so as to not trigger the fail-safe.

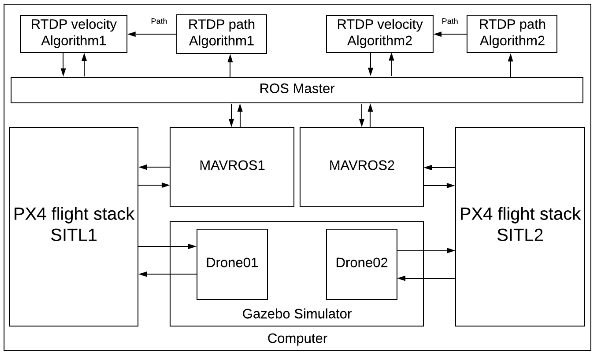


Fig. 8. Description of the complete system for software in the loop simulation tests of the algorithm

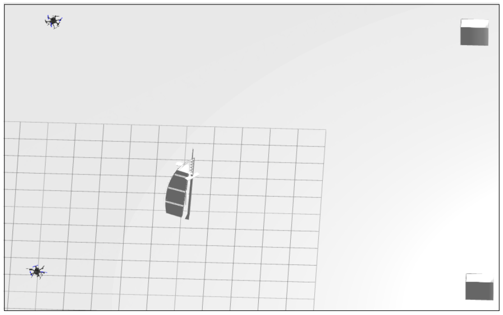


Fig. 9. Dual UAV and static obstacle SITL scenario

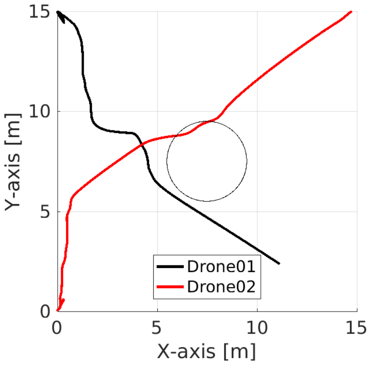


Fig. 10. Dual UAV and static obstacle SITL scenario

A. Multiple UAVs and static obstacles

Further simulation tests were performed, with two UAVs flying in the same plane, such that their paths are crossed. The complete setup of this simulation is shown in Fig. 8. The start and goal positions of the two UAVs are $[0, 0], [15, 15]$ and $[15, 0], [0.15]$. These coordinates were chosen to make sure we select a path which will otherwise result in collision, if our algorithm won‘t be working. In this simulation scenario, it is assumed that the UAVs are constantly sharing their locations with each other. The Gazebo environment for this test is shown in Fig. 9. An obstacle is placed at $[7.5m, 7.5m]$ with an assumed diameter of $2m$. The boxes are placed at the destination points of both drones. The resulting path of the both drones are shown in Fig. 10. It should be noted that although in Fig. 10 the paths of the two drones appear as if they are overlapping at some point. However, this plot does not show the temporal axis. Both drones may have reached at that point, but in different times which are 6.55 and 7.45 seconds respectively. The two drones successfully navigate around the obstacle and avoid collision with each other during their journey.

VI. Conclusion

A novel real-time cascaded motion planner was presented, where in one layer obstacle avoidance shortest path is generated using Real-Time-Dynamic Programming whereas in the second layer the optimum path is used to apply multi-criteria optimization to select velocity decision to favor either reduction of energy consumption or reduction of time spent in the journey [2]. The algorithm is useful for deployment in scenarios where multiple UAVs need to cooperatively navigate in an arena while avoiding collisions. The cascaded algorithm can be deployed in each agent, while the position of moving UAVs or dynamic obstacles can be either detected using on-board sensors, or transmitted via WiFi connections. Future work should include, the cascaded path planner for dual-UAV collaborative transport with obstacle avoidance.

References

[1] Amazon prime. [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/

[2] A. Mohiuddin, T. Tarek, Y. Zweiri, and G. Dongming, “UAV payload transportation via RTDP based optimized velocity profiles,” *Energies*, vol. 12, no. 16, 2019.

[3] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, “Continuous-time trajectory optimization for online uav replanning,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5332–5339.

[4] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, “Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 215–222.

[5] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, “Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments,” *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[6] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, “Search-based motion planning for quadrotors using linear quadratic minimum time control,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2872–2879.

[7] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, “Towards search-based motion planning for micro aerial vehicles,” *arXiv preprint arXiv:1810.03071*, 2018.

[8] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, “Robust and efficient quadrotor trajectory generation for fast autonomous flight,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[9] E. Ferrera, A. Alcántara, J. Capitán, A. Castaño, P. Marrón, and A. Ollero, “Decentralized 3d collision avoidance for multiple uavs in outdoor environments,” *Sensors*, vol. 18, no. 12, p. 4101, 2018.

[10] X. Yu, X. Zhou, and Y. Zhang, “Collision-free trajectory generation and tracking for uavs using markov decision process in a cluttered environment,” *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1-2, pp. 17–32, 2019.