Hybrid Path Planning for UAV Traffic Management

Eyal Zehavi[1] and Noa Agmon[2]

Abstract— **Unmanned Aircraft System Traffic Management (UTM) becomes a highly relevant complex challenge, as the UAV activity is rapidly growing bringing more amateur and professional drones to the urban skies. The main concern of managing such a system is safely navigating and controlling hundreds or thousands of drones simultaneously, flying in a crowded dense environments. This paper introduces an innovative approach of hybrid path planning, which tries to make the best out of the commonly used centralized and decentralized planning approaches. The Hybrid Path Planner (`HPP`) defines two configuration spaces: the *Local Zone*, which represents the crowded city zone with many obstacles and constrains, and the *Global Zone*, which represents the outer suburban zone, mostly open space with predefined flight corridors. The `HPP` server communicates with each UAV, assigning it a close-to-optimal path in the global zone, while leaving the relatively heavy-duty local zone path planning task to be performed by the UAV, mostly using stochastic methods like RRT*. This approach reduces the complex path panning task of the centralized server to a simpler task of calculating only the entry and exit points to and from the global zone. This robust approach supports handling a high number of UAVs, while keeping close to optimal performance.**

I. INTRODUCTION

The technological breakthroughs of recent years brings a revolution to the world of mobility. Highly advanced autonomous air vehicles, affordable and simple to operate, become available and popular. These small Unmanned Aerial Vehicles (UAV), while offering a breakthrough for many innovative applications, are at the same time burdening the already busy airspace, challenging public safety and privacy. Increased use and need of means of transport on one hand and the technological advancement on the other, have led to a development of a new segment of air transport - the urban air mobility (UAM). In the near future, skyline of cities is expected to change. Small UAVs carrying cargo are already in test and trials in many cities around the world [1]. Complex transportation ecosystems are developed to move people by air, expected to populate the low altitude volume above the urban areas within the coming decade. New solutions to air flow management are needed to accommodate the increase in air traffic. This complex challenge has to cope with environmental condition changes, conflicting priorities and high UAV numbers, and requires an adaptive robust solution [**?**]. The need to control this massive volume of UAVs in the city sky, has led to development of UTM (UAV Traffic Management) systems to control the traffic flow.

[1]Eyal Zehavi is with the Faculty of Computer Science, Bar Ilan University, Ramat Gan, Israel 52900, `eyalzehavi@bezeqint.net`

[2]Noa Agmon is with the Faculty of Computer Science, Bar Ilan University, Ramat Gan, Israel 52900, `agmon@cs.biu.ac.il`

[3]Watch video at: https://youtu.be/zmMQUJjTKxQ

A key factor in safe and efficient transportation is path allocation. Each user has its own requirements and limitations, the airspace has its own rules, and the environment is always dynamic and surprising. The challenge of the UTM system is to efficiently assign flight paths for each UAV, such that transportation flow will be maximized, highest level of safety will be maintained, and airspace restriction and limitations will not be violated. There are two leading approaches on how to achieve these goals: centralized and decentralized.
In **Centralized Planning**, path planning is done by a central controller, the controller receives flight plan requests, prepares an optimal plan, and assigns each agent with its customized plan. The advantage of this approach is the capability to optimize and reach maximum social welfare [2]. The main disadvantage is the difficulty to scale up since communication and computing load increases exponentially with number of users [3]. In **Decentralized Planning**, path planning is done by each agent individually. This approach offers maximum flexibility since each agent allocates its own computing resources. The challenge is to optimize the overall plan in order to reach a safe, efficient transportation system.

In this work we introduce the **Hybrid Path Planner** (`HPP`), an innovative architecture within the UTM space designed to maximize the advantages of the centralized and decentralized approaches, offering greater flexibility in supporting UTM operations. Realizing that the path for a UAV may be comprised of different segments requiring different approaches, the `HPP` divides the environment into two segments - *Local Zone* and *Global Zone*. In the local zone each UAV plans its own path, while in the global zone the UAV's path is centrally planned by the `HPP`. The `HPP` assigns each UAV with its entry and exit points to the global zone and its path along the global zone, in order to regulate the traffic and optimize the overall performance of the system. We review in this paper three methods for computing entry and exit points to the global zone, we then analyze and examine empirically the performance of `HPP` computation time and resulting travel distance and show a computationally-efficient algorithm for calculating a UAV's entry and exit points in a 2D configuration space. Finally, we demonstrate the `HPP` architecture in a 3D simulation in a realistic environment.

II. RELATED WORK

The recent growth in demand and complexity of the UTM operations has led to the development of different approaches for an optimized UTM system. In this section we survey relevant academic and practical approaches for such systems.

A. Motion Planning and Path Finding

Motion planning is one of the most basic problems in robotics, which refers to finding a sequence of valid configurations that move a robot from an initial to a goal configuration [**?**]. The problem is generally NP-hard in continuous spaces [4], thus solutions to the problem of motion planning usually include a discrete model of the world, or sampling-based world representation.

A basic algorithm using sampling of the state space to find a path for a robot is Rapidly-growing Random Trees (RRT) [5] (later extended to RRT* [6], and many other variants), used vastly for UAVs. Due to the multi-dimensionality of the state space, there is no algorithm that provides an exact analytic solution to such a problem. Even approximation algorithms operating on a 3D subspace of this problem space are difficult to compute in real time [7]. Comparison of path planning algorithms for UAVs with respect to optimality and computational complexity can be found in [8].

Another approach to reduce complexity was introduced in [9], an effective, cooperative, probabilistically-complete multi-robot motion planner, which avoids collisions and obey dynamics. This approach showed high efficiency in complex problems, relying on cooperation to find routes, and suggested that incorporating high-level planning will enable a team of robots to better perform in complex tasks, as suggested by our work.

Attempts to handle the complexity arising from planning paths for high-dimensional multi-robotic systems with non-linear dynamics and collision avoidance include planning in both discrete and continuous spaces is described in [10]. Sislak et. al [11] described a distributed flexible multi-layered de-confliction architecture, where collisions are avoided by an asset-to-asset negotiation and single A* progressive path planning. This approach allows to handle higher number of aircraft, decreased requirements for human operators and allows participation of non-cooperative agents.

Multi-Agent Path Finding (MAPF) algorithms are used where multiple agents need to plan a path simultaneously while assuring collision avoidance. The goal in these algorithms is to minimize the sum of all travel times, or minimize the maximal travel time of the agents. Solving MAPF optimally is computationally hard, and many different algorithms have been developed for this purpose over the years [12]. Solutions to the MAPF problems can be either centralized [2], distributed, or a combination of both [13]. However, solutions are usually restricted to situations in which all robots initiate their navigation simultaneously, and in discrete spaces (grid or graph environment). Both assumptions, as well as the high computation time, make it impractical to use this model and solutions in our domain.

A recent study by Ho et. al [14] examines a decentralized approach for aerial traffic management system, in which the UAVs are responsible for handling collision-avoidance using negotiation. The study shows that using this method lead to a significant reduction in costs for the aircrafts in terms of delays and rejected operations due to re-planning.

Our suggested HPP framework is designed to incorporate any distributed path planning method in the local zone, and cooperative path planning method in the global zone.

B. Current UTM Practice

UTM theoretical and practical research is lead by institutes such as NASA and U-SPACE CESAR, while experts from industry and academic fields work closely with and federal aviation administrations (such as FAA and EASA) towards the goal of integration of UAS into the national air space. NASA's UAS Traffic management (UTM) research initiative has demonstrated the flight of multiple UAVs flying in urban environments [15]. Most of the deconfliction has been demonstrated with **separate operational volumes** through different UAS service suppliers (USS). A UAV talks with the local USS to get flight approvals from the UTM and make contingency plans [16]. In their work, Chakrabarty et al. offer a complete autonomous architecture for flying a UAV in dense urban environments. The planning architecture consists of a global planner which plans path to the goal using RRT and local planner is involved in avoiding obstacles according to SAFE50 autonomy architecture [17], unlike the HPP local planner which is also involved in path planning.

One way to handle the complexity of the problem is to divide the airspace to multiple layers and apply different rules and algorithms for each one. in [18] a multidimensional grid with fixed top-level paths and lower level paths was used for local traffic between the departure and arrival points and the top-level paths. Layers with fixed paths ("drone highways") manage traffic implementing scheduling algorithms, while the rest of the layers used for traffic from departure points to the highways and from the highways to the arrival point, implement routing algorithms to handle traffic. In [15] Amazon proposed a concept of airspace design and management where the airspace is divided into three vertical layers with lateral "predefined low risk location": "low-speed localized traffic", "high-speed transit", and "no-fly" zone.

On December 28th 2020, The US FAA issued a final ruling on Remote ID [19] , forcing all drones to either fly in a predefined area (FRIA), or install a broadcast device publishing drone and operator real time information .

III. The Hybrid Path Planner Architecture

We developed the hybrid path planning (HPP) approach to cope with the complex environment challenges of UAV flights in urban areas. Obstacles and densely-populated areas on ground and strictly regulated traffic in predetermined routes in air, make high volume traffic in this environment a challenging task.

Looking at traditional air traffic regulation we find two principle approaches to control air traffic:

- Flying in predetermined routes regulated by a central controller, used to handle high volumes of traffic.
- Flying in predetermined volumes, self controlled (yet still subject to flight rules) in more scarce areas.

The HPP approach divides the flight path of the UAV in a similar manner into two differently regulated zones: the

global and *local* zones (see illustration in Figure 1). Global zones (SkyWay) include predetermined centrally controlled SkyWays, while the local zones define a self controlled confined volume.

The core of the HPP is a server utility that accepts path requests from UAVs, and assigns each one with optimized flight path. The HPP determines a UAV's **entry** point into the global zone and an **exit** point from it, and may take into account the expected load, no-fly zones and other global considerations when computing a path. If the HPP was required to accurately assess the travel time for each UAV from its start to goal locations, it would have needed to compute a full path (using RRT*, for example). However, computing the exact path for each UAV is impractical in terms of computation resources. Therefore, local zone navigation will not be calculated, but will be approximated by an estimation algorithm. We will review three methods for determining the entry and exit points of the UAV, as described section IV. We have implemented and evaluated all three methods to examine the differences between them in terms of calculation time, estimated UAV travel distance, and the difference between it and the actual UAV path length calculated by the HPP simulator, as described in section VI

Fig. 1. The global zone (SkyWay) and local zones for the HPP architecture in the 3D simulation, and in the correlated 2D simulation (top left corner)

A. Local Zone

The local zones is typically a geographical volume characterized by densely populated environment with variety of restrictions, for example a neighbourhood, city quarter, educational institutes, or industrial zone. This area is dynamic and changing, requiring slow careful navigation with highly efficient obstacle avoidance capabilities. Start and goal points will always be in the local zones, and transition to and from the global zone will be done via entry and exit points defined by the HPP Server. Since the local zone is nonholonomic space, the HPP approach suggests local zone path planning will be performed locally by the UAV using RRT* or other similar algorithm from the start to the entry point, and from the exit to the goal point.

B. Global Zone (SkyWay)

The global zone is typically a geographical volume characterized as open lower density space which supports higher speed UAV traffic. The global zone is allocated in areas like inter-cities with reduced risk and noise to the dense population, and will usually be restricted to routes between predefined fixed nodes, dictated by the regional transportation authority. Since the global zone space can be modeled as a graph, the HPP approach suggests global zone path planning will be performed centrally by UTM authority, using A* or other similar algorithm from the entry to the exit point.

C. Problem Definition

In this paper we examine different methods for hybrid path planning optimization. The global zone will be modeled as a 2D weighted undirected graph $G = (V, E, L)$ with nested local zones, such that each vertex $v_i \in V = (v_i^x, v_i^y)$ is an intersection in the SkyWay, an edge $e = (v_i, v_j) \in E$ connects two vertices $v_i, v_j \in V$ in a direct line in the SkyWay, and the weight of an edge $e_i \in E$, denoted by $e_i^l \in L$, represents load statistical data as a function of time t[1]. We assume that local zone navigation does not cross the SkyWay.

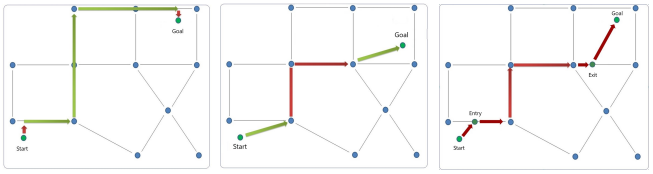
D. Path Planning Process

The Path Planning Process (flight plan) begins with a UAV requesting a flight path.Formally, a UAV i that needs to travel issues a *UAV Path Request* $R_i = < r_i^s,\ r_i^g,\ r_i^t >$ such that r_i^s is the requested start point r_i^g is the requested goal and r_i^t is the requested entry time to the global zone. The HPP server calculates a *UAV Path* $u_i = < u_i^s,\ u_i^e,\ u^{el},\ u_i^x,\ u_i^g,\ u^t >$ such that u_i^s is the path start point, u_i^e is the path entry point to the global zone, ${u^{el}}_i = \{e_{i1} \dots e_{ik}\}$ is a list of edges on G (the SkyWay), u_i^x is the path exit point, u_i^g is the path goal point and u_i^t is the path entry time. Once receiving the plan, the UAV takes off and navigates to the entry point (using any onboard navigation algorithm, in our case we have implemented RRT*). In this paper we assume that the UAV is avoiding obstacles and other UAVs autonomously. The UAV joins the SkyWay at the assigned time and location and navigates through the global zone at the assigned speed to the exit point. The UAV then enters the goal's local zone and navigates to its goal.

IV. HPP Methods

We will examine three possible global path planning methods that define for each UAV i the entry point, exit point and path in the global zone (SkyWay): MLT, MGT and OPT (defined below). We will test the HPP algorithms' performance, and compare HPP performance to an optimal solution which computes RRT* for each UAV as a benchmark. The evaluation will focus on the trade-off between 3 parameters: the actual travel time and distance compared to the estimated one, and the computation time. All three algorithms do not compute the actual path for the a UAV, but use other straight line calculations as estimations (MLT and MGT use simple straight line estimation, and OPT uses a weighted estimation, discussed in detail in section V).

[1]Since e_i^l is time dependent, its formal definition is $e_i^l(t)$, though we omit the time dependency wherever possible



(a) MLT (b) MGT (c) OPT

Fig. 2. Illustration of the three methods in HPP

A. Minimum Local Time (MLT)

Minimal local time method determines a path for each UAV i that minimizes its time in the *local* zone. The HPP will receive the UAV request and choose an entry point P_i^e on the SkyWay which is closest to the start point, and exit point P_i^x on the SkyWay which is closest to the goal point, and determine the SkyWay path with minimal cost between these two points. See illustration in Figure 2 (a).

B. Minimum Global Time (MGT)

Minimal global time method determines a path for each UAV i that minimizes its time in the *global* zone. The HPP will receive the UAV request and choose an entry point P_i^e and exit point P_i^x, such that the distance between P_i^e and P_i^x will be minimal. See illustration in Figure 2(b).

C. Approximated Optimal Time (OPT)

The approximated optimal time method (optimal time, in short) calculates a path for each UAV i that strives to minimize total travel time, both in global and local zones, considering obstacle density in local zones (affects RRT* path length) and SkyWay speed. See illustration in Figure 2(c). The calculation of the approximated optimal route is described in depth in the following section.

V. OPT Algorithm

Recall that the HPP does not compute the exact path in the local zones, but uses a straight-line approximation of path length. However, the actual path length can be quite different than the straight-line approximation, especially in very dense environments. In order to estimate the impact of the density of the obstacles in the zones on the length of the RRT* path, we define the notion of *RRT-Factor*, $\texttt{RRTF}(o)$ (denoted by RRTF, or F, in short), which indicates the length of the actual RRT*-based path compared to a straight line in an area with obstacle density o. $\texttt{RRTF}(o)$ is determined empirically, as shown in the evaluation section. We describe in detail the selection of the entry point to the SkyWay. The selection of the *exit* point is similar.

OPT algorithm works as follows. For a given start point, it examines for every edge e_j bounding the start zone, a point $P_i^e(v_j)$ along that edge that minimizes the travel time towards a vertex of the SkyWay, denoted as SFN (SkyWay's First Node, the node closest to the entry point along its way to the goal point). Then it calculates all possible SLN (SkyWay's Last Node) points bounding the goal's local zone, and determines the pair of entry and exit points minimizing the total travel time. The OPT algorithm considers several parameters including obstacles level in the local zone and travel speed in the SkyWay (denoted by Speed Factor). See illustration in figure 3 for a single SFN.

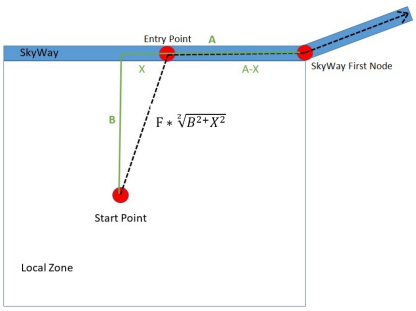


Fig. 3. Entry Point optimization

A. Entry Point Selection

In order to find the optimal entry point, we need to find the X value shown in figure 3 that will minimize the distance to the SFN. Specifically, we would like to minimize the sum of two values: the estimated travel time between the start point to entry point, $(X^2 + B^2)^{1/2} * \texttt{F}$, and from the entry point to the SFN: $(A - X)$. We take the distance equation first derivative and compare to zero, resulting in $X = \sqrt{B/(\texttt{F}^2 - 1)}$. Since B (distance of start point to the edge) and F (RRT* overhead factor) are known, the calculation of the X value yielding minimal travel time is straightforward.

B. SkyWay Speed Factor

A key factor in the optimal entry point calculation is the the UAV different travel speed in the local and the global zone. Because of this fact, there could be a bias to prefer global zone travel, which decreases x as can be seen in figure 3 (brings the entry point closer to the start point). We denote the increase (or decrease) in the speed on traveling in the SkyWay compared to the local zone by SF (Speed Factor). Therefore the optimization equation will then be updated as follows: $Dist = \texttt{F} * \sqrt{X^2 + B^2} + (A - X) * SF$. By deriving this equation we can again deduce the optimal X value: $\frac{B*SF}{\sqrt{\texttt{F}^2 - SF}}$

C. Time Complexity

The OPT algorithm within the HPP Planner wishes to find a close to optimal entry point to the SkyWay for each UAV. The number of edges bounding a local zone is bounded by constant value c, so the calculation of the optimal entry-exit is done in $O(c^2) = O(1)$, in addition to the calculation of the path along the SkyWay. However, in our HPP implementation we perform a pre-process phase to calculate distances between all vertices of the SkyWay surrounding the start zone, to all vertices of the SkyWay surrounding the goal zone (and between all zones, in general), using A*. This process is done once for all UAVs in offline and results are kept in an arraylist. Thus using the arraylist to select the SkyWay's first and last node is done in $\mathcal{O}(n)$.

VI. HPP Simulator

In order to support multi-cycle runs of HPP planning, we developed the HPP Simulator (HPPS) (see Figure 4).

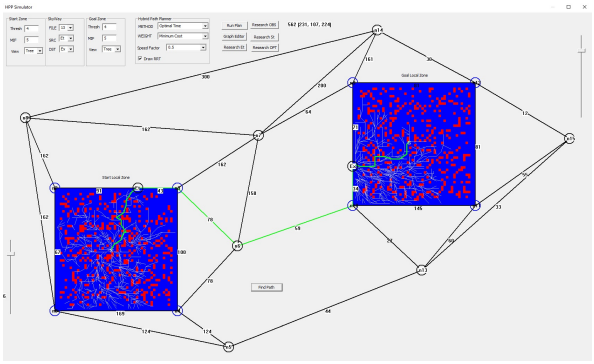


Fig. 4. HPP Simulator

A. Configuration Space

Local Zone

The local zone is represented as a 2D 50X50 units grid . Each unit can be defined as free or occupied by an obstacle. A UAV can travel in the local zone in all 8 surrounding cells: up/down/left/right and diagonally. In the local zone our selected path planning algorithm is RRT* since it is suitable for nonholonomic space with obstacles [20].

Global Zone

The global zone (SkyWay) is represented as an undirected weighted graph with nodes and edges, where each edge has a weight parameter that can either represent the distance between nodes, or other constraints along the edge (e.g., heavy traffic or bad weather). The Graph information (e.g nodes, edges, weights) is stored in a configuration file, and can be automatically replaced between runs.

B. Path Options

The HPPS supports several configurations of path planning in the local and global zones. Specifically, we have implemented $A*$ for path planning in the global zone, and in the local zone the path is calculated in one of the three options mentioned above (MLT, MGT, and OPT). We have also implemented RRT* for benchmark, although the actual path planning does not run RRT*.

C. Connectivity

The HPP can verify, as a pre-process step, that a valid path exists between the start point and the entry point (that is, they belong to the same connected component). First the algorithm verifies the points are not located in an obstacle, then it verifies that a path exists between the points using the flood fill algorithm. A flood fill operation begins at the start point and colours in blue all the grid points connected to the start point. The algorithm then verifies that the entry point is in a blue grid slot. If not, the HPPS can pop an alert or automatically adjust the entry point to a near blue grid slot, as demonstrated in Figure 5.

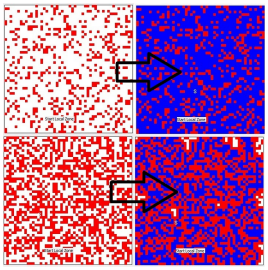


Fig. 5. Two examples of the Flood Fill Test

D. RRT Parameters

RRT* algorithm has a few parameters to control its behaviour allowing to customize the algorithm to the implementation case. The parameters used in the HPPS are:

Threshold The Threshold parameter is the maximum length of each tree branch. Higher threshold allows quicker tree expansion and faster time to reach the goal. Lower threshold allows better agility and path smoothing between obstacles. The value that seems to balance time calculation performance vs. path optimization was 4 grid units.

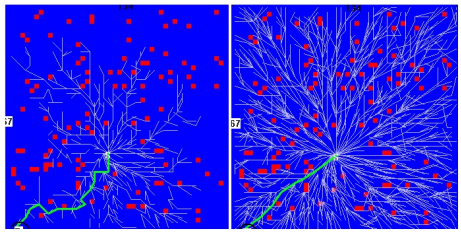


Fig. 6. Different Threshold Values (3 at the left, 5 at the right)

Maximum Iteration factor (MIF) In the RRT* algorithm each iteration tries to produces a branch. Theoretically iterations can continue to infinity optimizing the tree. The HPPS allows 2500 iterations (n^2) if goal is found, and if goal not found, continue up to 2500 * MIF times. The MIF value used in the HPPS is 5.

E. 3D Module

The HPP Simulator supports 3D representation using Unity 3D Engine. the HPP Simulator outputs the following configuration files to a shared folder:

- HppSkyWay - contains SkyWay's edges data (start point and end point)
- HppObs - contains obstacle data (x, y , height)
- HppDronePath# - contains the path data for a specific (#) drone (waypointx, y, speed)

The 3D Engine then initialize a predefined scene, loads SkyWay data and obstacle data, presents them, loads the drone paths, instantiates a drone for each path and simulates its takeoff, flight and landing.

VII. Empirical Evaluation

A. Calculation Time of RRT* vs. OPT

Multiple UAV operations require simultaneous path planning calculations for many platforms. RRT* algorithm is

highly effective in nonholonomic path finding but is also time consuming. As can be seen in Figure 7 calculation time starts from 1-2 seconds, and increases with obstacle level to more than 10 seconds per path (Intel i7 CPU) . These results were achieved when restricting number of RRT iterations, allowing more calculation time will increase results optimization. In Figure 7 we can see the results of 1100 iterations - mean time of 20 RRT* iterations for each obstacle levels 1-55%. We can see the trend of higher computation time as obstacle level increases. The large variation in the results is the outcome of the random nature of RRT*, causing some paths to be found quickly while others take more time.

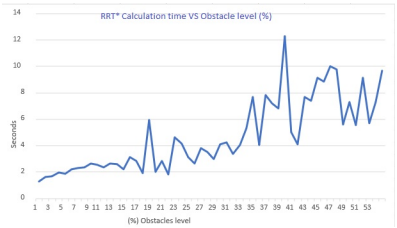


Fig. 7. RRT* Calculation time vs. Obstacle Level

OPT Algorithm's calculation time, on the other hand, is very fast and takes in average less than 1 m/sec per path.

B. MLT vs MGT

Our initial interest was to compare the performance of MGT with MLT. We ran this test with Speed Factor = 1, meaning the same speed traveling the local and global zones. Our assumption was that reducing travel time in the local zone will yield a shorter path, because the RRT*'s 'Zig Zag' motion will add substantial travel time. Tests runs in the simulator revealed otherwise. The MGTwas shorter in average of 19% as seen in Table I below. Each row indicates the average path length of 25 iterations.

obstacle density (%)	MLT	MGT	Diff
0	481.88	384.44	20.22%
5	264.28	240.52	8.99%
10	566.68	448.56	20.84%
15	465.32	383.54	17.57%
20	100.8	72.8	27.78%
Average	375.79	305.97	19.08%

TABLE I
MLT VS. MGT PATH LENGTH

In order to verify results integrity, we ran 1000 iterations of RRT* path calculations, each 40 runs with the exact same RRT* parameters, and then repeated 25 times with different obstacle level. We calculated the standard deviation of the cost results. The coefficient of variation (CV=standard deviation/mean) was very low and always under 1 which indicates the algorithm is robust.

C. OPT vs MLT vs MGT

In this test we ran 500 iterations comparing OPT, MLT and MGT. Every 100 iterations we changed the Speed Factor (SF), in order to see what influence (if any) it has on the different algorithms' performance. In Figure 8 below we can clearly see that the OPT algorithm performed significantly better in all SF levels 1-5 (SF 5 for example means global zone travel speed is 5 times faster than local zone travel speed). The average improvement from MGT was 22.2% and the average improvement from MLT was 9.74%. We can also see that as SF increases, the MLT becomes better than the MGT, due to the higher traveling speed in the SkyWay.

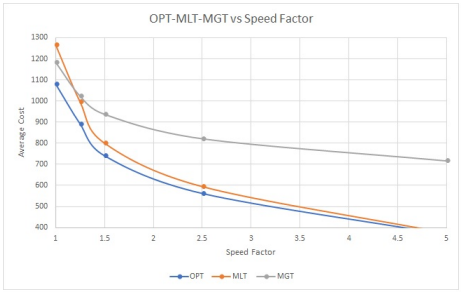


Fig. 8. OPT vs MLT vs MGT

D. Calculating the RRT Factor (RRTF)

RRT* by definition of its branching implementation, is always longer than a direct start-entry line. As discussed, we have defined a parameter called RRT* Factor (RRTF), which measures the relation between the RRT* distance, and direct distance between start and entry points (or exit and goal points) in the local zone. We further verified the logical assumption that higher obstacle level will increase the RRTF. Using the HPP Simulator we ran 10,000 tests recording the relation between RRT* path length and direct path length and how this relation is changing when applying different obstacle levels. We have run 1000 cycles of RRT* path planning in a given local zone. Each experiment was executed 20 times with the same obstacle level (from 1% to 50%), where obstacle locations were drawn at random. For each line the average RRTF was calculated. As seen in figure 9, there is a close to linear relation between the RRT* Factor and obstacle density. Using linear regression we found a factor that can help us estimate the distance to be traveled in the local zone, given a known obstacle level. The 'magic number' for RRT* distance prediction for a given obstacle level was found to be $\texttt{RRTF}(o) = 1.340984394 \times o$.

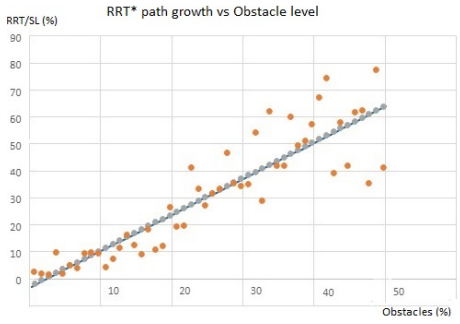


Fig. 9. RRTF prediction by obstacle level (o) using Linear Regression

In order to verify RRTF's integrity we have ran a series of 600 cycles, measuring the difference between the cost predicted by the HPP and then ran RRT* to get a real result. Every 20 cycles we changed the start point and obstacle level to measure RRTF accuracy in variable conditions. We recorded the difference in each cycle between the actual cost measured by the HPP Simulator, and the RRTF Predictor. Results show that the RRTF's accuracy is not affected by the obstacle level. The average error is 1.32% which is very low, though standard deviation reaches 13.4% which can be explained by RRT*'s random nature.

E. SkyWay Speed Factor (SF)

We have measured the cost reduction reaching the skyway's first node as seen in figure 3, when using the HPP entry point optimization in different speed factors. As expected - as speed factor increases, travel time decreases. In order to further verify that the speed factor does not bias entry point optimization, we ran 300 tests with constant speed factor 0.5, changing start point and obstacle level every 20 cycles. For each batch we measured the average cost difference between OPT and MLT. In figure 10 below we can see that for almost all obstacle levels the average cost difference was positive meaning MLT average cost was higher than OPT average cost. Average difference for all obstacle levels was 24.93 which was 10.2% from the average path cost.

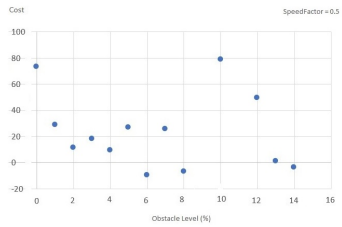


Fig. 10. OPT - MLT cost difference by Obstacle density

VIII. Conclusions and Future Work

We have introduced an innovative hybrid approach for path planning in the UTM space, combining distributed and central planning. The hybrid approach is based on dividing the air space into two zones: global and local zone, and finding the best way for a UAV to join the global zone and leave it. The architecture is based on using a path planning predictive model for an individual UAV, without the need to actually compute a path (using RRT*) for each request, thus allowing the system to handle many requests in parallel. We have empirically analyzed the performance of this optimal estimator algorithm against two other algorithms for determining the entry and exit points to the global zone, and have shown that it significantly outperforms them, and provides an accurate estimate to the actual path traveled by the UAV. During the research we have encountered several aspects required in real world situations, that seems highly applicable for future study, among those possible use of hierarchical representation within the local zones that takes into account such heterogeneous environments, extending the work to 3D environments, and accounting for dynamic and probabilistic obstacles.

References

[1] Ayalon-HighWays. Request for information (rfi) and request for demonstration (rfd) subject: Urban mobility in the aerial dimension, February 2020.

[2] Ofra Amir, Guni Sharon, and Roni Stern. Multi-agent pathfinding as a combinatorial auction. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[3] John H Reif. Complexity of the mover's problem and generalizations. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 421–427. IEEE, 1979.

[4] Jacob T Schwartz and Micha Sharir. On the "piano movers" problem. ii. general techniques for computing topological properties of real algebraic manifolds. *Advances in applied Mathematics*, 4(3):298–351, 1983.

[5] Steven M LaValle, James J Kuffner, BR Donald, et al. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, (5):293–308, 2001.

[6] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2), 2010.

[7] Chad Goerzen, Zhaodan Kong, and Bernard Mettler. A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65, 2010.

[8] Mohammadreza Radmanesh, Manish Kumar, Paul H Guentert, and Mohammad Sarim. Overview of path-planning and obstacle avoidance algorithms for uavs: A comparative study. *Unmanned systems*, 6(02):95–118, 2018.

[9] Duong Le and Erion Plaku. Cooperative multi-robot sampling-based motion planning with dynamics. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

[10] Erion Plaku. Planning in discrete and continuous spaces: From ltl tasks to robot motions. In *Conference Towards Autonomous Robotic Systems*, pages 331–342. Springer, 2012.

[11] David Šišlák, Michal Pěchouček, Přemysl Volf, Dušan Pavlíček, Jiří Samek, Vladimír Mařík, and Paul Losiewicz. Agentfly: Towards multi-agent technology in free flight air traffic control. In *Defence industry applications of autonomous agents and multi-agent systems*, pages 73–96. Springer, 2007.

[12] Roni Stern. Multi-agent path finding–an overview. In *Artificial Intelligence*, pages 96–115. Springer, 2019.

[13] Jirı Švancara and Roman Barták. Combining strengths of optimal multi-agent path finding algorithms. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 226–231, 2019.

[14] Florence Ho, Ruben Geraldes, Artur Gonçalves, Bastien Rigault, Benjamin Sportich, Daisuke Kubo, Marc Cavazza, and Helmut Prendinger. Decentralized multi-agent path finding for uav traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[15] Tao Jiang, Jared Geller, Daiheng Ni, and John Collura. Unmanned aircraft system traffic management: Concept of operation and system architecture. *International journal of transportation science and technology*, 5(3):123–135, 2016.

[16] Anjan Chakrabarty and Corey A Ippolito. Autonomous flight for multi-copters flying in utm-tcl4+ sharing common airspace. In *AIAA Scitech 2020 Forum*, page 0881, 2020.

[17] Corey A Ippolito, Kalmanje Krishnakumar, Vahram Stepanyan, Anjan Chakrabarty, and Josh Baculi. Safe50 reference design study for large-scale high-density low-altitude uas operations in urban areas. In *2019 AIAA Modeling and Simulation Technologies Conference*, 2019.

[18] Lasse Berntzen, Adrian Florea, Cristian Molder, and Noureddine Bouhmala. A strategy for drone traffic planning dynamic flight-paths for drones in smart cities. In *SMART 2019, The Eighth International Conference on Smart Cities, Systems, Devices and Technologies*, 2019.

[19] FAA. Faa's remote id final ruling, December 2020.

[20] R. Pepy and A. Lambert. Safe path planning in an uncertain-configuration space using rrt. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5376–5381, 2006.