2021 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. June 15-18, 2021

UAS Flight Path Planning using Numerical Potential Fields in Dense Non-segregated Airspace

Sajid Ahamed M A*, Satya Prakash K*, Shuvrangshu Jana[1] and Debasish Ghose[2]

***Abstract*— This paper addresses the problem of Urban aerial mobility (UAM), which refers to an air transportation system wherein unmanned aircraft systems (UAS's) are utilized to deliver small payloads on-demand by safely maneuvering in the known urban infrastructure occupied by dynamic air traffic. A multilevel framework is proposed that provides a solution for UAM in the non-segregated airspace. The proposed method takes advantage of a prediction-based zone partitioning algorithm that minimizes high-density traffic encounters. Numerical potential fields are used for global path planning as they provide optimum standoff clearance from obstacles and high-density traffic present in the environment. In the event of low-density traffic, a simple but effective octant-based local collision avoidance strategy is used.**

I. INTRODUCTION

Unmanned aircraft systems (UAS) are increasingly being used for the transport of small payloads over short distances. UAS deployed in an urban scenario for such tasks must be capable of navigating through densely populated urban airspace, avoid potential collisions, and deliver payload at the site on-demand. Implementations and safety concerns associated with such traffic call for a controlled air transportation network facilitating urban aerial mobility (UAM). The concept of operations for such a network was proposed by NASA under its ambitious research initiative, UTM (Unmanned aircraft system traffic management) [1], wherein the focus was on enabling large scale visual line of sight (VLOS) and autonomy beyond the visual line of sight (BVLOS) UAS operations in low altitude urban airspace. With the advancement of UTM technical capability level to TCL-4 [2], addressing BVLOS safe flying in low altitude (below 400 ft) highly populated urban environments, many governments and private entities have come forward with different modular architectures having core functionalities of collision detection and avoidance, geofence based conflict detection and resolution, and global path planning [3]–[7].

While [2]–[5], [7] propose a multilayer air traffic management model that restricts UAS traffic to segregated airspace associated with pre-planned routes of specified volume, generally devoid of obstacles, and allocated for exclusive use to a specific user by imposing geofence constraints; the present paper focuses on UAS path planning in non-segregated airspace, which is dense in both obstacles and air traffic for applications such as door-to-door delivery. Also, air space can be utilized better with traffic planning in non-segregated airspace.

Path planning in non-segregated airspace is reported in the literature using methods such as $A*$ search [8], Rapidly-exploring random trees (RRT) [9], Probabilistic road maps methods with ant colony optimisation [10], combination of $A*$ and artificial potential field [11].

In [8], a risk cost matrix considering various factors such as weather, operational constraints, network cost is accounted for in developing path planning algorithm using A* method; however, the formulation doesnot handle dynamic obstacles. In [9], sampling-based chance constrained RRT algorithm is used to obtain probabilistic guaranteed feasible path in the presence of uncertain obstacles. Sampling-based probabilistic algorithm will have less computational load; however, considering the safety requirement in UAM application, the trajectories should be safe in all cases. Path is chosen probabilistically using a centralized task planner in [10]. The problem with static algorithms like A* or its variations is that they need to replan their path from scratch if there is a change in environment, which could be infeasible in airspace with multiple dynamic objects [11]. To improve the limitations of the static path planning algorithms, path planning is made in a multilevel architecture to handle the dynamic environment. In [11], path planning is performed initially with A* algorithm, and local collision avoidance is handled using an artificial potential function. Path planning algorithms based on artificial potential methods are not suitable for planning globally due to the presence of local minima [12]. In order to solve the issues of local minima, different variations of modeling the potential force profile are reported in the literature, such as harmonic potential functions [13]–[16]. In [16], global path planning is formulated using harmonic functions, Kalman filter, and Markov Decision Process (MDP) for handling static obstacle avoidance, high traffic density region prediction, and dynamic obstacles in vicinity. Obstcale avoidance methods based on the analytical potential function donot ensure a optimum clearance from static obstacles. In the case of UTM applications, the map of the environment will be known to UAVs. Thus, instead of harmonic potential fields, using numerical potential fields concepts, we can ensure UAS path to have optimum clearance from static

* Equal contribution
Sajid Ahamed M A and Satya Prakash K are Research scholars in Department of Aerospace Engineering, Indian Institute of Science, Bengaluru, India. Email: sajidahamed@iisc.ac.in , satyak@iisc.ac.in
[1]Shuvrangshu Jana is a Postdoctoral Research scholar in Guidance, Control and Decisions Systems Laboratory (GCDSL) at the Department of Aerospace Engineering, Indian Institute of Science, Bengaluru, India. Email: shuvrangshuj@iisc.ac.in
[2]D. Ghose is a Professor in GCDSL at the Department of Aerospace Engineering, Indian Institute of Science, Bengaluru, India. Email: dghose@iisc.ac.in

2021 International Conference on Unmanned Aircraft Systems (ICUAS) | 978-1-6654-1535-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICUAS51884.2021.9476791

obstacles. Numerical potential field techniques are capable of handling local minima and are applicable to robots having multiple degrees of freedom [17].

In this paper, path planning architecture for UTM applications is proposed using the concept of numerical potential field, dynamic zone partitioning, and octant determination-based local collision avoidance strategy. The path is generated to maintain optimum clearance from the static obstacle using numerical potential field, and dynamic obstacles are handled using dynamic zone partitioning of the environment and an octant-based collision avoidance strategy. On the predicted traffic density, the paper proposes a formalized binary sequence algorithm for dynamic zone partitioning of the environment. There is significant literature on collision avoidance with dynamic obstacles [16], [18]–[20]. Collision avoidance strategy is developed using the position and velocity information of dynamic obstacles [18], software defined networking (SDN) system [19], visual predictive control scheme [20], Markov Decision Process using visual information [16]. Clearly, existing strategies are based on complex network or algorithms which requires large sensor information and computational load. Our approach is based on position information of dynamic obstacles. It is simple, and the computational load requirement is less compared to the above strategies.

II. Problem Definition

In this paper, we address the problem of urban air mobility (UAM) application, wherein several UAS coexist, and each UAS has to maneuver in a dynamically changing environment, reach a destination, duly avoiding collision with static obstacles such as buildings, other dynamic elements such as other UAS present in the environment.

A. Workspace Definition

Depending on the allowable aerial mobility in the given environment and ground height restrictions being imposed, the workspace $\mathcal{W}$ is defined as the cuboidal enclosure of points $x \in \mathbb{R}^3$ in the serviceable jurisdiction as shown in Fig. 1. With $\mathcal{W}$ being a bounded set in $\mathbb{R}^3$, by dividing each axis of the bounded set into N segments of length δ_l, $l = 1, 2, 3$ along respective axes, $\mathcal{W}$ is decomposed into finite set of cells. Further, by mapping each cell to an element in a 3 dimensional matrix $\underline{\mathcal{W}}$ indexed by the tuple $n \in \mathfrak{W} = \{(n_1, n_2, n_3) \mid n_1, n_2, n_3 \in [1, N] \cap \mathbb{Z}\}$; $\mathcal{W}$ is modeled as a 3D bitmap array. The reason for referring to it as a bitmap is that each element in $\{\varepsilon_n \mid \varepsilon_n \in \underline{\mathcal{W}}, \quad n \in \mathfrak{W}\}$ is either 1 or 0; where 1 represents cells occupied by static obstacles or intersected by $\mathcal{W}$ boundary and 0 represents those which are empty, resembling a $3D$ bitmap image.

We formally define a cell and its neighbourhood as follows. Say w_n is the cell mapped to element $\varepsilon_n \in \underline{\mathcal{W}}$ indexed by $n = (n_1, n_2, n_3) \in \mathfrak{W}$. Then, this cell w_n is defined as the set of points $x \in \mathcal{W}$ whose coordinates $x_d \in [\,(n_d - 1)\delta_d,\ n_d\delta_d\,]$, $d = 1, 2, 3$. The d-neighbourhood of this cell $\mathcal{N}_d(w_n)$, $d = 1, 2, 3$ is defined as the subset of cells in $\{w_{n'} \mid n' \in \mathfrak{W}\}$ whose tuples differ from n by 1 in at-most d coordinates. For example, 1-neighbourhood of cell has six neighbours $w_{n'} \in \mathcal{N}_1(w_n)$ whose tuples n' differ from n by 1 in at-most 1 coordinate, given as $n' \in \{(n_1 - 1, n_2, n_3), (n_1 + 1, n_2, n_3), (n_1, n_2 - 1, n_3), (n_1, n_2 + 1, n_3), (n_1, n_2, n_3 - 1), (n_1, n_2, n_3 + 1)\}$. Similarly, $\mathcal{N}_2$ has eighteen and $\mathcal{N}_3$ has twenty-six neighbours. Based on each neighbourhood definition, we form connected graphs ${}^d\mathcal{GW}$ with tuples n as nodes, that would serve as grid map facilitating cell to cell transition in $\{w_n \mid n \in \mathfrak{W}\}$.

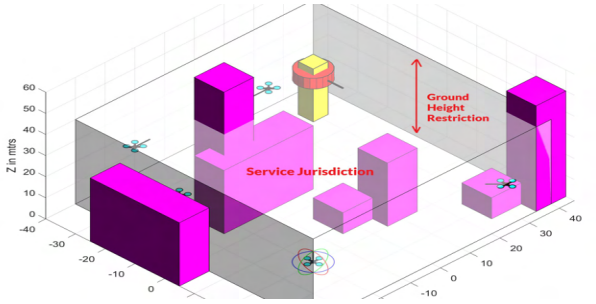
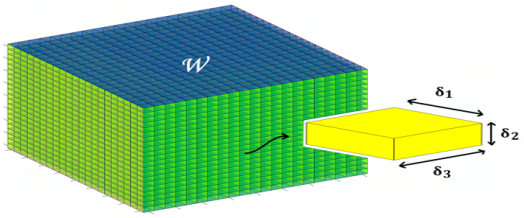


Fig. 1: Workspace environment for UAM



(a) Cell decomposition of workspace $\mathcal{W}$.

(b) 1-Neighborhood

(c) 2-Neighborhood

Fig. 2: Cell and its d-neighborhood definition.

The UAS, for which path planning from start to a goal point in the workspace is being undertaken, shall be continued to be referred to as UAS. However, from the viewpoint of this UAS, all other UAS will be referred to as known intruders (for which it is assumed that their underlying dynamics are known). All other dynamic elements in the environment are referred to as unknown intruders. Though 6-DOF quadrotor dynamics have been considered for UAS, it is assumed that the UAS would never encounter scenarios such as passage through narrow gaps, wherein it would be forced to perform roll, pitch, and yaw constrained maneuvers for avoiding a collision. This assumption allows the UAS to be treated as a 3DOF spherical robot (smallest sphere enclosing the quadrotor), with configuration space $\mathcal{C}$ being three dimensional ($\mathbb{R}^3$).

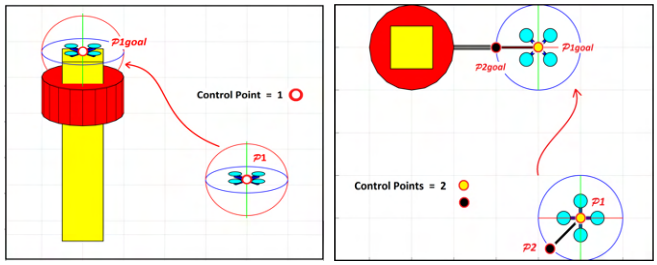
For a given configuration $q \in \mathcal{C}$, $\mathcal{A}(q) \subset \mathcal{W}$ denotes the set of points occupied by the UAS in $\mathcal{W}$ at that configuration.

The position coordinates ($x \in \mathcal{W}$) of each of these points $p \in \mathcal{A}(q)$ and the cell w_n that encloses this position can be determined by the following geometric maps, known as *forward kinematic map* X and *forward quantization map* Q.

$$X : \mathcal{A} \times \mathcal{C} \rightarrow \mathcal{W} \; ; \; (p, q) \mapsto x = X(p, q) \quad (1)$$

$$Q : A \times \mathcal{C} \rightarrow \mathfrak{W} \; ; \; (p, q) \mapsto n = Q(p, q) \quad (2)$$

Speaking of what could be a good choice for resolution of the cell w_n, lower the resolution, lower would be the computation and time complexity of the planning process. Keeping this in mind, while quantization, the cell with the largest dimension that encloses the spherical UAS but still retains a path in ${}^3\mathcal{GW}$ corresponding to the paths existing between static obstacles in $\mathcal{W}$ is considered. If such resolution is feasible, then it is sufficient to model $\mathcal{A}$ as point mass. In general, the spherical UAS is modeled as a finite set of selected points known as control points $p_i \in \mathcal{P} \subset \mathcal{A}, \; i = 1, 2, ..., s$, depending on the application (see Fig. 3). In this paper we address path planning in those applications where model can be restricted to one control point $p_i \in \mathcal{P}, \; i = 1$.



(a) (b)

Fig. 3: (a) UAS is modeled as one control point where yaw at goal is not important, (b) the UAS is desired to dock with a port mounted to a station, hence modeled as set of two control points, one at its center and other at the port terminal.

With respect to this control point, every cell in the empty part of the workspace is associated with a numerical potential V_{p_i} known as W-potential, that drives this control point to its respective desired goal cell w_{g_i}.

$$V_{p_i} \; : \; \{ \; w_n \mid n \in \mathfrak{W} \text{ and } \varepsilon_n = 0 \; \} \; \mapsto \; \mathbb{Z}_{\geq 0} \quad (3)$$

$$w_{g_i} \; = \; \underset{w_n}{\text{argmin}} \; V_{p_i} \quad (4)$$

III. Methodology

The proposed path planner has the following multi-stage architecture for generating a path from a start to a goal configuration in real-time with sub-modules 1) zone determination based on traffic density 2) local collision avoidance of dynamic obstacles 3) global path planning based on numerical potential field techniques. The local collision avoidance strategy is based on the sectorization of the neighborhood around UAS, followed by the identification of octants occupied by threats (known intruders or unknown intruders or both). The flow diagram for the proposed method is given in Fig. 4.

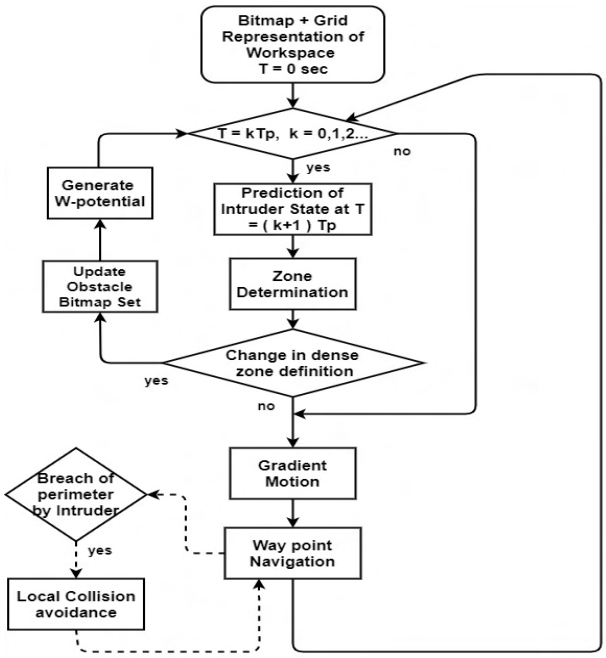


Fig. 4: Flow diagram for proposed path planner

A. Prediction and Zone Determination

This brings us to the first sub-module in the proposed planner, which refers to the 'Prediction of intruder state' and 'Zone determination' blocks of Fig 4. This sub-module is invoked every $t = kT_p$ instant, $\forall \; k \in \mathbb{Z}_{\geq 0}$, where T_p is a user-defined time period. At this instant, the entire workspace is partitioned into $Z = 2^r$ zones, wherein a binary sequence of r bit-length can be used to keep track of zones and retrieve the boundaries of each zone simultaneously. Every successive bit in the binary sequence indicates a partitioning action to improve zone resolution about (X, Y) coordinate axes alternatively (see Fig. 5).

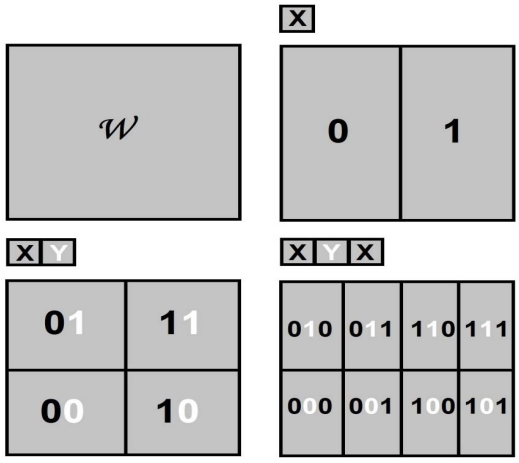


Fig. 5: Workspace partitioned into 2^1, 2^2, 2^3 zones. The 1^{st} and 3^{rd} bits correspond to partition action along X axis. The 2^{nd} bit corresponds to partition action along Y axis. Further inclusion of a 4^{th} at the LSB would mean partitioning $\mathcal{W}$ into 2^4 zones by partitioning along Y axis.

The intention is to predict zones in $\mathcal{W}$ that would be dense in known intruder traffic, T_p seconds ahead in time, and treat these zones as static obstacles until that future instant. By predicting the position of all known intruders T_p seconds ahead, we label each 2^r zones as a sparse zone, if atmost one intruder is in that zone and as a dense zone otherwise, as follows.

$$\mathcal{W} \;=\; \begin{cases} Zone_{sparse} & (D = 0, 1) \\ Zone_{dense} & (D > 1) \end{cases}$$

where, D is the predicted number of known intruders in the zone, at an instant T_p seconds ahead in the future. The information of present state $p_t = (x\ y\ z)^{\mathrm{T}}$, $v_t = (v_x\ v_y\ v_z)^{\mathrm{T}}$, acceleration a_t of all known intruders are assumed to be available to UAS using which their position and velocity could be computed T_p seconds ahead by propagating the linear discrete-time noise model given below.

$$\begin{bmatrix} p_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + \begin{bmatrix} \Delta t^2/2 \\ \Delta t \end{bmatrix} a_t \;+\; \nu_t \quad (5)$$

where, ν_t is noise in prediction, dependent on T_p. Thus, uncertainty in ν_t imposes restriction on users choice of T_p.

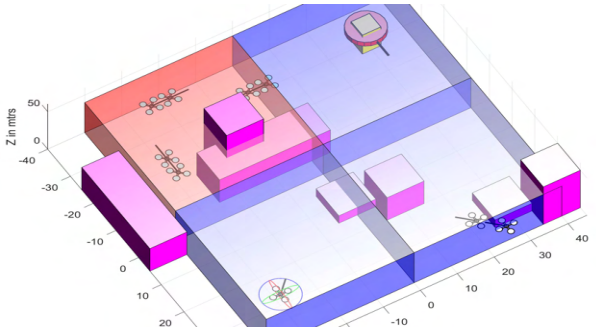


Fig. 6: Workspace partitioning and identification of zones as dense zone (depicted in red) and sparse zone (depicted in blue) based on predicted known intruder traffic density.

By planning a path for UAS avoiding the dense zones, we ensure that at any instant up to the next T_p seconds, the number of known intruders likely to be encountered by UAS in $\mathcal{W}$ is at most 1. Collision with this one leftover known intruder when encountered while traversing in the sparse zone can be handled using a local collision avoidance scheme.

The number of zones Z into which $\mathcal{W}$ is to be partitioned is dictated by the present position and the goal position of UAS, along with predicted positions of all known intruders. The zone determination algorithm continuously partitions $\mathcal{W}$ into $2^1, 2^2, ..., 2^{N_r}$ zones, simultaneously identifying them as dense and sparse zones depending on the predicted intruder traffic density until the conditions given below are satisfied.

1) The zone which encloses the present position of UAS must be a sparse zone.
2) If the distance between the present and goal position of UAS is within a certain threshold then the zone that encloses the goal position must be a sparse zone.
3) Both zones which enclose the present and goal positions of UAS must be connected via sparse zones.

The connectivity condition is to ensure that a path still exists between the present position and goal position though some portions of $\mathcal{W}$ are being removed and this can be verified by forming a grid map connecting midpoints of sparse zones and then performing a breadth-first search for the zone that encloses the goal position. If the above conditions are satisfied for some r bit length, then the corresponding bit sequences associated with each dense zone (except the zone that encloses the goal position) are stored in a list, which is referred to as *dense_zone_definition*.

At $t = 0$, entire $\mathcal{W}$ is considered as dense zone. Further, $\mathcal{W}$ is also modeled as $\underline{\mathcal{W}}$. The cells corresponding to $\{n \mid n \in \mathfrak{W},\ \varepsilon_n = 0\}$ are occupied by static obstacles or intersected by $\mathcal{W}$ boundary. These n are stored in an array, referred to as $\mathcal{BM}space$. Though zone determination submodule is invoked every $t = kT_p$ seconds, $k \in \mathbb{Z}_{\geq 0}$, the *dense_zone_definition* may or may not change (any change in elements of the list or bit length) depending on the predicted position of known intruders and above conditions related to UAS. But, whenever there is a change, the workspace potential generation sub-module is invoked, which refers to 'Update obstacle Bitmap set' and 'Generate W-potential' blocks of Fig 4. This is a computation heavy module, so having bigger zones as the outcome of zone partitioning is favorable as then change in *definition* every T_p seconds is less likely. It begins with computing half-plane boundaries of each dense zone in the *dense_zone_definition* using a binary search algorithm. Say $X_{min} \leq x \leq X_{max}$ and $Y_{min} \leq y \leq Y_{max}$ are half-plane boundaries of $\mathcal{W}$. For a dense zone, if the binary sequence was 01101, identify even position bits (X1X0X) and odd position bits (0Y1Y1) in the binary sequence. Evaluate the binary sequence following algorithm in Fig. 7.

```
Initialize x_min = X_min ; x_max = X_max ;
           y_min = Y_min ; y_max = Y_max ;
Set bit = MSB of the binary sequence;
while not all bits are evaluated do
    if odd position bit then
        if bit = 0 then x_max = (x_min + x_max)/2 ;
        else x_min = (x_min + x_max)/2 ;
        end
    else
        if bit = 0 then y_max = (y_min + y_max)/2 ;
        else y_min = (y_min + y_max)/2 ;
        end
    end
    half plane boundaries of dense zone →
    x_min ≤ x ≤ x_max ; y_min ≤ y ≤ y_max ;
    Set bit = next significant bit in sequence;
end
```

Fig. 7: Algorithm - Evaluation of Binary Sequence

Once the boundaries of the dense zones are obtained, they are treated as static obstacles. This update of the static obstacle set is achieved with the help of $\mathfrak{W}$. All such $n \in \mathfrak{W}$ corresponding to cells that are occupied by the dense zones along with $\mathcal{BM}space$ are collectively stored in an array referred to as $\mathfrak{W}_{occupied}$, with $\mathcal{GW}_{occupied}$ being the underlying connected grid map. The cells in the empty space of $\mathcal{W}$ can then be computed by the set difference of array $\mathfrak{W}$ and $\mathfrak{W}_{occupied}$, referred to as $\mathfrak{W}_{empty}$ with $\mathcal{GW}_{empty}$ being the underlying grid map. This sets the stage for the generation of W-Potential [17] in the empty space, which serves as the foundation for global path planning.

B. Workspace Potential Generation

Consider the L^1 distance between $n,\ n' \in \mathfrak{W}_{empty}$ corresponding to cell w_n and any cell in its 1-Neighbourhood $w_{n'} \in \mathcal{N}_1(w_n)$, to be 1. Starting at cell w_n, initialised with some distance value $V(w_n)$, the process of assigning all cells in $\{w_{n'} \mid n' \in \mathfrak{W}_{empty}\}$ a distance value $V(w_{n'}) = V(w_n) +$ L^1 distance between n' and n following ${}^1\mathcal{GW}_{empty}$ grid map is referred to as wave propagation with cell w_n as origin (refer algorithm in Fig. 8). During wave propagation, the list of cells $w_{n'}$ that have maximum distance, among the cells that had their distance value modified by the propagation is referred to as wave-front ($\mathcal{L}_0$ list in algorithm of Fig. 8).

```
Start at cell w_n, L_0 = {w_n} ;
Initialize V(w_n) = 0 ;
V(w_n') = γ some large value for n' ∈ 𝔚_empty/{n} ;
while L_0 ≠ φ do
    for every w_n' ∈ L_0 do
        L_1 = { w_n'' | w_n'' ∈ N(w_n') , V(w_n'') = γ } ;
        V(w_n'' ∈ L_1) = V(w_n' ∈ L_0) + 1 ;
    end
    Update L_0 = L_1 ;
end
```

Fig. 8: Algorithm - Wave propagation, origin at w_n

Each cell must be assigned a potential such that its negated gradient should pull the control point $\mathcal{P}$ to a cell corresponding to its goal configuration, also staying as far as possible from the static obstacles. This is achieved by first extracting a subset $\mathcal{S}$ in $\mathfrak{W}_{empty}$ such that $\{w_n \mid n \in \mathcal{S}\}$ serves a similar purpose as the generalised Voronoi diagram of $\{w_n \mid n \in \mathfrak{W}_{empty}\}$ for L^1 distance [17], and is referred to as workspace skeleton. By starting simultaneous wave propagations from each cell in $\{w_{n'} \mid w_{n'} \in \mathcal{N}_1(w_n)\ ,\ n \in \mathfrak{W}_{occupied}\ ,\ n' \in \mathfrak{W}_{empty}\}$ as their origin, we observe that wave-fronts of wave propagations which originated from different cells meet each other while propagating (refer algorithm in Fig. 9). The set of all such cells where the wave-fronts met is the W-Skeleton.

As every cell in the W-skeleton was a part of wave-front of some propagation at the instant they met, these cells have maximum distance from the cells $\{w_n \mid n \in \mathfrak{W}_{occupied}\}$ than any other cell in $\{w_n \mid n \in \mathfrak{W}_{empty}\}$. Thus, if $\mathcal{P}$ moves along $\mathcal{S}$ then it is guaranteed that it stays as far as possible from the static obstacles and $\mathcal{W}$ boundary, thus justifying the notion of optimum clearance from the static obstacles. Let $\{w_{g_i} \mid g_i \in \mathfrak{W}_{empty},\ i = 1\}$ denote the cell that encloses goal position associated with this control point $p_i \in \mathcal{P},\ i = 1$. This cell may or may not lie in $\{w_n \mid n \in \mathcal{S}\}$. If it does not lie then we need to explicitly connect this g_i to $\mathcal{S}$ and form an augmented skeleton.

```
Initialise Wave-front: L_0 = {w_n' | w_n' ∈ N_1(w_n) ,
                      n ∈ 𝔚_occupied , n' ∈ 𝔚_empty} ;
Origin of wave-front: O(w_n') = n', ∀ w_n' ∈ L_0 ;
Initialize V(w_n') = 0, ∀ w_n' ∈ L_0 ;
V(w_n'') = γ for {w_n'' | n'' ∈ 𝔚_empty}/L_0 ;
while for some n'' ∈ 𝔚_empty, V(w_n'') = γ do
    for every w_n' ∈ L_0 do
        for every w_n'' ∈ N_1(w_n'), n'' ∈ 𝔚_empty do
            if V(w_n'') = γ then
                V(w_n'') = V(w_n') + 1 ;
                O(w_n'') = O(w_n') ;
                Store w_n'' in L_1 ;
            else
                w_n'' ∈ wave-front of some propagation;
                if |O(w_n'') − O(w_n')| > 2 then
                    ⟹ wave-fronts of two different
                    wave propagation's have met ;
                    ⟹ Store n' in S ;
                end
            end
        end
    end
    Update Wave-front L_0 = L_1 ;
end
```

Fig. 9: Algorithm - Workspace Skeleton Extraction

If we observe the distance values associated with the cells in $\{w_n \mid n \in \mathfrak{W}_{empty}\}$ after $\mathcal{S}$ has been extracted, it starts with 0 at cells $\{w_{n'} \mid w_{n'} \in \mathcal{N}_1(w_n)\ ,\ n \in \mathfrak{W}_{occupied}\}$, monotonically increasing as we approach the skeleton. For connecting w_{g_i} to the skeleton we utilize this property. From w_{g_i}, we search for cells with largest distance value in its neighborhood and store it in a queue, further repeating the same for successive neighbours until a cell in $\{w_n \mid n \in \mathcal{S}\}$ is reached. By augmenting this queue with $\mathcal{S}$, we connect g_i with $\mathcal{S}$, thus connect goal cell to skeleton.

Each cell in $\{w_n \mid n \in \mathfrak{W}_{empty}\}$ must be assigned a W-potential value which is associated with this control point $p_i \in \mathcal{P}$. To achieve this, for the given control point p_i (respective $g_i \in \mathcal{S}$), w_{g_i} being the cell that encloses its goal position, we perform wave propagation with w_{g_i} as origin and $V_{p_i}(w_{g_i}) = 0$ as reference potential, proceed with assigning the L^1 distance as potential to only the cells in $\{w_n \mid n \in \mathcal{S}\}$. Once all cells in the skeleton have been assigned a potential value, we perform simultaneous wave propagations with each cell in $\{w_n \mid n \in \mathcal{S}\}$ as origin and $V_{p_i}(w_n)$ being the respective reference potentials. The

obtained W-Potential is

$$\{V_{p_i}(w_n) : n \in \mathfrak{W}_{empty}/\mathcal{S}\} > \{V_{p_i}(w_n) : n \in \mathcal{S}/\{g_i\}\} > \{V_{p_i}(w_n) : n \in \{g_i\}\} = 0 \quad (6)$$

By performing gradient motion on the above generated W-Potential (no local minima guaranteed since monotonic increase of potential from goal cell), the spherical UAS would eventually be driven towards goal position avoiding static obstacles and all known intruders in the dense zones of $\mathcal{W}$.

C. Gradient Motion and Random Motion

The nature of W-Potential is such that wherever the control point lies in the empty space $\{w_n \mid n \in \mathfrak{W}_{empty}\}$, it would first be driven towards Workspace skeleton $\{w_n \mid n \in \mathcal{S}\}$, away from the static obstacles, then it would slide along the skeleton eventually reaching its goal cell w_{g_i}. This is illustrated in Fig 10.

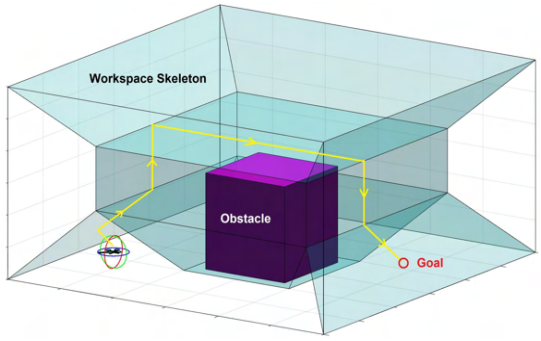


Fig. 10: Gradient motion in $\mathfrak{W}$, the control point $\mathcal{P}$ is attracted towards skeleton $\mathcal{S}$, then slides on the surface of $\mathcal{S}$ eventually reaching goal $\mathcal{G}$.

The W-Potential is assigned only in empty space; hence the control point $p_i \in \mathcal{P}$ is restricted to traverse in the empty space, along an underlying connected graph that connects the midpoint of cells according to ${}^3\mathcal{GW}_{empty}$ grid map (maximum of twenty-six such neighbors can exist).

Say at $t = kT_p$ seconds, the control point $\mathcal{P}$ is at midpoint of w_n. Of the 26 possible neighbours of w_n, $\mathcal{P}$ searches for the cell $w_{n'} \in \mathcal{N}_3(w_n)$ that has least potential and then pushes its midpoint into a queue. From this cell $w_{n'}$, it again searches for the cell that has the least potential in $\mathcal{N}_3(w_{n'})$ and pushes its midpoint in the queue. This subroutine of negated gradient search by $\mathcal{P}$ is referred to as 'gradient motion' (of Fig. 4) in respective spaces and is repeated until the desired number of cells to be traversed in T_p seconds (say 5) are stored in the queue.

Each midpoint stored in the queue is popped using a First In First Out (FIFO) strategy, each serving as a waypoint for generating minimum jerk trajectory [21] for time $t : kT_p \rightarrow (k+1)T_p$. It is at this stage, after which the UAS actually starts tracking the generated trajectory with the help of a tracking controller and is referred to as 'waypoint navigation.' Since quadrotor dynamics assumed for UAS are differentially flat, any smooth trajectory in $\sigma(t) = (x(t), y(t), z(t), \psi(t))$ can be followed by the UAS [22].

D. Local collision avoidance

Performing the waypoint navigation, with midpoints of cells belonging to the skeleton of the sparse zone as waypoints, the UAS is restricted to traverse in the sparse zones avoiding high traffic density regions. However, as the definition of the sparse zone (zone having at most one known intruder) suggests, the UAS may or may not encounter this one known intruder. In case this intruder or any sudden appearing unknown intruder comes in the vicinity of the UAS, then the 'Local collision avoidance' submodule of Fig. 4 will be invoked. The algorithm that we have proposed to handle such a scenario performs well in a sparse environment. The UAS and intruders have been assumed to be point objects.

Detection: Let a UAS, H moving along skeleton $\mathcal{S}$ be equipped with some proximity sensor M such that it can sense the position and velocity of the intruder I. The velocity of I is assumed constant. Based on the position of intruders obtained from M; Range (R), Octant of intruder $O_c(I)$, and its estimated trajectory $T(I)$ is obtained. (R) is the the shortest distance (L_2) between H and I in a three dimensional space.

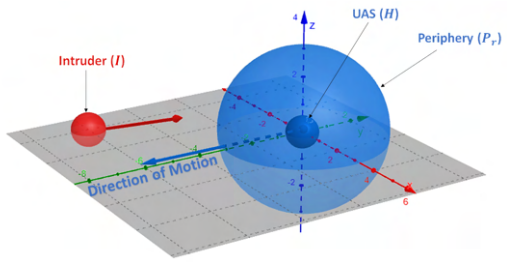


Fig. 11: UAS with periphery P_r

In order to detect the intruder, the H is assumed to have a virtual periphery P_r of radius r_p around it (Fig.11).The range R is continuously monitored by H. When $R \leq r_p$, the periphery of UAS is said to be breached by I. The path $T(I)$ and the octant $O_c(I)$ is determined, and subsequently, the collision avoidance strategy is employed.

Collision avoidance strategy: The neighborhood of H is divided into eight virtual octants in a three-dimensional space. The H is situated at the origin of those octants. The octants are divided into two virtual hemispheres, namely Forward (F_h) and Backward (B_h) hemisphere. The set of four octants which are ahead of the UAS in the direction of motion lie in F_h, and rest of them belongs to B_h, see Fig.12, $F_h = \{Oct3, Oct4, Oct7, Oct8\}$ and $B_h = \{Oct1, Oct2, Oct5, Oct6\}$.

Collision of H with I can be treated as a coplanar or non-coplanar engagement depending on the number of I in vicinity. In coplanar engagement with H at origin, the plane being divided into quadrants, I following a straight

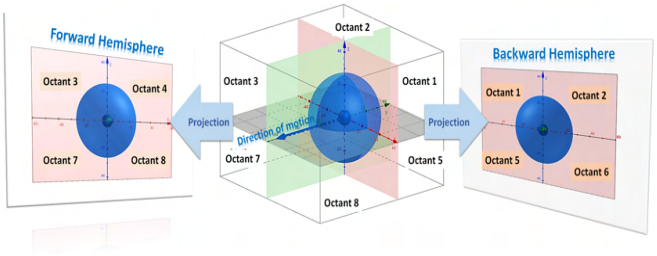


Fig. 12: Forward and Backward hemisphere around H

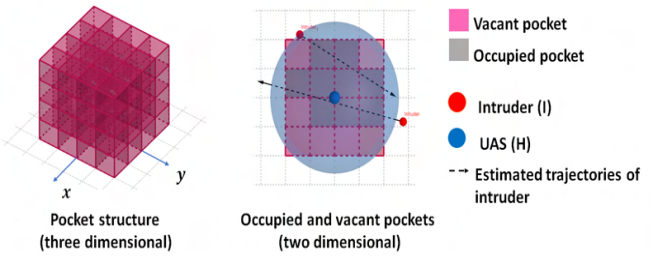


Fig. 13: Virtual pockets around H

line trajectory not passing through origin, it may be noted the cardinality of set of quadrants through which the trajectory passes will be atmost three. Similarily in a non-coplanar engagement, three dimensional space being divided into octants with H as origin, an I moving along straight line trajectory not passing through origin, will pass through at most four octants.

The proposed collision avoidance strategy is based on this idea of the identification of vacant quadrant or octant in forward or backward hemisphere. Whenever an intruder I breaches the periphery through point q on P_r, its path is estimated within P_r, through (7).

$$\frac{x-a}{r} = \frac{y-b}{s} = \frac{z-c}{t} \tag{7}$$

where a, b, c are coordinates of point q and r, s, t are direction vectors $\overrightarrow{d}$ such that $d(r,s,t) \in \mathbb{R}$. For example, in case $\overrightarrow{d}$ is $d(r,0,t)$, then (7) becomes, $\frac{x-a}{r} = \frac{z-c}{t}$, $y = b$. The estimated trajectory of intruder $T(I)$ within periphery P_r is examined for the octants $O_c(I)$ through which it passes. The cardinality of the set of octants occupied by intruder $|O_c(I)| \leq 4$. In order to escape the intruder a set of available vacant octant for H, $V_O(H)$, is inspected. It is preferred to have vacant octant from set F_h such that $V_O(H) \cap F_h \neq \phi$, so as to have shortest path for escape. The cardinality of set of forward hemisphere $|F_h| = 4$, so, $T(I)$ can occupy maximum three quadrants in F_h. Hence, if there is one I within P_r, there exist a non-empty set $W = V_O(H) \cap F_h$ such that $|W| \geq 1$. This ensures that at-least one octant will be available in F_h for escape. If there are more than one threat (could be another UAS or intruder), let us say 2, then the following cases are possible.

1) **Breach from same Octant:** In this case, both the threats breach P_r from the same octant simultaneously. Here, cardinality of occupied octants $1 \leq |O_c(I)| \leq 7$, and so, the number of available vacant octants $|V_O(H)| \geq 1$. Hence, it is guaranteed to have at-least one vacant octant for escape. But, set W may or may not be null. If $W = \phi$, then octant remaining at the backward hemisphere B_h is selected for escape.
2) **Breach from Different Octant:** In this case, both the threats breach P_r from different octants simultaneously. Here, the cardinality of the set of occupied octants $2 \leq |O_c(I)| \leq 8$ and so, the number of available vacant octants $|V_O(H)| \geq 0$. Hence, there is a possibility that $|V_O(H)| = \phi$. In such case the strategy given below is adopted,
   a) Sectorize the neighborhood of H into virtual pockets p of appropriate size in three-dimensional space, such that they can accommodate H, see Fig. 13. This sectorization is denser than the previous one. The number of pockets required is proportional to,
      i) The number of possible imminent threats present in environment. Higher the number of threats, larger should be the number of pockets.
      ii) The size of periphery P_r. A larger P_r require bigger pockets.
   b) Find a vacant pocket, preferably lying in F_h.
   c) Position the UAS within the vacant pocket until threats are out of P_r.

IV. Simulation Results

The proposed path planner is tested over a miniature UAM simulation environment of dimension 300 m×300 m×120 m with static obstacles of varying dimensions. Obstacles that could be modeled by convex sets were considered for ease of simulation. However, it can be extended to semi-algebraic sets as well. Based on road traffic and altitude restrictions in UAM, the altitude of the workspace is restricted between 12 m - 100 m.

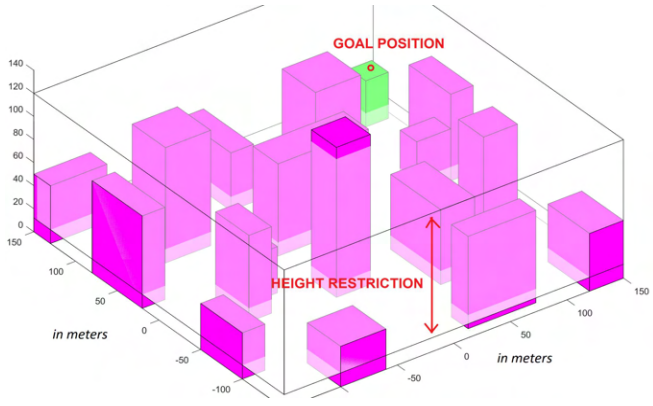


Fig. 14: Simulation Workspace Environment with static obstacles and ground height restrictions.

The workspace $\mathcal{W}$ is modeled as bitmap array by dividing each axis into $N = 30$ segments, giving 27000 cells

of dimension $\delta = 10$ m$\times$10 m$\times$4 m. Each cell is addressed as $w_{(n_1,n_2,n_3)}$ where $n_1, n_2, n_3 \in [1, N]$. The obstacles are modeled as convex sets. Using $\underline{\mathcal{W}}$, we store the 3-tuple n of cells that are enclosed by the half-planes defining the boundaries of obstacles or intersected by boundaries of $\mathcal{W}$ in the array $\mathcal{BM}space$.

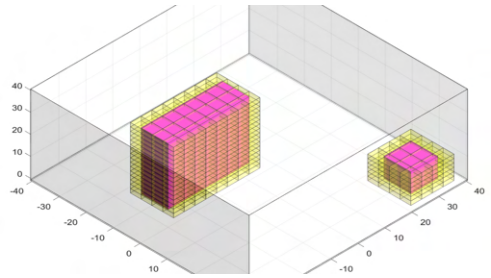


Fig. 15: Cells occupied by obstacles shown in yellow.

Eight constant velocity intruders were considered, each traversing along a straight line route, starting at some random position on the line, moving up or down the line with arbitrary velocity as illustrated in Fig. 16.

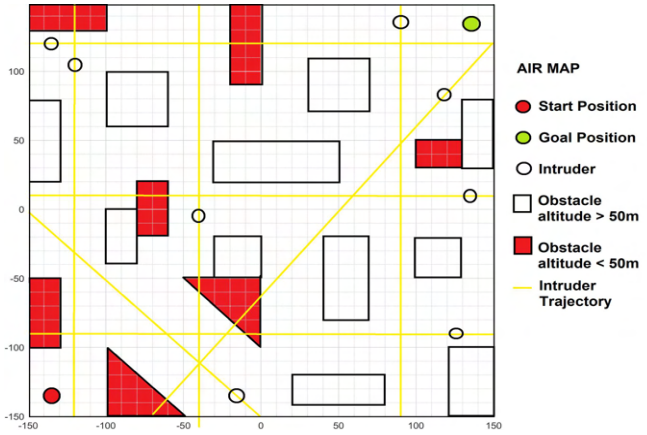


Fig. 16: Air map of known Intruders and obstacles in the workspace environment. UAS is indicated in red circle.

A. Zone Determination

Figure 17 illustrates the zone determination algorithm wherein at instants $t = kT_p$ seconds, $T_p = 10$, $k = 0, 1, ...$, the position of intruders (depicted as yellow circle) are predicted T_p seconds ahead in time (depicted as red circle), and based on this predicted position, $\mathcal{W}$ is partitioned in 2^r zones, starting with 2^1 partitions until all the conditions are satisfied. The goal is within the threshold limits of the present UAS position (shown in blue circle). At $r = 2$, all three conditions are violated. At $r = 3$, the zone enclosing the present position is still a dense zone. At $r = 4$, the sparse zones enclosing the present and goal position of UAS are not connected through sparse zones. Finally, at $r = 5$, all conditions are satisfied and the *dense_zone_definition* is updated in the obstacle set. The *dense_zone_definition* does not change for T_p seconds. In the next iteration, it may or may not change depending on the spread of zones.

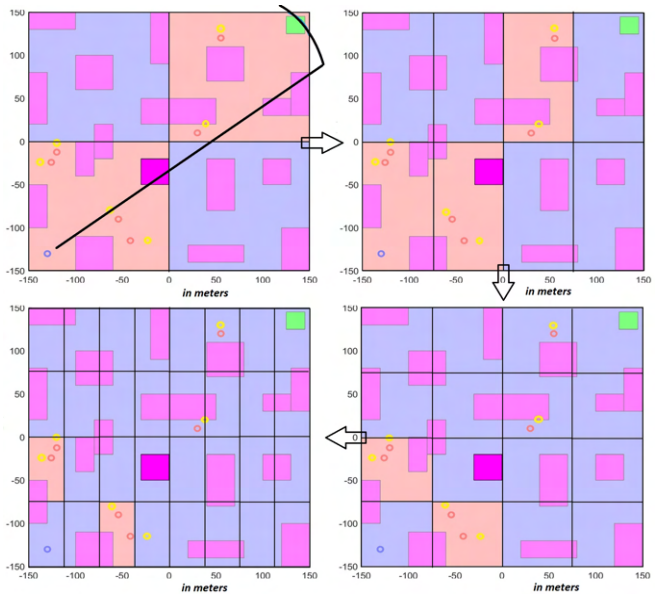


Fig. 17: Zone Determination based on predicted intruder positions, yellow and red circles indicate intruder present and predicted positions, blue circle for UAS present position.

B. W-Potential Generation

Appending $\mathcal{BM}space$ with n of cells occupied by dense zone, we get $\mathfrak{W}_{occupied}$ from which $\mathfrak{W}_{empty}$ can be obtained. For extracting W-skeleton $\mathcal{S}$, we begin wave propagation from cells in $\{w_{n'} \mid w_{n'} \in \mathcal{N}_1(w_n),\ n \in \mathfrak{W}_{occupied},\ n' \in \mathfrak{W}_{empty}$ at the origin following the algorithm in Fig. 9. Figure 18 shows the snapshots of the W-skeleton as it is being extracted from the empty space. Once the W-skeleton is extracted, the goal cell is connected to the skeleton, followed by assigning potential by wave propagation with the origin at the goal cell. The W-potential so obtained resembles Equation 6.

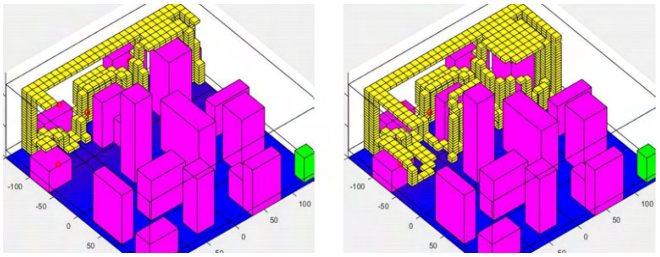


Fig. 18: Snapshots of the W-Skeleton, cells depicted in yellow, $\mathcal{S}$ being extracted from $\mathfrak{W}_{empty}$ space

C. Gradient motion and Waypoint navigation

The UAS is modeled as a single control point $\mathcal{P}$ with midpoint of cell w_g being the goal position. The W-potential is generated with respect to w_g. Gradient motion is performed every $T_p = 10$ seconds searching for at most $c = 3$ desired cells in $\{w_n : n \in \mathfrak{W}_{empty}\}$ whose midpoints could serve as waypoints for generating a minimum jerk trajectory given

by Equation 8, for time $t : (t_0 = kT_p) \rightarrow (t_f = (k+1)T_p)$.

$$\sigma^*(t) \;=\; arg\min_{\sigma(t)} \int_{t_0}^{t_f} (\dddot{\sigma_1}^2 + \dddot{\sigma_2}^2 + \dddot{\sigma_3}^2 + \dddot{\sigma_4}^2)\; dt \quad (8)$$

where $\sigma(t)$ is piecewise continuous function given by

$$\sigma_n(t) = \begin{cases} \sigma_n^1(t) = \sum_{i=0}^{5} b_n^{(1,i)} t^i \;, & t_0 < t < t_1 \\ \quad \ldots \\ \sigma_n^c(t) = \sum_{i=0}^{5} b_n^{(c,i)} t^i \;, & t_{f-1} < t < t_f \end{cases}$$

subject to waypoint position and velocity constraints for ensuring the smoothness of trajectory. Figure 19-20 illustrates the waypoint navigation for a simulation time of 130 seconds with 'zone determination', 'gradient motion' submodules being invoked every $T_p = 10$ seconds and 'W-potential generation' invoked when there is a change in *dense_zone_definition*.

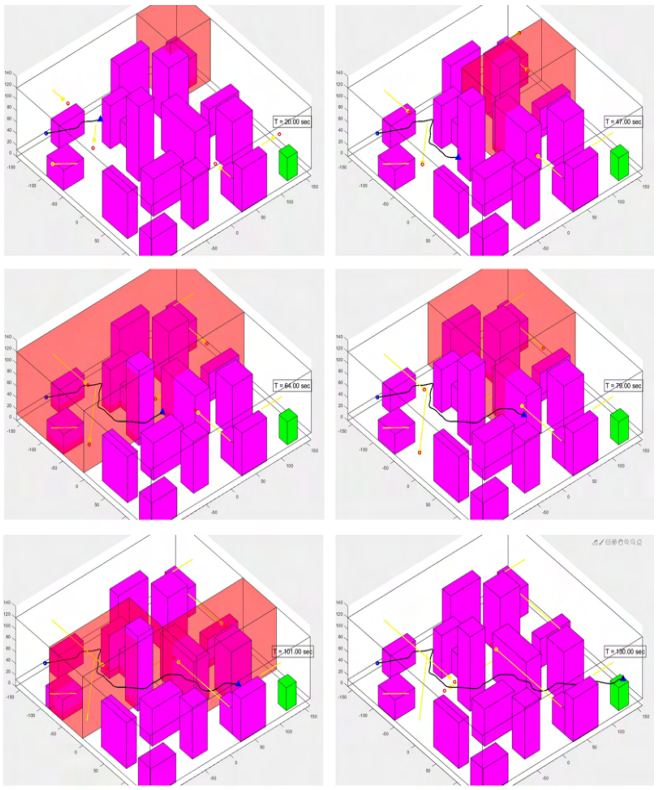


Fig. 19: Snapshots of UAS performing the way point navigation (black trajectory) from start (blue circle) to goal (green) in $\mathcal{W}$ duly avoiding static obstacles (magenta) along with intruders (red circle) in the dense zones (red patch) of $\mathcal{W}$.

D. Local collision avoidance

The simulation for collision avoidance of host UAS with two threats when detected (a known and an unknown intruder) for both coplanar and non-coplanar engagement is shown with $r_p = 2.5$ m, and velocity of UAS and intruder is < 0.5 m/s.

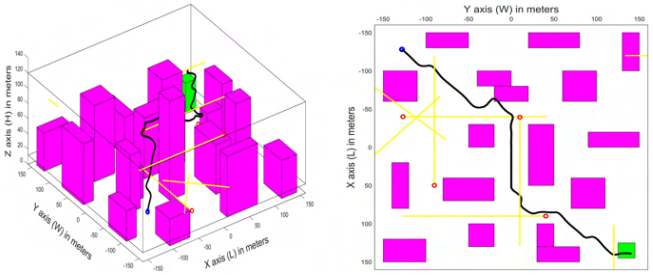


Fig. 20: Waypoint navigation from start (blue circle) to goal maintaining optimum standoff distance from static obstacles.

1) **Coplanar collision avoidance:** In the case of coplanar collision avoidance, the UAS and the intruder move in the same plane. Figure 21 shows the motion of an intruder and two UAS with periphery at different instances of time, right from the beginning, to engagement and then separation. The trail shows the path traversed by moving objects. During motion, both the UAS avoid collision among themselves as well as with the intruder. Also, the selection of octants by the UAS is dynamic depending upon the position of imminent threat.
2) **Non-coplanar collision avoidance:** In non-coplanar condition, the UAS and the intruder move in a different plane, and unlike coplanar engagements, here the collision avoidance strategy has been extended to all the eight octants space. Fig. 22 shows the motion of UAS and intruder in three-dimensional space, from initial to final positions of UAS along with the trails.

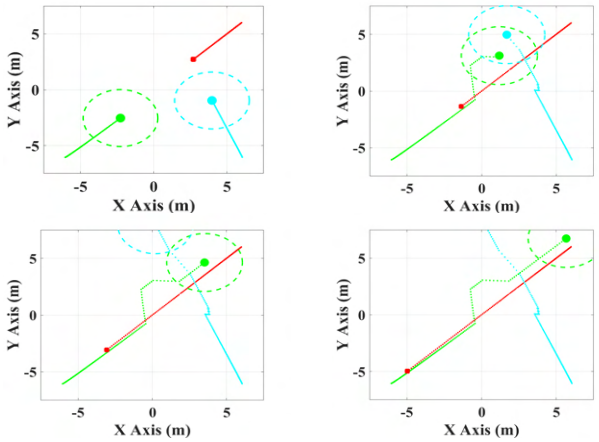


Fig. 21: Coplanar collision avoidance. Positions of UAS-1 (cyan), UAS-2 (green) and intruder (red) at different instances.

The simulation results show that the local collision avoidance strategy discussed above has the capability to cater to imminent threats in three-dimensional space. The strategy works well, as long as the collision avoidance algorithm finds a vacant octant or pocket to escape at any instant of time. The order of complexity of this algorithm is $\mathcal{O}(n)$

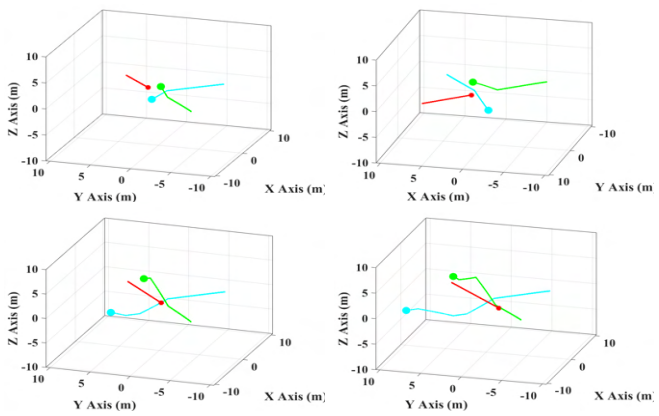


Fig. 22: Non-coplanar collision avoidance. Positions of UAS-1 (cyan), UAS-2 (green) and intruder (red) at different instances.

and in case of denser sectorization $\mathcal{O}(\frac{np}{8})$ where n is the number of approaching threats to UAS. The UAS moves along the skeleton S, formed using the numerical potential field. During avoidance of collision, they depart from the skeleton in order to find a safer place, but after threats have elapsed, the UAS returns back to the skeleton due to the nature of W-potential and then reaches the goal.

V. Conclusion

In this paper, the proposed planner for UAS flight paths is based on zone determination and numerical potential field which generates a safe path from start to goal, maintaining optimum standoff distance from the static obstacles as well as from dense, dynamic known intruder traffic. It is capable of detecting and avoiding any potential collision with known and unknown intruders while maneuvering in the densely populated urban environment. The proposed method is scalable in terms of handling a growing number of static obstacles and known intruders in the environment. The method proposed for local collision avoidance is simple but serves the purpose while cutting down the computation complexity. Also, with numerical potential fields, the algorithm has flexibility in handling UAS with very large degrees of freedom if an application demands sub-linear growth in time complexity. The simulation results successfully demonstrate the generation of a safe and efficient flight plan for achieving UTM TCL-4 urban aerial mobility in dense low altitude airspace.

Acknowledgments. The authors acknowledge partial funding support from RBCCPS, IISc, and EPSRC-GCRF (EP/P02839/1). Authors also thank the members of the AE291 group for very useful discussions.

References

[1] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson, "Unmanned aircraft system traffic management (UTM) concept of operations," in *AIAA Aviation Forum*, 2016.

[2] A. Chakrabarty, V. Stepanyan, K. S. Krishnakumar, and C. A. Ippolito, "Real-time path planning for multi-copters flying in (UTM-TCL4)," in *AIAA Scitech 2019 Forum*, 2019, p. 0958.

[3] "The unmanned air system traffic management UTM directory." https://www.unmannedairspace.info/wp-content/uploads/2019/06/UTM-directory.-June-2019.-v1.pdf, (Accessed on 28 February 2021).

[4] M. Consiglio, C. Muñoz, G. Hagen, A. Narkawicz, and S. Balachandran, "ICAROUS: Integrated configurable algorithms for reliable operations of unmanned systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–5.

[5] L. A. Tony, A. Ratnoo, and D. Ghose, "Corridrone: Corridors for drones, an adaptive on-demand multi-lane design and testbed," *ArXiv*, vol. abs/2012.01019, 2020.

[6] ——, "Lane geometry, compliance levels, and adaptive geo-fencing in corridrone architecture for urban mobility," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS), Greece*, 2021.

[7] N. Samir Labib, G. Danoy, J. Musial, M. R. Brust, and P. Bouvry, "Internet of unmanned aerial vehicles—A multilayer low-altitude airspace model for distributed UAV traffic management," *Sensors*, vol. 19, no. 21, p. 4779, 2019.

[8] B. Pang, Q. Tan, T. Ra, and K. H. Low, "A risk-based UAS traffic network model for adaptive urban airspace management," in *AIAA Aviation Forum*, 2020, p. 2900.

[9] P. Wu, L. Li, J. Xie, and J. Chen, "Probabilistically guaranteed path planning for safe urban air mobility using chance constrained RRT," in *AIAA Aviation Forum*, 2020, p. 2914.

[10] F. Adolf, A. Langer, L. d. M. P. e Silva, and F. Thielecke, "Probabilistic roadmaps and ant colony optimization for UAV mission planning," *IFAC Proceedings Volumes*, vol. 40, no. 15, pp. 264–269, 2007.

[11] L. R. Sahawneh, M. E. Argyle, and R. W. Beard, "3D path planning for small UAS operating in low-altitude airspace," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 413–419.

[12] M. G. Park and M. C. Lee, "Experimental evaluation of robot path planning by artificial potential field approach with simulated annealing," in *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, vol. 4. IEEE, 2002, pp. 2190–2195.

[13] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using laplace's equation," in *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, 1990, pp. 2102–2106.

[14] A. A. Masoud, "Motion planning with gamma-harmonic potential fields," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 2786–2801, 2012.

[15] P. Vlantis, C. Vrohidis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Orientation-aware motion planning in complex workspaces using adaptive harmonic potential fields," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8592–8598.

[16] T. He, I. Mantegh, L. Chen, C. Vidal, and W. Xie, "UAS flight path planning for dynamic, multi-vehicle environment," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 211–219.

[17] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[18] A. Chakravarthy and D. Ghose, "Generalization of the collision cone approach for motion safety in 3-D environments," *Autonomous Robots*, vol. 32, no. 3, pp. 243–266, 2012.

[19] A. Kumar, R. Krishnamurthi, A. Nayyar, A. K. Luhach, M. S. Khan, and A. Singh, "A novel software-defined drone network (SDDN)-based collision avoidance strategies for on-road traffic monitoring and management," *Vehicular Communications*, p. 100313, 2020.

[20] A. Mcfadyen, L. Mejias, P. Corke, and C. Pradalier, "Aircraft collision avoidance using spherical visual predictive control and single point features," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 50–56.

[21] J. Yu, Z. Cai, and Y. Wang, "Minimum jerk trajectory generation of a quadrotor based on the differential flatness," in *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*. IEEE, 2014, pp. 832–837.

[22] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.