Temporal and Spatial Routing for Large Scale Safe and Connected UAS Traffic Management in Urban Areas

Ziyi Zhao[1*], Zhao Jin[1*], Chen Luo[1], Haowen Fang[1],
Franco Basti[3], M. Cenk Gursoy[1], Carlos Caicedo[2], Qinru Qiu[1]
[1] *Department of Electrical Engineering & Computer Science*, *Syracuse University*, Syracuse, NY 13244, USA
[2] *School of Information Studies*, *Syracuse University*, Syracuse, NY 13244, USA
[3] *Thales Digital Aviation Customer Success and Innovation*, *Thales USA*, Arlington, VA 22202, USA
{zzhao37, zjin04, cluo05, hfang02, mcgursoy, qiqiu}@syr.edu, ccaicedo@syr.edu, franco.basti.e@thalesdigital.io

***Abstract*—Small Unmanned Aircraft Systems (sUAS) will be an important component of the smart city and intelligent transportation environments of the near future. The demand for sUAS related applications, such as commercial delivery and land surveying, is expected to grow rapidly in next few years. In general, sUAS traffic scheduling and management functions are needed to coordinate the launching of sUAS from different launch sites and plan their trajectories to avoid conflict while considering several other constraints such as expected arrival time, minimum flight energy, and availability of communication resources. However, as the airbone sUAS density grows in a certain area, it is difficult to foresee the potential airspace and communications resource conflicts and make immediate decisions to avoid them. To address this challenge, we present a temporal and spatial routing algorithm for sUAS trajectory management in a high density urban area. It plans sUAS movements in a spatial and temporal maze with the consideration of obstacles that are either static or dynamic in time. The routing allows the sUAS to avoid static no-fly areas (i.e. static obstacles) or other in-flight sUAS and areas that have congested communication resources (i.e. dynamic obstacles). The algorithm is evaluated using an agent-based simulation platform. The simulation results show that the proposed algorithm outperforms reference route management algorithms in many areas, especially in processing speed and memory efficiency. Detailed comparisons are provided for the sUAS flight time, the overall throughput, the conflict rate and communication resource utilization. The results demonstrate that our proposed algorithm can be used as a solution to improve the efficiency of airspace and communication resource utilization for next generation smart city and smart transportation.**

***Index Terms*—smart city, sUAS, trajectory routing, temporal-spatial traffic management, UTM**

I. Introduction

The introduction of unmanned aircraft systems (UAS) into the global airspace creates both an opportunity and a challenge for the aviation industry as a whole. The traffic demand from new entrants in low-altitude airspace is forecasted to be orders of magnitude far greater than existing commercial aviation. Demands for controlling and monitoring this airspace will increase in particular in large, metropolitan areas. By 2022 the FAA expects over 2 million hobbyist sUAS (under 55 lbs) and over 450,000 commercial Unmanned Aerial Vehicles (UAVs) to be in operation [1]. Emerging solutions such as UAS Traffic Management (UTM) will soon be required to manage the increased traffic within the airspace.

Even with increased on-board autonomy, given the large number of the drones in the air, a centralized coordinator, or multiple coordinators attending different zones are necessary to ensure mission efficiency and safety. This requires the sUAS to have reliable communications with one or several ground control stations (GCSs) supported by the resources of a reliable communications network to help ensure the safety of the UAS flight operations. Enabling high-rate, low-latency and reliable wireless communications between the sUAS and their GCS is key to ensure their incorporation into the provision of services to society.

We consider both air space and communication network capacity as resources. The demands and utilizations of these two resources in a sUAS system are highly correlated, and hence should be managed together. Our goal is to plan the trajectory of each sUAS to better utilize those two resources, such that there are no conflicts between the trajectory of any two sUAS and a good quality connection to cellular base stations is maintained during the entire mission.

In this paper, we present a temporal and spatial (T-S) routing algorithm for sUAS trajectory planning. The algorithm helps in the management of the air traffic of a metropolitan area that has high sUAS densities by assisting in the planning of each sUAS trajectory in advance. The centralized management ensures sUAS safety by proactively avoiding conflicts while ensuring the availability of communication resources. This comes with a marginal cost of extended flight time and longer launching intervals. Using a multi-agent air traffic resource usage simulator (MATRUS) [2], we compare the performance of our proposed algorithm to other methods. Simulation results show that, compared to the scenario without traffic management, our proposed UAS routing algorithm completely eliminates the potential conflicts while maintaining a 100% connectivity during the mission with 2$\sim$3.3% reduction in throughput and

*These authors contributed equally.

978-1-7281-3197-9/19/$31.00 ©2019 IEEE

less than 2.74% increase in flight time.

The sUAS communicate with ground control stations (i.e. a command and the control center) through the cellular network, hence the availability of the communication resources will impact the decisions related to sUAS traffic management. The effectiveness of such management will eventually determine the scale of sUAS applications/services that can be supported by the given air space and existing communication/monitoring infrastructure, and also shape the planning of future infrastructure deployment. The UAS traffic management system must be fully autonomous, so that it can handle a large number of sUAS simultaneously and continuously.

The rest of the paper is arranged as follows: In Section II, we review related work in UAS route management and conflict avoidance. This is followed in Section III by details about the proposed algorithm. Section IV describes our experiments and evaluation of the results. Finally, Section V summarizes this work.

II. Related Work

Over the past 5 years, sUAS have played an increasingly critical role in many fields [3]. With the rise in popularity of sUAS, many important and notable issues regarding sUAS traffic management have been discovered. Therefore, numerous methods and paradigms have been proposed to solve these issues. These proposed methods can be divided into two main categories. One category focuses on the centralized scheduling and management of multiple sUAS, via unmanned air system traffic management systems [4] [5]. We refer to these approaches as centralized control. The other category focuses on the actions of a single sUAS, such as obstacle and collision avoidance [6], and is referred to as distributed control.

Much of the existing UAS trajectory generation research focuses on the generation of trajectories for a single sUAS that are energy efficient and stable. Constraints such as obstacle avoidance and rigid body dynamics are considered. Some of the classical approaches apply rapidly-exploring random trees [7] and Voronoi graphs [8] [9]. In [10], the author presented an indoor algorithm to navigate sUAS to avoid collisions. By importing geometrical constraints, [11] proposed a solution to avoid collisions in a static environment. In [12], to improve obstacle avoidance for sUAS, a method based on optical flow is presented. Others use machine learning approaches. For example, [13] [14] developed a deep reinforcement learning framework that learns how to perform energy efficient waypoint planning. However, those works assume static obstacles and a single sUAS.

In addition, multiple sUAS trajectory planning has been studied as a multi-agent cooperative system and solved in a rolling horizon approach using dynamic programming [15] or mixed integer linear programming [16]. Another strategy involves setting an artificial reactive field around each UAS [17]. However, these approaches do not consider any additional resources other than the airspace. The availability of the communication resource has not been integrated as a constraint into these frameworks.

Most of the existing trajectory planning algorithms consider continuous Euclidean space and the sUAS can have an arbitrary trajectory as long as certain constraints are satisfied. Hence a closed-form representation of the trajectory can be obtained. Although this may allow us to find simple analytical optimal solutions, it leads to unstructured trajectories. When the number of sUAS increases, such irregularity leads to a traffic pattern that is unpredictable. Furthermore, with a large number of sUAS, to describe all constraints (i.e. collision avoidance) in closed-form and solve the optimization problem analytically is almost impossible. Recently, a very strict and rigid airspace structure to handle dense operation in the urban low altitude environment was suggested by work on Unmanned Aircraft System (UAS) Traffic Management (UTM) at NASA in [18]. The author explored UAS operations in non-segregated airspace and managed the risk of mid-air collision to a level deemed acceptable to regulators. In the paper, the airspace is divided into multiple layers, and the layers are further divided into orthogonal sky lanes. However it focused only on defining the regulations of the UTM system instead of solving any optimization problems.

In this work, we focused on optimizing the computed trajectory for sUAS in order to prevent collisions, achieve shortest distance, and at the same time meet other realistic environmental constraints [19]. These constraints include avoiding no-fly zones or restricted areas, avoiding areas that do not have cellular signal coverage or that are temporarily experiencing congestion in the cellular network, etc. These constraints will be modeled as static and/or dynamic obstacles in the airspace, and can be considered simultaneously as part of the airspace management environment.

III. The MATRUS Platform & Trajectory Routing Algorithm

A. Platform Structure

The MATRUS platform is an integrated environment for air traffic simulation, communication resource estimation, data analysis, and traffic animation for sUAS applications [2]. The simulation platform was developed to evaluate UAS air traffic management policies over a metropolitan area. The modularized design considers each sUAS and the base stations of the communications network as an agent, hence, it provides an interface for us to plug-and-play different resource management policies and evaluate their performance.

As shown in Fig. 1, the MATRUS framework consists of several components, a multi-agent simulator, a temporal-spatial (T-S) maze router, that performs simple trajectory planning, a trajectory animation tool based on Google Earth, and a data analysis tool. The geographical environment information, such as location and size of no-fly zones, and the mission information for each UAS, including its start time, start and end locations are provided to the simulator as the input.

The core functionality of MATRUS is built on top of the REPAST (Recursive Porous Agent Simulation Toolkit) Simphony platform [20]. Agent-based techniques are used for the modeling of UASs and base stations. Each UAS agent will

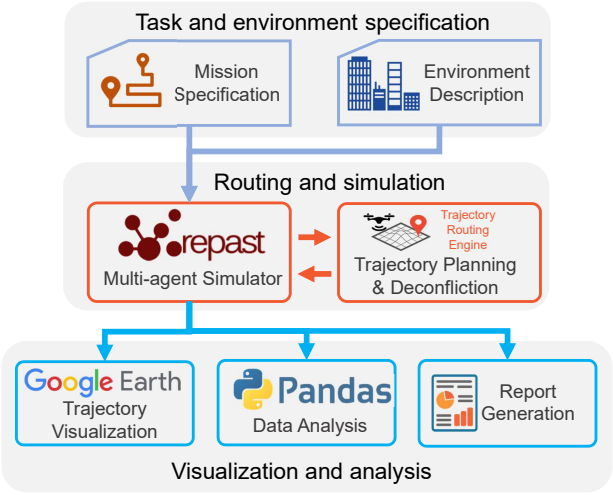


Fig. 1: The Structure of MATRUS Framework

follow the predefined trajectory and update its location and speed periodically. It will also connect to the available base station channel that has the highest signal quality. Each base station will provide a communication link to a UAS if it has the resources to do so and it will keep track of the remaining available resources (i.e. channels). The wireless communication link is characterized by a distance-based path loss model that considers the radio frequency (RF) power dissipation as a function of the communication distance. Probabilistic line-of-sight and non-line-of-sight propagation models are incorporated into the link characterization. By updating the agent model, the simulator can easily be extended to consider more complex behavior and interactions between UASs and base stations.

B. Environment Assumptions

One of the main objectives of sUAS traffic management is to maximize the throughput while avoiding any potential conflicts. The conflict is defined as the situation in which the distance between two sUASs is smaller than the given threshold. We divide deconfliction techniques into two categories, reactive and proactive. A sUAS with reactive deconfliction capabilities perceives an imminent conflict and adjusts its trajectory locally to avoid it. The conflict could be detected via on-board sensors, or through communication with nearby sUAS and the control center. In either case, to ensure safety operation, the sUAS needs to have a high amount of computing power to respond in a short time and avoid the conflict. Furthermore, reactive deconfliction leads to unpredictable traffic patterns. During trajectory adjustment, a sUAS not only needs to consider the upcoming conflict, but also any potential new conflict that may be caused by the changing of its current trajectory. In a high density area, this problem will soon become intractable. This has been confirmed by the works in [21] [22], which stated the importance of architecting a UTM solution capable of handling high UAS traffic demand and that in some situations free flight operations with fully decentralized trajectory planning are not feasible or will result in very inefficient airspace operations.

The proactive deconfliction technique plans a conflict free trajectory for each sUAS at launch time or at the time when it enters controlled airspace. Because the control center has the trajectory information of all sUAS in a designated airspace, it can easily find a conflict free path for the incoming sUAS if such path exists. If such path cannot be found, the launch of the sUAS will be delayed or the sUAS cannot enter the airspace until a path is available. Although the routing procedure may have high complexity, it is done in the control center, hence energy or computing resources will not be a limiting factor. Other air traffic constraints, such as no-fly zones can easily be integrated into the routing procedure.

With a large number of sUAS in a given airspace area, if the sUAS trajectory has the freedom of taking any angle at any speed, the routing will be extremely difficult as the search space is infinite. Some constraints on the sUAS trajectory must be imposed to reduce the route planning search space. In this work, we adopt the “sky lane” concept proposed by the NASA UTM group [18] and limit the trajectory to be a Manhattan style trajectory, i.e. the sUAS can only make 90 degree turns, and they fly at constant speed. To improve the predictability and to ease the de-conflict cost. Similar to [18], we divide the airspace using a grid pattern. The size of the grid cell is defined by the minimum separation distance between UASs for safe operations. It is guaranteed that UASs will be conflict-free if they travel along the center of each grid cell.

To simplify discussion and illustration, we assume that all sUASs fly at the same height and our search space consists of only 3 dimensions: x, y and t (i.e. two dimensions for space and one for time). The entire 3 dimensional space is divided into equal sized grids as shown in Fig. 2. The size of each grid cell is (W, W, δ), where W is the minimum distance between sUASs which ensures that they are not in conflict. The amount of time that a UAS needs to travel a distance W is denoted as δ. A function $M(x, y, t) \rightarrow \{-1, 0\}$ maps each grid to a label, where ”-1” represents an occupied grid cell and “0” represents unoccupied grid cell. As we can see, if location (x, y) belongs to a no-fly zone, then $M(x, y, t) = -1$, $0 \leq t \leq \infty$. If a UAS flies through a location (x, y) at time t_1, then $M(x, y, t_1) = -1$. A UAS trajectory starts from a source grid cell (x_s, y_s, t_s), and end at any one of the destination grid cells, (x_d, y_d, t_d), where (x_s, y_s) and (x_d, y_d) are the coordinates of the flight source and destination, respectively, and t_s, t_d are the flight start time and arrival deadline. We refer to this grid system as the temporal-spatial (T-S) maze.

Traditional maze routing has been widely used in electronic design automation to route the on-board or on-chip interconnects. Two interconnects cannot occupy the same grid cell, otherwise it will cause a short circuit. However, by using the t axis, two sUASs can occupy the same space as long as they are there at different times. Hence, the trajectory search space should consist of three dimensions, x, y and t. Two sUASs are conflict free, if their trajectory has no intersection in the multi-dimensional spatial and temporal space. In a T-S maze,

obviously, any route must move towards the direction where t increases.

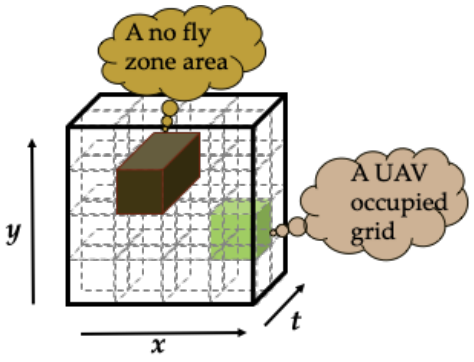
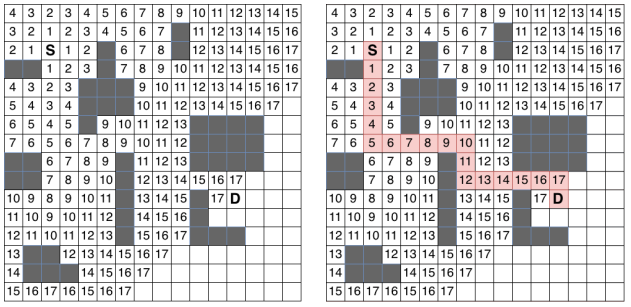


Fig. 2: 3 Dimensional Spatial-Temporal Environment

C. Baseline T-S Routing Algorithm

In this paper, the baseline method we selected is a Breadth First Search (BFS) based maze routing algorithm. It consists of two stages, flooding and traceback, as shown in Fig. 3. The flooding stage essentially performs a breadth first search. Starting from the source grid, every non-occupied grid is labeled by its Manhattan distance from the source grid cell using BFS until one of the destination grid cells is reached. Then following the descending order of the labels, a path is traced back from the destination to the source. In order to reduce the control thrust during the flight, the grid cell that is in the same direction (towards the source) as the previous move will be picked with higher priority during the traceback to reduce the number of turns in the trajectory. The complete algorithm is given in Algorithm 1. In the pseudo code and the following content, the neighbor is defined as the unexplored non-diagonal adjacent cells of current position.



(a) Stage 1: flooding (b) Stage 2: traceback

Fig. 3: Two Stage BFS Routing Algorithm

D. Sparse Represented Temporal-Spatial (SRTS) Routing

In the worst case, the BFS routing needs to label all grids in the 3D T-S maze to reach the destination. A naive implementation has the space complexity $O(X \times Y \times T)$, where X, Y and T are the maximum dimensions of the airspace and the maximum time that the air traffic will last. Its memory and computation complexity is prohibitively high. To improve the routing computation speed for real-time applications, we propose a Sparse Represented TS routing (SRTS) procedure based on the A$*$ routing algorithm [23] [24] [25] [26].

T-S Routing $(E(x', y', t'), s_x, s_y, s_t, d_x, d_y)$;
Input : Environment Matrix, Start Coordinates(s_x, s_y, s_t), Destination Coordinates(d_x, d_y)
Output : The optimal routing trajectory of one UAS

Queue Q;
Q.enqueue($start\_position$);
$mark\ start\_position\ as\ VISITED$;
while $Q \neq \emptyset$ **do**
  $node$ = Q.poll();
  if $node == destination$ **then**
    $set\ DestinationTimeStep$;
    break;
  foreach $neighbor \in Neighbors(node)$ **do**
    if $neighbor\ IS\_VALID$ **then**
      $mark\ neighbor\ as\ VISITED$;
      Q.enqueue($neighbor$);
if $DestinationTimeStep\ IS\_EXIST$ **then**
  $get\ trajectory\ by\ TRACEBACK$;
foreach $position \in trajectory$ **do**
  $mark\ E[position.x][position.y][position.t]\ as\ OBSTACLE$;
return $trajectory$;

Algorithm 1: T-S Routing Algorithm

The UAS route planning problem has to consider two types of obstacles, static and dynamic. The static obstacles refer to the invariant geographical constraints in the route planning area. The dynamic obstacles represent the time variant constraints, as explained in Fig. 2. Fig. 4 further explains the interactions among dynamic obstacles. The green grid cell represents the sUAS for which a trajectory was recently planned, the red grid cell stands for the landing area for that sUAS, and all the grey cells represent the locations of other sUASs whose trajectories have been planned in previous time steps. In Fig. 4a, each obstacle is labeled with an arrow which represents the heading direction of the sUAS for the next time step. Therefore, from Fig. 4b and Fig. 4c, we can notice that the position of the obstacles have changed.

Unlike original T-S routing, our routing method uses a 2D map with dimensions $X \times Y$. (This can be extended to a 3D map if the UASs fly in different altitudes.) The obstacles are divided into 2 categories. The static obstacles are projected onto a 2D map, each of the obstacles occupies a specific location. Since the static obstacles are time invariant, the information along the t axis is redundant and hence it can be eliminated. For the dynamic obstacles, we exploit their spatial sparsity and store them using hash tables along the t axis. Each location on the t axis is associated with a hash table, which stores the

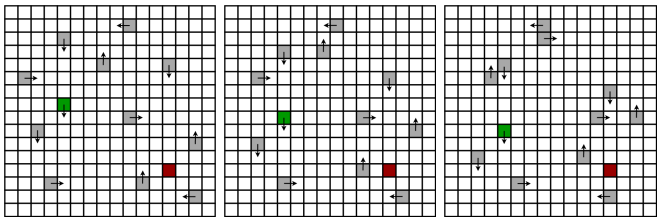


Fig. 4: Time Variance Dynamic Obstacles

(x, y) coordinates of dynamic obstacles at the corresponding time. Using an instant refreshing mechanism, the SRTS routing algorithm only stores dynamic obstacles at or beyond the current time step. All the obstacles in the prior time will not affect the trajectory planning of the current sUAS, hence will be removed automatically. In this way, the 3D environment considered in the original T-S routing is compressed into a set of sparsely represented 2D points corresponding to dynamic obstacles sampled at each time step from the present up to a future time T'. The value of T' is determined by the longest flight time for UASs currently in the air or about to launch. The experimental results show that the instant refreshing mechanism can significantly reduce the demand of the memory resources.

Using the new environment representation, the routing algorithm needs to check both 2D static obstacles and 3D dynamic obstacles to acquire next moveable neighbor cell. Only the neighbor cell which has no conflict with either type of obstacle will be selected for the next potential movements, as shown in Algorithm 2.

Candidate Selection $(currNode, ClosedList)$;
Input : Current Position(currNode), Past Selected Positions(ClosedList)
Output : The candidate neighbors of current position

$neighbors = \emptyset$;
foreach *position in Directions* **do**
  check position in 2D static projection;
  check position in 3D dynamic projection;
  $CheckSignalStrength$;
  if *position* IS_VALID *&*
  no obstacle in 2D/3D projections &
  have enough signal support &
  $position \notin ClosedList$ **then**
    position initialization;
    $neighbors$.add($position$);
return $neighbors$;

Algorithm 2: Candidate Selection Algorithm

E. Routing for Connectivity

After checking the airspace resources, the algorithm also needs to check the availability of communication resources in order to decide the next move in the trajectory. In this work, a simplified communication model is used, where each base station has N orthogonal communication channels and each channel can serve only one sUAS. At anytime, a sUAS will connect to one base station through one of the channels that are available at that base station. We adopt the following simple log-distance path loss model:

$$PL(d) = PL(d_0) + 10n\log_{10}(\frac{d}{d_0}) + x, d_f \leq d_0 \leq d \quad (1)$$

where $PL(d)$ is the path loss in dB at distance d and $PL(d_0)$ is the path loss in dB at a reference distance d_0, x is a Gaussian distributed random variable, however for simplicity, it is set to 0 in this work. The parameter n is the path loss exponent, which is set to 3 for line-of-sight links and 3.5 for non-line-of-sight. Since the sUAS connect to base stations from a high altitude, the connection has a higher chance to be line-of-sight. In this work we set the line-of-sight probability to be 0.9 and the non-line-of-sight probability to be 0.1.

We assume that a link can be established between a sUAS and a base station if the signal loss of the path between them is less than a given threshold. Otherwise, the communication link cannot be established. At any given time, the sUAS only connects to the base station that has the highest signal strength through an available channel. We assume that the connection between sUAS and base station is dynamic and we ignore the time and cost of the handover process.

Algorithm 3 shows the procedure for communication resource availability checking. First, the communication resources (i.e. channels) will be allocated to the sUASs that are already in the air to give them higher priority than the sUAS that are about to launch. After updating the base station's list of available resources, the distance between the potential new location of the UAS after a movement step and each base station will be calculated and the potential signal loss will be estimated. At the end, the movement step will be considered as a valid step if a base station with an available communication channel and acceptable link quality can be found. Otherwise, the movement step will not be considered for the current position of the UAS.

During the route planning, when checking for communication resource availability, we assume the connections between sUAS and the base stations are line-of-sight. This may not always be the case if there are obstacles present in the environment. Therefore, the estimated signal strength during the routing stage may not be the exact signal strength during the real flight. We refer to the former as the $inner\ belief$ of the resource utilization and the latter as the ground truth. Our simulation results show that, under the simple channel model, the $inner\ belief$ will be close to the ground truth.

F. Overall SRTS Routing Algorithm

The SRTS routing is performed on the 2D surface with the static obstacle information. The algorithm calculates two costs for each potential neighbor cell. The first cost is called the movement cost, which describes the expense of moving from the current position to the potential neighbor cell. For each

Signal Strength Check (x, y, t);
Input : Candidate Position(x, y, t)
Output : Whether a position can build the communication link with a base station

allocate base station resources for in-flight $UASs_{(t)}$;
sort BaseStationCandidates by distance in ascending order;
foreach *basestation in BaseStationCandidates* **do**
  get distance between UAS and basestation;
  calculate SignalLoss;
  if *basestation has available channel &* $SignalLoss \leqslant LOSS\_THRESHOLD$ **then**
    reset base station resources;
    return $TRUE$;
reset base station resources;
return $FALSE$;

Algorithm 3: Signal Strength Check Algorithm

position, the movement cost is accumulated, so it can relieve the oscillation between two adjacent positions. The other cost is called the destination cost, which stands for the expense of moving from the potential neighbor cell to the destination. The definition of the two costs are adopted from the original A$*$ routing algorithm. We choose to calculate the costs based only on the X, Y distances, because in general UAS flight energy increases as the travel distance increases. However, as a T-S routing algorithm, it is possible that the GCS will ask an UAS to stay at a specific location to wait to resolve the potential conflict. It will be part of our future work to incorporate flight time into the cost function. The pseudo code of the SRTS algorithm is given in the Algorithm 4.

IV. Experimental Results & Analysis

We demonstrate the performance of the SRTS algorithm in UAS trajectory planning on the MATRUS simulation framework. The environment settings in MATRUS contain parameters that specify the base station configurations, individual UAS action configurations and air space configurations.

In this paper, the number of base stations and the channels available for UAS communications at each base station are fixed to be 10 and 8, respectively. For each UAS, the trajectory mode is set to be Manhattan style. The UAS mission generation interval varies from 10, 20, and 30 seconds. They will be referred to as the high, medium and low traffic configurations, respectively, in the rest of this section. And, the number of no-fly zones is either none or 2. Each combination of parameters defines one specific scenario. The reported results is the average of 10 runs for each specific scenario. For each run of one scenario, the simulation time is 20,000 time steps which corresponds to 20,000 seconds. Based on the observations from the experiments, the UAS simulation behavior becomes

Sparse Represented TS Routing $(s_x, s_y, s_t, d_x, d_y)$;
Input : Start Coordinates(s_x, s_y, s_t), Destination Coordinates(d_x, d_y)
Output : The optimal routing trajectory of one UAS

$OpenList$ = $\emptyset$;
$ClosedList$ = $\emptyset$;
$OpenList$.add($start\_position$);
while $OpenList \neq \emptyset$ **do**
  $node$ = $OpenList$.poll();
  Instant Refreshing Mechanism
  if $node$ == $destination$ **then**
    $trajectory$ = retriveTrajectory($node$) ;
    break;
  $ClosedList$.add($node$);
  foreach $neighbor \in CandidateSelection$ **do**
    if $neighbor \in OpenList$ **then**
      $neighbor$ = $OpenList$.get($neighbor$)
    calculate neighbor's $NewMovementCost$;
    if $NewMovementCost < OldMovementCost$ || $neighbor \notin OpenList$ **then**
      update neighbor's $MovementCost$;
      update neighbor's $DestinationCost$;
      $OverallCost$ = $MovementCost$ + $DestinationCost$;
    if $neighbor \notin OpenList$ **then**
      $OpenList$.add($neighbor$);
foreach $position \in trajectory$ **do**
  mark $OBSTACLE$ *in* $3D$ *dynamic projection*;
return $trajectory$;

Algorithm 4: Sparse Represented TS Routing Algorithm

stable after 300 time steps. Therefore, the simulation length is sufficient for us to analyze different scenarios.

Four sUAS launching areas and four landing areas are distributed in the map, which is 90 square miles in size. Their locations are selected based on the distribution of business and residential areas in Upstate, New York. Each launching area has a different launching probability. In a given interval, each launching area will request to launch a sUAS with a given probability. For each launch request, the landing area is randomly selected from the four candidates. Given the launch and landing areas, the exact launch and landing spots are randomly picked within the areas.

One of the traffic scenario environments is illustrated in Fig. 5. In the figure, the red rectangles represent the sUAS launching areas, the blue rectangles stand for the sUAS landing areas and the grey rectangles indicate blocked areas (i.e. the no-fly zones). The location of 10 base stations are also marked on the map. The coordinates of those base stations are set based on the actual base station facilities registered with the FCC.

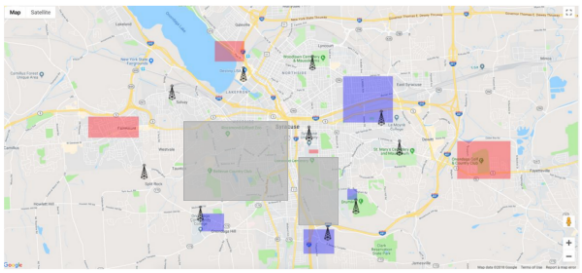


Fig. 5: Traffic Scenario Environment

A. Evaluation Metrics

Three metrics have been introduced to evaluate the performance of the routing algorithm: Average Throughput, Average Flight Time and Average Conflict Ratio.

To ensure flight safety, two flying sUAS must be separated by a sufficient distance. Because the current sUAS cannot make sharp turns or slow down immediately to a stop, leaving enough space for each sUAS is necessary for safety considerations. Therefore, centered at every location (x, y) of the map, a square is drawn whose dimension is equal to the minimum separation distance. If a square is occupied by more than one sUAS at the same time, then location (x, y) has a conflict at this particular time. By default, we use 18 meters as the minimum separation distance in the experiment.

In our evaluation, the conflict ratio is used to analyze the safety metric. It is defined as the number of missions that have encountered at least one conflict during the flight divided by the total number of launched sUAS missions.

The sUAS throughput indicates the capacity of the simulated air space. It is measured by the number of launched sUAS during a fixed time. There is a fundamental trade-off between safety and throughput. The trajectory planning algorithm can significantly reduce the conflict ratio, however it will also affect the throughput. The goal of the routing algorithm is to achieve maximum throughput while avoiding any potential conflicts.

Besides the average throughput of the entire simulated area, the performance of every single sUAS is also crucial. In this paper. The average flight time of individual sUAS has been considered as the last metric to evaluate the performance of the routing algorithm. In general, a longer average flight time indicates more detours during the flight and higher energy consumption. Hence, a viable routing algorithm should not lead to a large increase in the sUAS flight time.

B. Conflict Elimination

In the first experiment, we compare the maximum sUAS density in different air traffic scenarios, and demonstrate the effectiveness of our proposed routing algorithm. We visualize the distribution of maximum density of sUASs per grid location for the simulation scenario of high traffic without no-fly zones in Figs. 6, 7 and 8. The black boxes in the density maps represent the sUAS launching areas and the green boxes stand for the sUAS landing areas. In the density map, the light blue spots indicate normal traffic density, i.e. the maximum density of sUAS in that area is 1 sUAS per grid cell. In contrast, the bright red spot indicates conflict, i.e. the maximum sUAS density is equal to or greater than 2 in the specific location. The white areas are those where no sUAS has ever visited. Hence, the maximum density also represents the distribution of the sUAS trajectory. If a sUAS passes through the blocked area, it will be considered as a conflict. Table I compares throughput, flight time and conflict rate for traffic scenarios without routing, with T-S routing, and with SRTS routing.

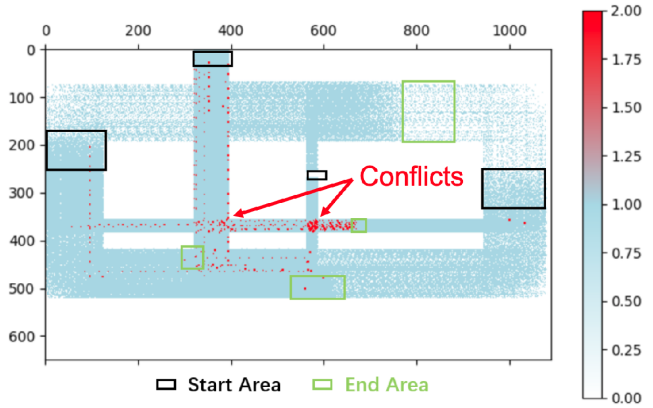


Fig. 6: sUAS Trajectory Density without Traffic Management

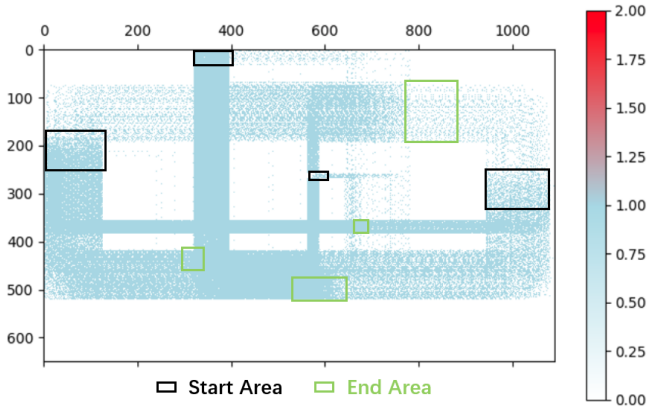


Fig. 7: sUAS Trajectory Density with Traffic Management (T-S Routing)

The first thing we can observe from Table I is that, in the heavy traffic scenario, without trajectory management, 21.73% missions will have conflicts. The conflict ratio has further increased to 46.10% if no-fly zone conflicts are also considered. By applying the traffic management, both the original T-S routing algorithm and the SRTS routing algorithm can eliminate all the conflicts. The cost is 2.2∼3.3% reduction in throughput and less than 2.74% increase in the flight time. The reason of the throughput reduction is because, with trajectory management, the sUAS that cannott find a conflict free path will not be launched. Therefore, the more restrictive the constraints are, the fewer sUAS that will be launched.

TABLE I: Routing Algorithm Comparison

Traffic Type	Avg. Throughput		Avg. Flight Time		Avg. Conflict Ratio	
	0 No-Fly Zone	2 No-Fly Zones	0 No-Fly Zone	2 No-Fly Zones	0 No-Fly Zone	2 No-Fly Zones
Heavy traffic (generation/10s)						
No Routing	4006	4006	491.65s	491.65s	21.73%	46.10%
Baseline T-S Routing	3897 (-2.72%)	3874 (-3.30%)	495.23s (+0.73%)	505.10s (+2.74%)	0.0%	0.0%
SRTS Routing	3901 (-2.62%)	3880 (-3.15%)	494.79s (+0.64%)	504.67s (+2.65%)	0.0%	0.0%
Medium Traffic (generation/20s)						
No Routing	1985	1985	491.73s	491.73s	11.53%	34.92%
Baseline T-S Routing	1937 (-2.42%)	1930 (-2.77%)	494.86s (+0.64%)	503.52s (+2.40%)	0.0%	0.0%
SRTS Routing	1940 (-2.27%)	1934 (-2.57%)	494.74s (+0.61%)	503.76s (+2.45%)	0.0%	0.0%
Light Traffic (generation/30s)						
No Routing	1315	1315	490.82s	490.82s	7.96%	31.8%
Baseline T-S Routing	1288 (-2.05%)	1286 (-2.21%)	493.72s (+0.59%)	502.45s (+2.37%)	0.0%	0.0%
SRTS Routing	1289 (-2.00%)	1288 (-2.05%)	493.85s (+0.62%)	502.32s (+2.34%)	0.0%	0.0%

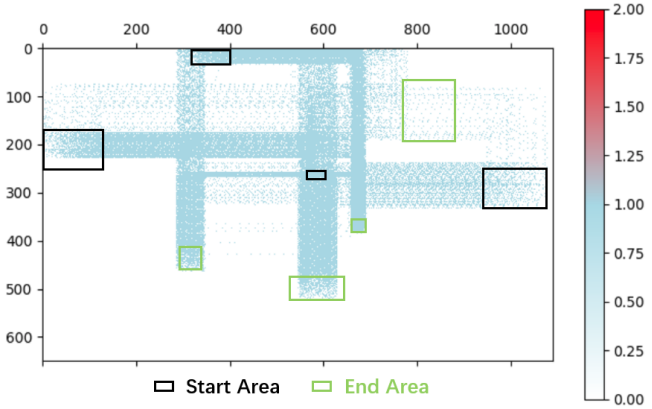


Fig. 8: sUAS Trajectory Density with Traffic Management (SRTS Routing)

TABLE II: The Comparison with Connectivity Check Algorithm

Traffic Type	Avg. In-flight UASs	Avg. Flight Time	No Link Rate
Heavy Traffic (generation/10s)			
SRTS Routing w/o. Conn. Check	95.95	491.46s	85.15%
SRTS Routing w. Conn. Check	70.03	517.82s	0.00%
Medium Traffic (generation/20s)			
SRTS Routing w/o. Conn. Check	49.28	491.43s	43.48%
SRTS Routing w. Conn. Check	41.44	513.12s	0.00%
Light Traffic (generation/30s)			
SRTS Routing w/o. Conn. Check	32.44	491.44s	26.25%
SRTS Routing w. Conn. Check	28.66	512.82s	0.00%

From Fig. 6 we can see that, without routing, some bright red points (i.e. conflicts) exist around the center of map. From Fig. 7 and 8 we can see that applying the TS routing and SRTS routing algorithm fully eliminate the conflicts. The sUAS trajectory concentrates in the upper part of the map in Fig. 8, because the candidate next move selection in SRTS follows a fixed priority, where going west or east always has higher priority than going north or south if all other conditions are the same.

C. Communication Connectivity Improvement

In the second experiment, we demonstrate how SRTS routing can improve the connectivity of the sUAS with the cellular network. Using the log-distance path loss model given in Section III-E, the UAS will establish a communication link with a base station if the path loss is less than 140dB. Otherwise, the communication link cannot be established. Table II compares the routing results of the SRTS algorithm without consideration of connectivity (row 1) and with the consideration of connectivity (row 2). Fig. 10 and Fig. 9 show the sUAS trajectories with and without the connectivity check. The circles in the figure indicate the areas that are covered by a base station.

Our simulation results show that without checking the connectivity, in heavy traffic situation, the no link rate is 85.15%. This means 85.15% of sUAS will experience a certain period of time in its mission in which no cellular link can be established to communicate with the GCS. The no link rate reach 43.48% and 26.25% in medium and light traffic. Although the routing algorithm can plan a conflict free trajectory for those sUAS, some locations along the trajectory either do not have coverage from the cellular network (as shown in Fig. 9) or the available channels have been depleted due to congestions. By applying the connectivity check, the no link rate is reduced to 0%. From Fig. 10 we can see that the sUAS only fly in the areas which are covered by the base stations. We can also observe that with the connectivity check, the average number of sUASs in the air is decreased by 27%, and the average flight time of the sUAS is increased by 5%. Since the availability of channels in the environment is limited, a sUAS will not be launched if communication links cannot be established in the flight path. The percentage throughput reduction is less when the traffic becomes lighter.

We visualize the resource usage of each base station at some sampled time steps. And the time steps 500, 5000, 10000 and 198000 are chosen. Since the total simulation duration is 20000, the number of airbone UASs at time 5000 and 10000 are representative of the peak value for in-flight UAS in the whole simulation. Fig. 11 shows the distribution of available

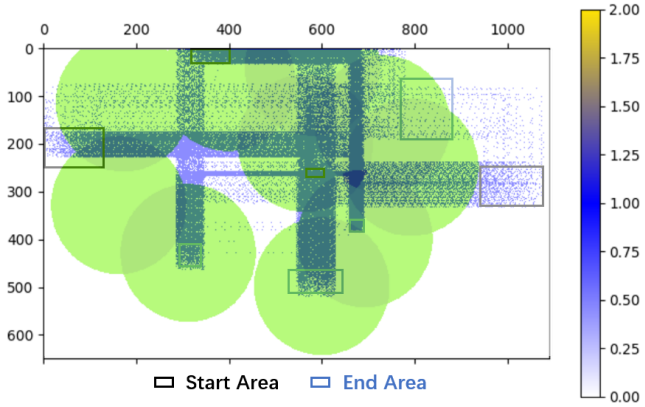


Fig. 9: Planned Trajectory without Connectivity Check Algorithm

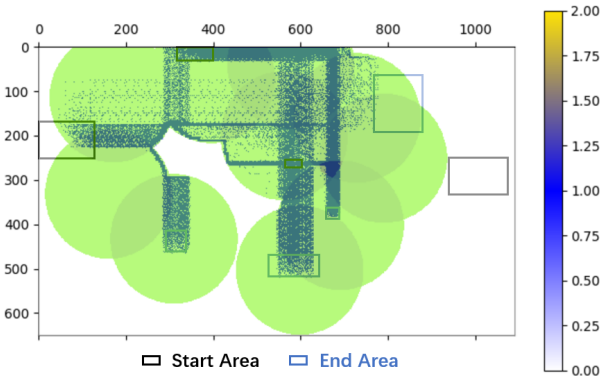
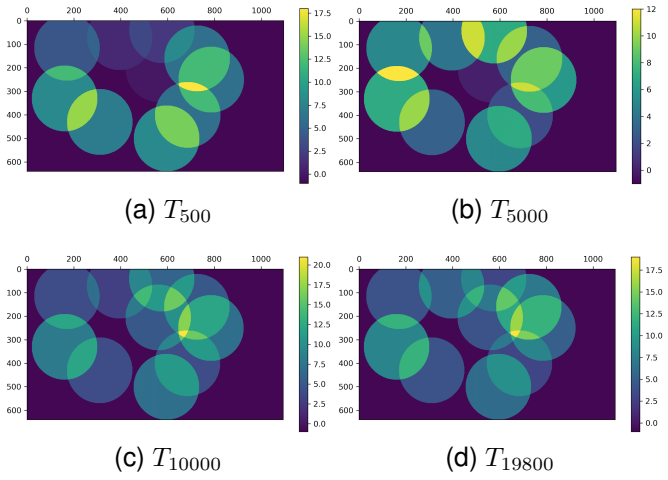


Fig. 10: Planned Trajectory with Connectivity Check Algorithm

communication channels at different time during the simulation. The sample time 500 and 19800 approach to the start time and the end time of the simulation, therefore, the number of in-flight UAS is sparse. The bright yellow represents the available channels are sufficient at that time. However, the deep blue stands for the area where all the communication channels are occupied.

The distribution of available channels in Fig. 11 are collected from the simulation, hence they represent the "ground truth" information of available communication resources. Based on our observation, the inner belief of the communication resource distribution during the routing stage is very close to the ground truth. Due to the space limit, we do not plot them here, however, they look just the same as Fig. 11. The similarity is expected because the connections between sUAS and the base stations has 90% of chance to be line-of-sight as mentioned in section III-E. This means the path-loss is mainly a function of the distance between the sUAS and the base station, and is highly predictable.



(a) T_{500} (b) T_{5000}

(c) T_{10000} (d) T_{19800}

Fig. 11: Distribution of available cellular channels (Ground Truth)

D. The Resource Usage of the Routing Algorithm

In the last experiment, we compare the memory usage and computing time of different routing algorithms. Almost all the routing algorithms need to store the environment information of a vast airspace, therefore, there may be a high demand for memory storage during the runtime. We also analyze the time to compute a route for each sUAS. This computation must finish in a very short amount of time so that the launch of the sUAS will not be delayed. Hence, we record the average memory usage and the average routing time during our simulation. The comparison between T-S routing and SRTS routing is given in Table III.

From Table III we can see that compared to T-S routing, the SRTS routing method reduces memory usage by more than 70%. With the help of the instant refreshing mechanism, only the present and future obstacle information will be preserved. And the history obstacle information will been removed automatically. Hence, when increasing from light traffic to medium and heavy traffic, the memory demand of SRTS only increases 2% and 6% respectively for scenarios with no-fly zone constraint, and 2% and 4% respectively for scenarios without no-fly zone constraint. While the T-S routing's memory demand increases 11% and 21% respectively for scenarios with no-fly zone constraint, and 10% and 18% respectively for scenarios without no-fly zone constraint. These number show that SRTS routing is much more scalable than T-S routing in terms of storage complexity. Moreover, the results indicate that including geographical constraints (i.e. no-fly zones) is not a heavy burden for our routing algorithm. Compared with the simulation scenario without the trajectory management, even with 2 no-fly zones, the memory usage of our proposed algorithm increases only about 10.5%, 7.5% and 5.5% in the cases of heavy, medium and light traffics, respectively.

Moreover, the results from Table IV show that, by using

TABLE III: Routing Algorithm Memory Usage Comparison

Traffic Type	Heavy Traffic (generation/10s)		Medium Traffic (generation/20s)		Light Traffic (generation/30s)	
	0 No-Fly Zone	2 No-Fly Zones	0 No-Fly Zone	2 No-Fly Zones	0 No-Fly Zone	2 No-Fly Zones
No Routing	675MB±2.2%	675MB±2.2%	667MB±2.1%	667MB±2.1%	666MB±1.5%	666MB±1.5%
Baseline T-S Routing	3513MB±3.7%	3608MB±3.3%	3257MB±3.1%	3315MB±2.9%	2963MB±3.8%	2974MB±3.5%
SRTS Routing	725MB±3.4%	746MB±2.3%	713MB±3.8%	717MB±2.7%	698MB±3.2%	703MB±3.0%

TABLE IV: Routing Algorithm Running time Comparison

Traffic Type	Heavy Traffic (generation/10s)		Medium Traffic (generation/20s)		Light Traffic (generation/30s)	
	0 No-Fly Zone	2 No-Fly Zones	0 No-Fly Zone	2 No-Fly Zones	0 No-Fly Zone	2 No-Fly Zones
Baseline T-S Routing	2.33ms	3.20ms	1.81ms	2.27ms	1.49ms	1.86ms
SRTS Routing	0.37ms (-84.12%)	0.49ms (-84.69%)	0.35ms (-80.66%)	0.42ms (-81.50%)	0.34ms (-77.18%)	0.40ms (-78.49%)

our SRTS routing algorithm, the UAS route planning time can be significantly reduced. Compared with the original TS routing algorithm, we can achieve 84.69%, 81.50% and 78.49% planning time reduction in the scenarios of high, medium and light traffics, respectively.

In addition, we evaluate the routing time of the SRTS routing algorithm in different airspace conditions, and the results are displayed in Fig. 12. The X-axis represents the average number of UAS in-the-air when the trajectory is planned and the Y-axis stands for the average planning time in milliseconds. The blue line and the red line are the time consumption of the approved launching and the failed launching. First, the experiment results show that the declined launching always take longer planning time than approved launching. This is because the system needs to try all the possible movements until there are no options to choose, then it will decline a launch request. For the approved launching, after one trajectory is found, the other possible movements will not be considered. Second, the results reveal that our proposed routing algorithm has a linear time complexity to the length of trajectory that was found. The average planning time is highly correlated to the average trajectory length. That explains why in the same simulation environment, the planning time will increase marginally when there are more UASs in the air. Again, the results show that the SRTS algorithm is scalable to the congestion level of the airspace.

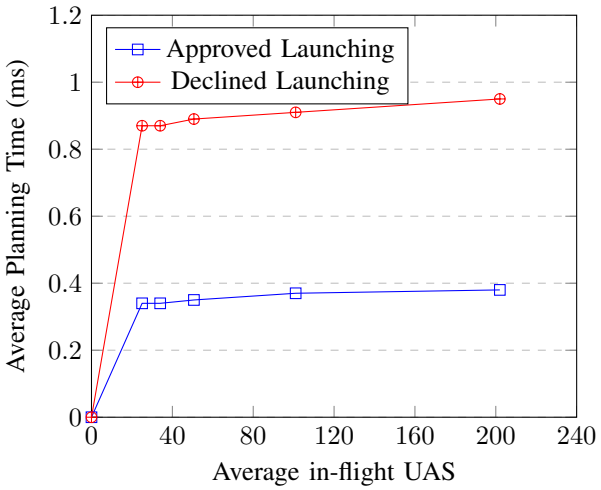


Fig. 12: Routing Time in Various Airspace Conditions

V. Conclusions

In this paper, we have proposed a new sparse represented temporal spatial routing algorithm for Unmanned Aircraft Systems (UAS) traffic management. The proposed algorithm allows the sUAS to avoid static no-fly areas (i.e. static obstacles) or other in-flight sUAS and areas that have contested communication resources (i.e. dynamic obstacles). The core functionality of the routing algorithm supports the instant refresh of the in-flight environment making it appropriate for highly dynamic air traffic scenarios. In addition, our characterization of the routing time and memory usage demonstrate that our algorithm outperforms a traditional T-S routing algorithm. Finally, the results have shown that the proposed algorithm has the ability to evaluate different sUAS traffic management policies. Moreover, the SRTS routing algorithm can be easily integrated with other simulation tools for further study.

References

[1] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Uavs for smart cities: Opportunities and challenges," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 267–273, IEEE, 2014.

[2] Z. Zhao, C. Luo, Z. Jin, F. Basti, M. C. Gursoy, C. Caicedo, and Q. Qiu, "A simulation framework for fast design space exploration of unmanned air system traffic management policies," *arXiv preprint arXiv:1902.04035*, 2019.

[3] A. Puri, "A survey of unmanned aerial vehicles (uav) for traffic surveillance," *Department of computer science and engineering, University of South Florida*, pp. 1–29, 2005.

[4] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson, "Unmanned aircraft system traffic management (utm) concept of operations," 2016.

[5] S. Sastry, G. Meyer, C. Tomlin, J. Lygeros, D. Godbole, and G. Pappas, "Hybrid control in air traffic management systems," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, vol. 2, pp. 1478–1483, IEEE, 1995.

[6] B. Albaker and N. Rahim, "A survey of collision avoidance approaches for unmanned aerial vehicles," in *2009 International Conference for Technical Postgraduates (TECHPOS)*, pp. 1–7, IEEE, 2009.

[7] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[8] S. A. Bortoff, "Path planning for uavs," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 1, pp. 364–368, IEEE, 2000.

[9] J. Tisdale, Z. Kim, and J. K. Hedrick, "Autonomous uav path planning and estimation," *IEEE Robotics & Automation Magazine*, vol. 16, no. 2, pp. 35–42, 2009.

[10] M. Odelga, P. Stegagno, and H. H. Bülthoff, "Obstacle detection, tracking and avoidance for a teleoperated uav," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 2984–2990, IEEE, 2016.

[11] H. L. N. N. Thanh and S. K. Hong, "Completion of collision avoidance control algorithm for multicopters based on geometrical constraints," *IEEE Access*, vol. 6, pp. 27111–27126, 2018.
[12] C. Wang, W. Liu, and M. Q.-H. Meng, "Obstacle avoidance for quadrotor using improved method based on optical flow," in *2015 IEEE International Conference on Information and Automation*, pp. 1674–1679, IEEE, 2015.
[13] Y. Li, H. Eslamiat, N. Wang, Z. Zhao, A. K. Sanyal, and Q. Qiu, "Autonomous waypoints planning and trajectory generation for multi-rotor uavs," *Proceedings of Design Automation for CPS and IoT*, 2019.
[14] H. Eslamiat, Y. Li, N. Wang, A. K. Sanyal, and Q. Qiu, "Autonomous waypoint planning, optimal trajectory generation and nonlinear tracking control for multi-rotor uavs,"
[15] R. W. Beard and T. W. McLain, "Multiple uav cooperative search under collision avoidance and limited range communication constraints," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 1, pp. 25–30, IEEE, 2003.
[16] B. D. Song, J. Kim, and J. R. Morrison, "Rolling horizon path planning of an autonomous system of uavs for persistent cooperative service: Milp formulation and efficient heuristics," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 241–258, 2016.
[17] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *Departmental Papers (ESE)*, p. 323, 1992.
[18] D.-S. Jang, C. A. Ippolito, S. Sankararaman, and V. Stepanyan, "Concepts of airspace structures and system analysis for uas traffic flows for urban areas," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, p. 0449, 2017.
[19] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, pp. 9–18, 2005.
[20] M. J. North, E. Tatara, N. T. Collier, J. Ozik, *et al.*, "Visual agent-based model development with repast simphony," tech. rep., Tech. rep., Argonne National Laboratory, 2007.
[21] P. H. Kopardekar, "Unmanned aerial system (uas) traffic management (utm): Enabling low-altitude airspace and uas operations," 2014.
[22] R. A. Clothier, B. P. Williams, and N. L. Fulton, "Structuring the safety case for unmanned aircraft system operations in non-segregated airspace," *Safety science*, vol. 79, pp. 213–228, 2015.
[23] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
[24] B.-C. Seet, G. Liu, B.-S. Lee, C.-H. Foh, K.-J. Wong, and K.-K. Lee, "A-star: A mobile ad hoc routing strategy for metropolis vehicular communications," in *International Conference on Research in Networking*, pp. 989–999, Springer, 2004.
[25] N. J. Nilsson, *The quest for artificial intelligence*. Cambridge University Press, 2009.
[26] S. Edelkamp, S. Jabbar, and A. Lluch-Lafuente, "Cost-algebraic heuristic search," in *AAAI*, pp. 1362–1367, 2005.