2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), 3-7 October 2021, San Antonio, Texas, USA
DOI: 10.1109/DASC52595.2021.9594424

An RRT* Based Method for Dynamic Mission Balancing for Urban Air Mobility Under Uncertain Operational Conditions

Junlin Lou
School of Aerospace, Transport and Manufacturing
Cranfield University
Bedford, UK
junlin.lou@cranfield.ac.uk

Burak Yuksek
School of Aerospace, Transport and Manufacturing
Cranfield University
Bedford, UK
burak.yuksek@cranfield.ac.uk

Gokhan Inalhan
School of Aerospace, Transport and Manufacturing
Cranfield University
Bedford, UK
inalhan@cranfield.ac.uk

Antonios Tsourdos
School of Aerospace, Transport and Manufacturing
Cranfield University
Bedford, UK
a.tsourdos@cranfield.ac.uk

***Abstract*—Urban air mobility provides an enabling technology towards on-demand and flexible operations for passenger and cargo transportation in metropolitan areas. Electric vertical-takeoff and landing (eVTOL) concept is a potential candidate for urban air mobility platform because of its lower carbon emissions, lower noise generations and potentially lower operational costs. However, such a transportation model is subject to numerous complicated environmental and urban design factors including buildings, dynamic obstacles and micro-weather patterns. In addition, communication, navigation and surveillance quality-of-service and availability would be affected on the overall system performance and resilience. Some social factors such as privacy, noise and visual pollution should also be considered to provide a seamless integration of the urban air mobility applications into the daily life. This paper describes an integrated RRT* based approach for designing and executing flight trajectories for urban airspace subject to operating constraints, mission constraints, and environmental conditions. The generated path is energy-efficient and enables aerial vehicle to perform mid-flight landing for battery changing or emergency situations. Moreover, this paper proposes another approach that allows on-the-fly path re-planning under dynamic constraints such as geofences or micro-weather patterns. As such, the approach also provides a method toward contingency operations such as emergency landing on the fly.**

***Index Terms*—Urban air mobility, RRT*, Energy-efficient path, Re-planning, Uncertain conditions**

I. Introduction

Urban air mobility (UAM) provides an enabling technology towards on-demand and flexible human and cargo transportation in metropolitan areas. Moreover, UAM can also play various roles in many specific fields such as carrying out medical treatments and rescue operations in areas where ground transportation is not possible and high-rise fire fighting missions. Also, the UAM vehicles can be used instead of the helicopter platforms for passenger and cargo transportation in offshore marine industry in a safer and cost-effective way [1]. However, when compared to numerous unmanned aerial systems (UAS) that are usually utilized in a specific geographical area, UAM applications will bring significant operational risks and safety challenges to daily operation of existing airspace system [2] [3] [4]. In addition, such a transportation model is subject to a considerable number of complicating environmental, urban design and social factors. The above-mentioned issues limit the working space of UAM, create static, dynamic or uncertain no-fly zones in the restricted air space, and leading to a complex operation environment. Beside of the collision avoidance requirement in the dense operation space, impact of wind gust and turbulence should be considered to provide safety and efficiency. In the urban environment, buildings produce eddy current which will lead to sudden change of wind speed and direction. This phenomenon, known as gust, may result in difficulties to maintain position, altitude and stability, and even directly may cause loss of control [5].

UAM operations are usually performed over residential and commercial areas, thus noise is obviously an important factor. A team of Airbus has made an assessment on public perception of UAM in Los Angeles, Mexico City, New Zealand and Switzerland. The result displayed that noise problem is mostly focused by respondents [6]. The further away from noise source, the lower the volume of the noise received [7]. Therefore, in addition to using quieter motors, it is also necessary to set a suitable minimum operating altitude of UAM. A higher minimum operating altitude have to be set around high-rise buildings or the surrounding space should be directly defined as the flight restricted area. Similar to noise, additional problems arise about visual pollution and privacy-related issues that may reduce the quality of life of urban residents. A research investigated the general perception of UAS by text-mining semantic analysis, discovering that the public may be annoyed by small flying vehicles because they can disrupt vision and produce shadows [8]. Due to the characteristics of UAM operating in airspace at relatively low altitude, it possibly accentuate annoyance over the proximity of the flights and the perceived privacy loss [9]. These problems may produce physical or psychological burden on people who live in the urban environment. It may also lead another social factors and creates worry about reduction in property values [10]. Each of these factors requires to define

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

a minimum separation distance between the UAM vehicles and buildings. Here, each separation forms a spatial envelope. By superimposing the spatial envelopes, a no-fly zone which has to be avoided in path planning can be created.

The UAM operations need support of communication, navigation and surveillance (CNS) systems. However, in the urban environment, buildings can block satellite from direct line of site to the GPS receiver, which may greatly reduce navigation quality of service (QoS), or even block the signal completely. An experiment has proved that in the urban environment, when GPS signal gets interference, the flight path of UAV will deviate significantly [11]. Moreover, some regions with specific atmospheric conditions, such as urban canyons, will lead to change of GPS positioning accuracy, which will cause the path to deviate possibly over 20 meters [12]. The path deviation may cause collision or flying into restricted space. Therefore, UAM vehicles should avoid flying into such regions during the missions.

For surveillance in low-altitude UAM operations, traditional radar is unreliable because the radar signal will also reflect off of the ground and objects on the ground, causing interference. In high-density environment, the frequency band of automatic dependent surveillance-broadcast (ADS-B) will possibly be over-saturated [13]. This problem can be alleviated by considering the near-saturation space of ADS-B as no-fly zone and trying to avoid it in path planning process. However, mature management and higher flight freedom need more advanced and complex CNS technology.

No-fly zone can be in both static and dynamic characteristics. Due to emergency situations such as fire fighting mission and severe weather conditions, a dynamic geofence may occur in the urban airspace and aerial platforms need to avoid it. For the aerial vehicles which experience emergency situations such as low CNS quality of service issues, micro-pattern weather, etc., can take emergency landing to a specific ground infrastructure as a risk avoidance strategy. Such kind of dedicated ground infrastructures are required for the UAM operations are commonly called *vertiports* in literature. Vertiports are responsible for the take-off, landing, maintenance and even repairing of the UAM vehicles. They can be placed on rooftops, on barges over sea, inside highway junctions and on top of existing ground-transport infrastructure [14].

Currently, electric vertical takeoff and landing aircraft (eVTOL) platform is a promising concept for UAM applications because of its lower carbon emissions, lower noise generations and potentially lower operational costs. Although in recent years, the breakthrough of distributed electric propulsion technology has provided technical support for electric aircraft, the energy density issue of lithium polymer battery still limits the cruise range of eVTOL [15]. Therefore, there are still some scenarios that eVTOL needs to land mid-flight at vertiport and change batteries since limited energy storage. For instance, some missions in mega-cities may exceed the designed cruise range of eVTOL, strong wind conditions may reduce battery performance and increase energy consumption, precipitation may decrease the flight performance of eVTOL platforms, icing may reduce thrust efficiency as it affects the propeller aerodynamics [2].

As a result of above-mentioned problems that may arise in the urban airspace, several additional factors should be considered to generate safe, cost-effective and feasible flight paths. First, noise abatement requirements should be defined in the urban airspace to reduce noise pollution. This can be done by defining a minimum separation distance and integrating the path planning algorithm. Second, GPS and communication signal-blocking areas in the urban airspace should be defined and this information should also be integrated into the path planning algorithm. Third, the algorithm should be able to handle with possible static and dynamic geofencies resulted in different effects. Fourth, energy consumption of the UAM platform should be considered in the path planning algorithm to provide energy requirements and battery replacement process. Last, the path planning algorithm should consider vertiport operations and should be able to account for mid-flight landing on these areas in an emergency situation and/or battery changing for the rest of the flight.

In this paper we propose an RRT* based algorithm that apply energy consumption function of rotary-wing eVTOL and provide mid-flight landing ability to able to produce energy-efficient and safe flight trajectories in the urban airspace. In this method, different from the RRT*, we developed variants of ChooseParent and Rewire algorithms to enable the planner to find the most suitable vertiport for mid-flight landing while generating energy-efficient path. Several simulation studies are performed for different environmental conditions to evaluate the system performance. According to these results, the proposed algorithm generates flight paths with higher path quality and less processing time when compared to the RRT*. In addition, we also propose an algorithm that reuses the tree data of RRT* to provide path rapid re-planning ability for avoiding dynamic geofence and/or conducting emergency landing, which is crucial for urban airspace applications.

The rest of paper is organized as follows; in Section II, we define the requirements of the developed path planning algorithm that need to be satisfied with according to the above problems. In Section III, we describe an RRT* based approach for generating energy-efficient path in various wind conditions, conforming to environmental and operational limitations, while finding the most suitable vertiport for mid-flight landing for battery changing or emergency situations. In Section IV, we propose algorithms based on the RRT* for rapid re-planning, which enables aerial vehicles to re-plan paths for dynamic geofences or emergency landing with quick response. In Section V, concluding remarks and future works are given.

II. Problem Statement

In order to ensure safety and efficiency of the UAM operations, it is critical to develop an effective method for planning the path that satisfies the operation limits of the eVTOL, and considers the urban environmental factors in three dimension space. The generated flight path needs to meet several conditions that are listed below;

- It needs to avoid all Geo-Fences.
- It should be feasible, i.e. it should be within the flight envelope and should be executable by the flight control system.
- It should be generated by considering the mid-flight landing locations and vertiports for replacing batteries.
- Path should be energy-efficient for both light and severe wind conditions. Both legs of the mission are obliged to be within energy limit.

When the operational constraints (i.e. energy status of the vehicle) or environmental conditions (i.e. severe wind in an urban canyon cannot satisfy the requirement of completing an ongoing mission, path should be redesigned by following this approach to conform the energy storage and operational limitations.

Moreover, in the urban environment, safety is the most important factor in the UAM operations which require re-planning capability of the flight path in a short time. For example, a dynamic geofencing may appear on the path and intersect it when the eVTOL has already been in a mission. In these situations, the re-planned path should be consistent with the original path as much as possible since urban air traffic controllers could provide sufficient guidance in the presence of conflicts during the operation. If this kind of conflicts occur, the usual solution is 'sense' and 'avoid'. However, such a solution requires a greedy algorithm to resolve imminent conflict, which is highly likely to reduce the airspace throughput and safety in traffic dense environment [16]. Additionally, when the vehicle has already been affected by some unexpected factors (i.e. microburst, GPS signal loss, etc.) and this situation results in a high safety risk, that leg of trajectory should be re-designed with re-planning method for enabling the vehicle to do emergency landing on vertiport as soon as possible.

Graph search algorithm and sampling based algorithm are candidates to plan a path while conforming to energy storage and operational limitations. Nevertheless, Graph search algorithm such as A* typically grids the space and then stores all generated grid nodes in memory, the spatial complexity of graph search algorithm has to be considered, since the operation environment needs to be three-dimensional [17].

Different from the usual path planning, problems given in this section need considering the energy status of UAM vehicles, battery changing when landing at a vertiport during the mission and selection of vertiport. RRT*, as a sampling-based algorithm, quickly explores the configuration space and manages the generated random samples in a tree structure [18]. Candidate paths obtained from the tree and the most qualified path for requirements of operations and energy limitation is finally chosen among them.

Therefore, the approach in this paper is developed based on the RRT* which is a sampling based algorithm. RRT* extends the tree and explores the configuration space by randomly-sampled points, extending a certain distance from the existing node to the sampling point and generating a new node if there is no collision with the obstacles. It characterizes its *ChooseParent* and *Rewire* operation. In the *ChooseParent* operation, it traverses the set of potential parent nodes that are inside a specific Euclidean distance of a child node, detecting collision, updating the tree with most qualified parent node. Then, in the *Rewire* operation, it iterates through nodes within neighbourhood area around new node. The new node becomes a parent of some nodes if no collision and cumulative cost reduction happens. It finally produces a cost-optimal path that meets all conditions defined in the mission requirements.

III. RRT* BASED ENERGY-EFFICIENT PATH PLANNING THROUGH VERTIPORT

In this section, we describes an RRT* based approach for generating energy-efficient path by replacing cost function of Euclidean distance with energy consumption cost function of rotary-wing eVTOL. This approach also enables aerial vehicle to find the most suitable vertiport for mid-flight landing for battery changing or emergency situations, through variants of *Chooseparent* and *Rewire* procedure. Moreover, paths under various wind conditions are generated in simulations.

A. Cost function definition

RRT* algorithm updates the tree and selects the final path by calculating and comparing the cumulative cost of nodes. In original RRT* algorithm, cost of each edge in the tree is calculated by using its Euclidean distance of vertexes. Here, as the purpose of this paper is to generate an energy-efficiency path for rotary-wing eVTOL vehicle, the cost function will be developed based on energy consumption of the aerial platform. Every edge in the tree generated by the RRT* algorithm can be considered as flight phases of forward flight, forward flight - climbing or forward flight - descending. When the aircraft is at nominal flight speed (i.e. 25 knots to 30 knots), the required power is approximately calculated as the linear superposition of the required power for the horizontal flight and vertical climbing or descending [19].Therefore, Eq. (1) can be obtained [19] [20] [21]:

$$E = P\,t \approx \left(\left(\sum_{m=1}^{n} (P_i + P_o)_m + P_p \right)_{V_{c/d}=0} + \sum_{m=1}^{n} (P_{c/d})_m \right) t \quad (1)$$

where E is energy, P_i is induced power, P_o is rotor profile power, P_p is parasite power, $P_{c/d}$ is power term of climbing or descending, $V_{c/d}$ is vertical velocity of forward flying with climbing or descending, n is number of rotors, t is time spending for flying from a vertex of an edge to another vertex.

Then, Eq.(2) can be obtained by expanding each power term concretely, the induced power is mainly derived from momentum theory and Bernoulli equation [19] [20] [21]:

$$E \approx \left(n\Big(\frac{kT_f^2}{2\rho A V_f B} + \frac{1}{8} C_{do} N c R (1 + 4.6\mu^2) \rho (\omega R)^3 \Big) + \frac{1}{2} f \rho V_f^3 + \frac{n T_f V_{c/d}}{\tan\alpha} \right) t \quad (2)$$

where k is non-uniform flow correction, T_f is forward thrust, ρ is air density, A is area of rotor disk, V_f is forward velocity, B is tip loss factor, C_{do} is drag coefficient, N is number of

blades, c is blade chord, R is rotor radius, μ is rotor advance ratio, ω is rotor rotation speed, f is equivalent flat-plate drag area, α is angle of attack.

The wind factor in operating environment will have an impact on the actual energy consumption as it directly affects thrust and velocity. As a result, the actual values of thrust and speed are different from those used to estimate energy consumption. Values for energy consumption estimation and the actual values has a ratio, which is called Air/Ground ratio in this paper. When there is uniform and constant X-axis positive wind in the environment, Air/Ground ratio λ is deduced as follows in Eq.(3)(4)(5)(6):

$$V_{ground}^2 = V_x^2 + V_y^2 \quad T_{ground} = T_f \quad V_{ground} = V_f \tag{3}$$

$$\begin{aligned} V_{air}^2 &= (V_x - V_{wind})^2 + V_y^2 \\ &= V_x^2 - 2V_x V_{wind} + V_{wind}^2 + V_y^2 \\ &= V_{ground}^2 - 2V_x V_{wind} + V_{wind}^2 \\ &= V_{ground}^2 - 2\cos\theta V_{ground} V_{wind} + V_{wind}^2 \end{aligned} \tag{4}$$

$$\left(\frac{V_{air}}{V_{ground}}\right)^2 = \left(\frac{V_{wind}}{V_{ground}}\right)^2 - 2\cos\theta\left(\frac{V_{wind}}{V_{ground}}\right) + 1 \tag{5}$$

$$\lambda = \frac{V_{air}}{V_f} = \frac{T_{air}}{T_f} = \sqrt{(\frac{V_{wind}}{V_f})^2 - 2\cos\theta(\frac{V_{wind}}{V_{ground}}) + 1} \tag{6}$$

where V_{ground} is ground speed, T_{ground} is resultant thrust, V_{air} is air speed, V_{wind} is wind velocity, θ is Angle between mapping of vehicle flying direction on horizontal plane and positive direction of x-axis.

Substituting Eq.(6) into Eq.(2):

$$\begin{aligned} E \approx \Bigg(n\Big(\frac{\lambda^2 k T_f^2}{2\lambda\rho A V_f B} + \frac{1}{8}C_{do}NcR(1+4.6\mu^2)\rho(\omega R)^3\Big) \\ + \frac{1}{2}f\rho V_f^3 + \frac{n\lambda T_f V_{c/d}}{\tan\alpha_{windy}}\Bigg)t \end{aligned} \tag{7}$$

The horizontal wind has little direct effect on vertical motion of aircraft. When aircraft enters wind environment, it can maintain the original motion in vertical direction by adjusting angle of attack. Therefore, vertical component of thrust under wind condition can be considered to be equal to that in absence of wind:

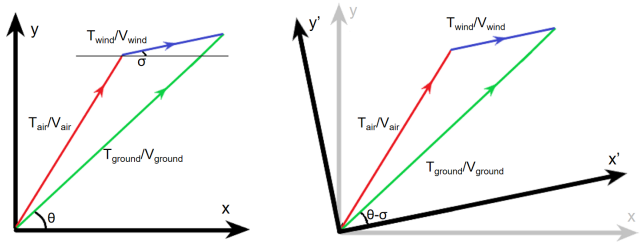
$$\frac{\lambda T_f}{\tan\alpha_{windy}} = \frac{T_f}{\tan\alpha} \tag{8}$$

where α_{windy} is angle of attack under windy environment condition.

By substituting Eq.(8) into Eq.(7), the energy consumption function under wind condition can be obtained:

$$\begin{aligned} E_{windy} \approx \Bigg(n\Big(\frac{\lambda k T_f^2}{2\rho A V_f B} + \frac{1}{8}C_{do}NcR(1+4.6\mu^2)\rho(\omega R)^3\Big) \\ + \frac{1}{2}f\rho V_f^3 + \frac{nT_f V_{c/d}}{\tan\alpha}\Bigg)t \end{aligned} \tag{9}$$

Energy consumption under various wind angles can be easily estimated by further derivation of Air/Ground ratio λ.



(a) Wind with different angles (b) Equivalent wind correction

Fig. 1. Correction for wind with different angles

When the angle between direction of wind and positive X-axis is σ, the scenario can be equivalent to that angle between V_{ground} and positive X-axis is $\theta - \sigma$ under positive X-axis wind condition. Through further derivation, Air/Ground ratio is:

$$\lambda = \frac{V_{air}}{V_f} = \frac{T_{air}}{T_f} = \sqrt{\left(\frac{V_{wind}}{V_f}\right)^2 - 2\cos(\theta-\sigma)\left(\frac{V_{wind}}{V_{ground}}\right) + 1} \tag{10}$$

Time-varying wind environment can also be created:

$$V_{wind} = a\,\sin(b\,t_{cu}) + l \tag{11}$$

$$\sigma = \gamma\, t_{cu} \tag{12}$$

$$t_{cu} = \frac{l_{cu}}{V_f} \tag{13}$$

where t_{cu} is cumulative time spending, σ is angle between wind direction and positive x-axis, l_{cu} is cumulative path length, parameters a, b, l and γ are constants. The coordinate axes x', y' are obtained by rotating axes x, y anticlockwise around the origin by σ degrees.

After substituting Eq.(12) into Eq.(10), and substituting Eq.(10) and Eq.(11) into Eq.(9), the energy estimation cost function under time-varying wind conditions can be obtained.

B. Algorithm design

In this section, an RRT* based algorithm for generation of energy-efficient flight path through several vertiports is proposed. This algorithm is able to find a feasible and energy-efficient flight path for the UAM vehicle while considering landing on vertiports and changing battery in mid-flight. In the configuration space with multiple vertiports, original RRT* algorithm needs to be executed many times to get the global optimal solution while the proposed method only needs to be executed once. Original RRT* algorithm explores the environment through RRT method, updating connections between nodes through *Chooseparent* and *Rewire* operations so as to make the cumulative path length of each node as short as possible. The core idea of the proposed algorithm is trying to make each node in the tree connect to one of vertiports instead of minimizing the cumulative path length. This means all tree branches tend to have connections with one of vertiports. Based on this, the cumulative energy consumption of each node is minimized as much as possible.

Algorithm 1 Energy-Efficient path through a Vertiport based on RRT*

$V \leftarrow x_{init}$; $E \leftarrow \emptyset$;
$verti_k \in Vertiports$
$PassVerti(Verti_k) \leftarrow k$
for $i \leftarrow 1 : m$ **do**
  for $k \leftarrow 1 : size(Vertiports)$ **do**
    $Samplingbias(verti_k)$
  end for
  if $rand < biasprob$ **then**
    $x_{simp} \leftarrow Samplingbias_i$
  else
    $x_{simp} \leftarrow SamplingFree_i$
  end if
  $x_{nearest} \leftarrow Nearest(G = (V, E), x_{simp})$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{simp})$
  if $ObstacleFree(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow Neighbour(G = (V, E), x_{new})$
    $PassVerti(x_{new}) \leftarrow 0$
    $V \leftarrow V \cup x_{new}$
    $x_{min} \leftarrow x_{nearest}$
    $c_{min} \leftarrow Cost(x_{init}, x_{nearest}) + Cost(x_{nearest}, x_{new})$
    for *each* $x_{near} \subset X_{near}$ **do**
      if $CollisionFree(x_{near}, x_{new}) \wedge$
$Cost(x_{init}, x_{near}) + Cost(x_{near}, x_{new}) < c_{min}$ **then**
        $x_{min} \leftarrow x_{near}$
        $c_{min} \leftarrow Cost(x_{init}, x_{near}) + Cost(x_{near}, x_{new})$
      end if
    end for
    $E \leftarrow E \cup (x_{min}, x_{new})$
    for *each* $x_{near} \subset X_{near}$ **do**
      if $CollisionFree(x_{new}, x_{near}) \wedge Cost(x_{init}, x_{new})$
$+Cost(x_{new}, x_{near}) < Cost(x_{init}, x_{near})$ **then**
        $E \setminus (Parent(x_{near}), x_{near})$
        $Parent(x_{near}) \leftarrow x_{new}$
        $Cost(x_{init}, x_{near}) \leftarrow Cost(x_{init}, x_{new})$
$+Cost(x_{new}, x_{near})$
        $E \leftarrow E \cup (x_{new}, x_{near})$
        Propogate Cost Changes to Leaves
      end if
    end for
  end if
return $G = (V, E)$
end for
Save paths to Vertiports
Generate path to Goal

The proposed algorithm is given in Algorithm 1. Here, it is executed until m^{th} sampling. It is similar to the original RRT* in general. However, compared with original RRT*, the proposed algorithm has sampling bias around vertiports, and gives a new attribute $PassVerti$ to nodes on the tree, which makes the matrix of tree data have one more column. After m^{th} sampling, algorithm 2 and algorithm 3 will be executed.

For clarity of the proposed method, terminology used throughout this algorithm is defined as follows;

- $SamplingFree_i$: A procedure returns a new state x_{simp}, which is generated randomly.
- $Samplingbias_i$: A procedure returns a new state x_{simp}, which is near one of vertiports.
- $Nearest$: A procedure returns the nearest node of x_{simp} in tree.
- $Steer$: A procedure returns a new node x_{new} steering from $x_{nearest}$ to x_{simp} at a certain distance.
- $ObstacleFree$: A procedure checks if there is collision in steering or not.
- $Neighbour$: A procedure returns nodes in tree whose Euclidean distance to x_{new} is less than a predefined value.
- $PassVerti$: A label used for marking a node whether it is on a branch connected with a vertiport and defining which vertiport is connected.
- $CollisionFree$: A procedure tests if collision occurs on the straight-line segment between two nodes.
- $Propogate\ Cost\ Changes\ to\ Leaves$: A procedure iterates through children, then recurse to update their costs until no any child found in tree data.
- $Cost$: A procedure returns the cumulative energy cost from initial node, energy estimation of each edge is based on energy consumption function in section A. Some variables in the function are derived from edges and vertexes during planning process. Assume:
  - *Angle of attack and parasite power are constant*
  - *Vehicle is flying at nominal and constant forward speed with constant magnitude of forward thrust component*
  - *Power required for steering vehicle is ignored.*

In energy consumption cost function, there are some variables that can only be known in *ChooseParent* and *Rewire* operations. These two procedures require to calculate energy consumption cost through coordinates of x_{near} and x_{new} many times. At this time, these variables can be derived by Eq.(14)(15)(16)(17):

$$t = \frac{\sqrt{\Delta x^2 + \Delta y^2}}{V_f} \tag{14}$$

$$V_{c/d} = Vf \tan\beta = V_f \sqrt{\frac{\Delta z^2}{\Delta x^2 + \Delta y^2}} \tag{15}$$

$$\theta = \arccos \sqrt{\frac{\Delta x^2}{\Delta x^2 + \Delta y^2}} \tag{16}$$

$$\omega = (1 + 0.03 V_{c/d})\omega_0 \tag{17}$$

where Δx, Δy and Δz are coordinates differences of a pair of nodes in X, Y, Z axes, β is angle between vehicle flight direction and horizontal plane, ω_0 is rotor rotation speed at level forward flight with uniform and constant forward velocity.

After executing Algorithm 1, every node in the configuration space can be considered as having optimal connection. Due to $Samplingbias_i$ procedure, there are many nodes near vertiports. Energy efficient flight path from initial node to each vertiport can be obtained by executing the Algorithm 2, and then cumulative cost and path information are saved for subsequent planning and final path generation. This step is necessary because subsequent algorithms will destroy the structure of existing tree generated by Algorithm 1.

Algorithm 2 Save paths to Vertiports

for $k \leftarrow 1 : size(Vertiports)$ **do**
  $X_{near} \leftarrow Neighbour(Verti_k)$
  $Parent(Verti_k) \leftarrow \emptyset$; $Cost(x_{init}, Verti_k) \leftarrow \infty$
  for *each* $x_{near} \subset X_{near}$ **do**
    if $CollisionFree(x_{near}, Verti_k) \wedge$
$Cost(x_{init}, x_{near}) + Cost(x_{near}, Verti_k)$
$< Cost(x_{init}, Verti_k)$ **then**
      $Parent(Verti_k) \leftarrow x_{near}$
      $Cost(x_{init}, Verti_k)$
$\leftarrow Cost(x_{init}, x_{near}) + Cost(x_{near}, Verti_k)$
    end if
  end for
  Save $Cost(x_{init}, Verti_k)$
  Backward Recurse to Generate $Path(x_{init}, Verti_k)$
  Save $Path(x_{init}, Verti_k)$
end for

Algorithm 3 is executed between the $(m+1)^{th}$ and n^{th} sampling. *ChooseParent* and *Rewire* operations in Algorithm 3 are significantly different from the original RRT* structure. After n^{th} iteration, Algorithm 3 generates the energy-efficient path between the final selected vertiport and goal position through backward recursion, and then generates the final path by combining with the path which is saved in the Algorithm 2. Both two legs of final path have to pass the energy check and meet the defined energy limit requirements. The planned flight path finally fits with second order B-spline, which enables the UAM vehicle to follow the path in real application.

Algorithm 3 Generate path to Goal

for $i \leftarrow m+1 : n$ **do**
  for $k \leftarrow 1 : size(Vertiports)$ **do**
    $Samplingbias(verti_k)$
  end for
  if $rand < biasprob$ **then**
    $x_{simp} \leftarrow Samplingbias_i$
  else
    $x_{simp} \leftarrow SamplingFree_i$
  end if
  $x_{nearest} \leftarrow Nearest(G = (V, E), x_{simp})$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{simp})$
  if $ObstacleFree(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow Neighbour(G = (V, E), x_{new})$
    $PassVerti(x_{new}) \leftarrow 0$
    $V \leftarrow V \cup x_{new}$
    $x_{min} \leftarrow x_{nearest}$
    $c_{min} \leftarrow Cost(x_{init}, x_{nearest}) + Cost(x_{nearest}, x_{new})$
    ChooseParent Variant
    Rewire Variant
  end if
return $G = (V, E)$
end for
$X_{near} \leftarrow Neighbour(G = (V, E), x_{goal})$
$Parent(x_{goal}) \leftarrow \emptyset$; $Cost(x_{init}, x_{goal}) \leftarrow \infty$
for *each* $x_{near} \subset X_{near}$ **do**
  if $CollisionFree(x_{near}, x_{goal}) \wedge PassVerti(x_{near}) > 0$
$\wedge Cost(x_{init}, x_{near}) + Cost(x_{near}, x_{goal}) < Cost(x_{init}, x_{goal})$
$\wedge EnergyCheck = TRUE$ **then**
    $Parent(x_{goal}) \leftarrow x_{near}$
    $Cost(x_{init}, x_{goal}) \leftarrow Cost(x_{init}, x_{near})$
$+Cost(x_{near}, x_{goal})$
  end if
end for
Backward Recurse to Generate Final Path

Algorithm 4 ChooseParent Variant

if $CollisionFree(Verti_k, x_{new}) \wedge$
$Distance(Verti_k, x_{new}) < Threshold$ **then**
  $Parent(x_{new}) \leftarrow Verti_k$
  $PassVerti(x_{new}) \leftarrow k$
  $Cost(x_{init}, x_{new}) \leftarrow Cost(x_{init}, Verti_k)$
$+Cost(Verti_k, x_{new})$
else
  for *each* $x_{near} \subset X_{near}$ **do**
    if $CollisionFree(x_{near}, x_{new})$
$\wedge PassVerti(x_{near}) > 0$ **then**
      if $PassVerti(x_{new}) = 0$ **then**
        $c_{min} \leftarrow \infty$
      end if
      if $Cost(x_{init}, x_{near}) + Cost(x_{near}, x_{new}) < c_{min}$ **then**
        $x_{min} \leftarrow x_{near}$
        $c_{min} \leftarrow Cost(x_{init}, x_{near}) + Cost(x_{near}, x_{new})$
        $PassVerti(x_{new}) \leftarrow PassVerti(x_{near})$
      else if $CollisionFree(x_{near}, x_{new})$
$\wedge PassVerti(x_{near}) = 0 \wedge Cost(x_{init}, x_{near})$
$+Cost(x_{near}, x_{new}) < c_{min}$ **then**
        $x_{min} \leftarrow x_{near}$
        $c_{min} \leftarrow Cost(x_{init}, x_{near}) + Cost(x_{near}, x_{new})$
      end if
    end if
  end for
  $E \leftarrow E \cup (x_{min}, x_{new})$
end if

Algorithm 5 Rewire Variant

for *each* $x_{near} \subset X_{near}$ **do**
  if $CollisionFree(x_{new}, x_{near}) \wedge PassVerti(x_{new}) > 0$
$\wedge x_{near} \not\subset Vertiports$ **then**
    if $PassVerti(x_{near}) = 0$ **then**
      $Cost(x_{init}, x_{near}) \leftarrow \infty$
    end if
    if $Cost(x_{init}, x_{new}) + Cost(x_{new}, x_{near})$
$< Cost(x_{init}, x_{near})$ **then**
      $E \setminus (Parent(x_{near}), x_{near})$
      $Parent(x_{near}) \leftarrow x_{new}$
      $Cost(x_{init}, x_{near}) \leftarrow Cost(x_{init}, x_{new})$
$+Cost(x_{new}, x_{near})$
      $E \leftarrow E \cup (x_{new}, x_{near})$
      $PassVerti(x_{near}) \leftarrow PassVerti(x_{new})$
      Propogate Cost and PassVerti Changes to Leaves
    end if
  else if $CollisionFree(x_{new}, x_{near}) \wedge PassVerti(x_{near}) = 0$
$\wedge PassVerti(x_{new}) = 0 \wedge Cost(x_{init}, x_{near}) + Cost(x_{new}, x_{near})$
$< Cost(x_{init}, x_{new}) \wedge x_{near} \not\subset Vertiports$ **then**
    $E \setminus (Parent(x_{near}), x_{near})$
    $Parent(x_{near}) \leftarrow x_{new}$
    $Cost(x_{init}, x_{near}) \leftarrow Cost(x_{init}, x_{new})$
$+Cost(x_{new}, x_{near})$
    $E \leftarrow E \cup (x_{new}, x_{near})$
    Propogate Cost Changes to Leaves
  end if
end for

Algorithm 4 is a variant of *ChooseParent* operation. The logic is that if x_{new} is a node of near one of vertiports, the vertiport will be set as its parent node, labeling that the tree branch that x_{new} locates is connected with a vertiport. Then cumulative cost of x_{new} is replaced by sum of cost of vertiport saved by Algorithm 2 and the energy consumption from vertiport to x_{new}. If x_{near} has connection with a vertiport, but x_{new} does not, cost of x_{new} is set as infinite large to make sure x_{near} can be parent of x_{new}. If both of x_{near} and x_{new} have already been through a vertiport or have not yet

been through a vertiport, Algorithm 4 updates connection for minimizing the cumulative energy cost. When parent node of any node is updated, $Passverti$ attribute of the node will be updated.

Algorithm 5 is a variant of *Rewire* operation. It will inevitably update connection as well as $Passverti$ attribute if $Passverti$ attribute of x_{new} is not 0 and $Passverti$ attribute of x_{near} is equal to 0. Otherwise it will not update connection. If both of x_{near} and x_{new} have already been through a vertiport or have not yet been through a vertiport, Algorithm 5 will update connection for minimizing cumulative energy cost as performed in the Algorithm 4. When an update occurs, $Passverti$ attribute is also updated.

C. Simulation and Results

In this section, simulation studies are performed to illustrate the effectiveness and feasibility of the proposed algorithm. Fig. 2 display environment in configuration space and Fig. 3 shows an example of planning. The white zones in Fig. 2 are start position, goal position and vertiports for mid-flight landing. Outside the white area, UAM vehicle has a minimum altitude limit. The yellow zones are obstacles and there are three kinds of height, among which the medium height and low height obstacles can be flied over from above. In reality, due to privacy problems, noise problems and other factors, the created superimposing clearance boundaries of obstacles is irregular. In this paper, this aspect is simplified, the obstacles are represented by cuboids. If further study this aspect, boundaries need to be described as mathematical expressions.

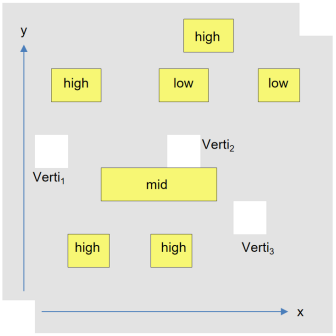


Fig. 2. Configuration space

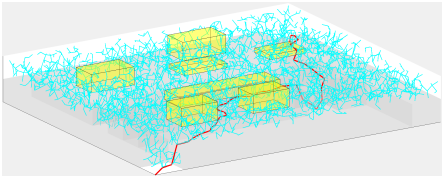


Fig. 3. An example of planning

Fig. 4 to Fig. 9 displays views of the path distribution. In Fig. 4, the cost function is Euclidean distance, path distributions in Fig. 5 is under no-wind environment condition, Fig. 6 shows weak uniform constant positive X-axis wind as wll as Fig. 7 shows strong uniform constant positive X-axis wind. There is time-varying wind with same parameters in Fig. 8 and 9, but the total number of iterations in simulations of these two figures are different, which are 5000 times in Fig. 8 and 30000 times in Fig. 9.

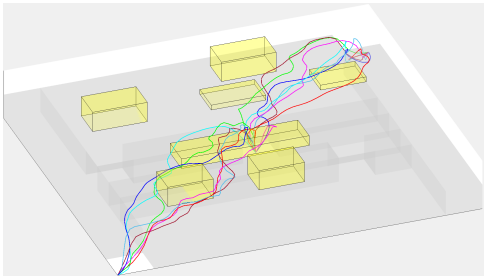


Fig. 4. Euclidean distance cost

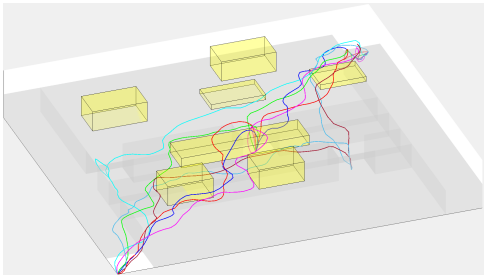


Fig. 5. No Wind

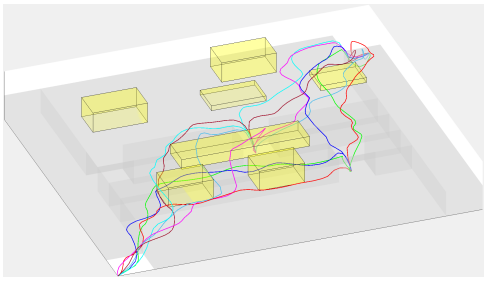


Fig. 6. Week positive X-axis wind

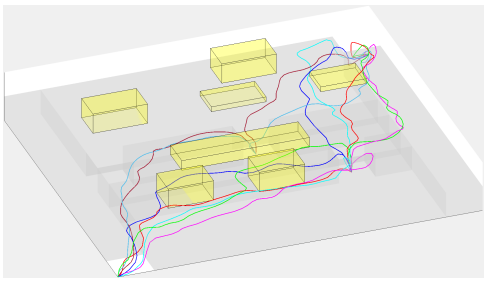


Fig. 7. Strong positive X-axis wind

It can be found in Fig. 4 and 5 that the energy efficient paths under windless condition tends to maintain at a lower altitude and has less motion in vertical direction when compared to paths that are planned by algorithm with Euclidean distance cost function. Fig. 6 and 7 display that, in this configuration space, when the wind direction is positive X-axis, stronger wind will lead to more obvious trend mentioned above. Moreover, it is difficult to infer the effect of time-varying wind to paths in Fig. 8. However, Fig. 9 illustrates when the predefined

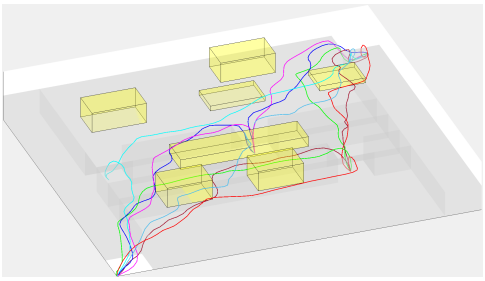


Fig. 8. Time-varying wind, 5k samples

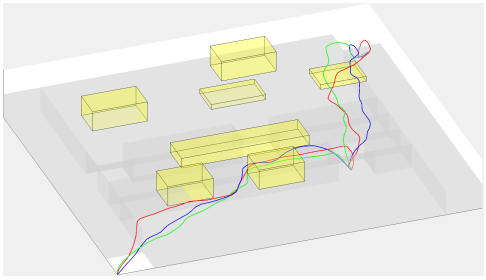


Fig. 9. Time-varying wind, 30k samples

parameter of maximum number of iterations in each simulation is sufficient and predefined parameters of wind stay constant, the resulting paths are similar and approximately optimal. Therefore, it means that in time-varying wind environment, there is still a common optimal path.

Fig. 10 shows the performance of the proposed algorithm. Each column in this box-plot has 25 sets of data. Column 1 shows cumulative costs of paths departing from start, landing mid-flight at $Verti_2$, taking off again and finally reaching goal by using RRT* with energy cost function. In each of 2 legs, the predefined parameter of maximum number of iterations of RRT* is 5000. Column 2 shows cumulative costs of paths generated by the proposed approach in this paper, with maximum total number of sampling nodes in the configuration space defined as 5000. In column 3, RRT* with energy cost function are utilized again for generating paths landing mid-flight at $Verti_2$ as well and cumulative costs are recorded. The difference is, here RRT*'s parameter of maximum number of iterations in each leg is defined as 3200. The processing time counted is normally 80 to 100 seconds in each simulation and it costs about 160 seconds to 200 seconds for a whole mission in column 3. The processing time in column 2 is 170 seconds to 200 seconds and it is close to time costs in column 3. The results show that path quality of the approach proposed in this paper is better than RRT* under the similar processing time condition. When the defined maximum number of iterations of RRT* in each leg is the same as the maximum total iterations that this paper's approach executes for a whole mission, path quality of this approach is slightly worse than RRT*, but compare with time spending of processing in column which is 320 to 350 seconds, the processing time of this approach is much shorter. Moreover, this approach has a unique advantage, it can obtain the most suitable vertiport for mid-flight landing while planning a path.

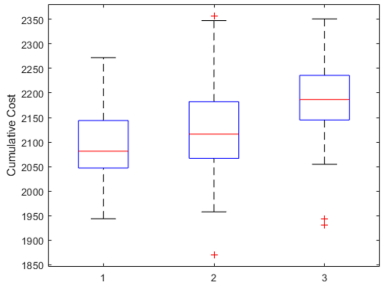


Fig. 10. Box-plot for path quality

IV. Rapid on-the-fly re-planning

In this section, we propose an approach that allows rapid on-the-fly re-planning under dynamic constraints by reusing tree data of RRT*. The approach also provides a method toward contingency operations such as emergency landing on the fly.

A. Algorithm Design

As a traffic mode operating in low airspace in urban environment, safety is the first consideration. When dynamic Geo-fence conflicting with subsequent path is founded or emergency landing is needed and if a new path cannot be planned in a short time, it will lead to high safety risk in the aerial operation. In addition, the generated avoidance path should not be much different from the original path so as to reduce the risk of disturbing the current air traffic and increasing the airspace safety and throughput.

Reusing the tree data of RRT* in the re-planning process is a potential solution for this problem. When the tree data already exists, a path from root to any node can be generated in extremely short time. However, the nodes in the RRT* only have the cumulative cost from the initial node, and all connections are optimized for the root of the tree. Therefore, for developing on-the-fly re-planning algorithm based on the existing tree data of RRT*, two problems need to be solved. Firstly, a method for moving the root of the tree to subsequent nodes needs to be found. Secondly, connections between nodes should be optimized for the new root.

Algorithm 6 implements the complete update of tree data. It firstly moves the root from initial node x_0 to its child node on the generated path. The specific operation of this step is shown in Fig. 11. The nodes within an appropriate Euclidean distance around the child node are considered as x_{near}. Algorithm 6 performs *Rewire* operation, traverses all near nodes, updates connections and cost for qualified nodes, then propagates the cost update to leaves by recursion method. The cost of root node must be updated. Thus, the cost of all nodes in the tree will be updated. After executing tree root movement once, root position can be moved to new root node's child node on the path by the same approach. Through several iterations, root can be moved to the desired node.

If *Chooseparent* and *Rewire* operations are utilized to update the connections of the global tree directly after moving the root position, it will take much longer than the time required to perform RRT* with same number of iterations once. The time required for the RRT* algorithm to complete an episode in

Algorithm 6 Tree Update

$G = (V, E) \leftarrow treedata$
$ConfigSpace = Lim_x \times Lim_y$
$X \leftarrow PathData$
$x_n \subset X$
$V_{simp} \leftarrow x_n$; $E_{simp} \leftarrow \emptyset$;
for $i = 1 : n$ **do**
  $X_{near} \leftarrow Neighbour(G = (V, E), x_i)$
  for *each* $x_{near} \subset X_{near}$ **do**
    if $CollisionFree(x_i, x_{near}) \wedge$
$Distance(x_i, x_{near}) \leqslant Cost(x_{i-1}, x_{near}) + Cost(x_{i-1}, x_i)$ **then**
      $Parent(x_{near} \leftarrow x_i$
      $Cost(x_{near}) \leftarrow Distance(x_i, x_{near})$
      Propogate Cost Changes to Leaves
    end if
  end for
end for
for $j = 1 : a$ **do**
  for $k = 1 : b$ **do**
    $A_{jk} \leftarrow Area[\frac{Lim_x}{j-1}, \frac{Lim_x}{j}; \frac{Lim_y}{k-1}, \frac{Lim_y}{k}]$
    Iterate though Node $\subset A_{jk}$
    Find x_{jk} *with* $Cost_{min}$
    Backward Recurse Get V_s, E_s
    $V_{simp} \leftarrow V_{simp} \cup V_s; E_{simp} \leftarrow E_{simp} \cup E_s$
  end for
end for
$G_{simp} \leftarrow (V_{simp}, E_{simp})$
for *each* $x_{node} \subset G_{simp} = (V_{simp}, E_{simp})$ **do**
  $ChooseParent$;$Rewire$
return $G_{simp} = (V_{simp}, E_{simp})$
end for

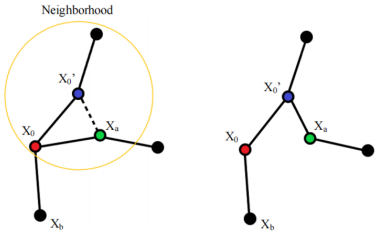


Fig. 11. Root moving with Rewire operation

iteration increases with the number of samples or the number of nodes in the tree data. After a large number of samplings, algorithm needs to spend much longer time to traverse nodes in tree data. For the existing tree, there are already a considerable number of nodes in the matrix of tree data, it thus takes long time to complete an episode from the beginning. The second step of Algorithm 6 is applied to reduce the required time to update global tree connections by simplifying the tree. The specific method for tree simplifying is used to segment the configuration space into $j \times k$ rectangles according to environmental factors such as dense or sparse obstacles. After that, Algorithm 6 finds the node with the minimum cumulative cost in each rectangle, obtains a branch connected with root of tree by backward recursion. The nodes in simplified tree data will keep their original indexes, only one duplicated node will be retained by conducting *unique* function. By observing the complete RRT* tree graph and complete matrix of the tree data, it can be found that a considerable number of nodes in the tree have similar coordinates and similar cumulative costs. Moreover, this approach for selecting the nodes can make nodes of the simplified tree cover the whole configuration space. Therefore, this simplified method is feasible in practical application which just slightly sacrifices quality of the re-planned path. However, it saves abundant computing resources and greatly improves response speed. Based on the moved root and simplified tree data, Algorithm 6 updates the connections between the nodes through *ChooseParent* and *Rewire*.

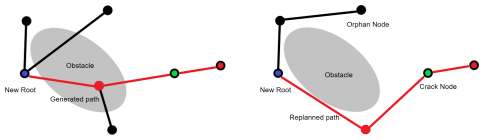


Fig. 12. Re-planning with new root

Algorithm 7 runs based on the tree which is updated and simplified by Algorithm 6. When the dynamic obstacles appear on the path, Algorithm 7 deletes the nodes within dynamic obstacles and edges that conflict with the obstacles. This causes some nodes in the tree to lose their parents, such nodes are called orphan nodes x_{orphan} in this paper. Moreover, the node on the path that loses the child node on the path will become the new root. After finding out all orphan nodes, the tree will be updated in new environment by executing *ChooseParent* and *Rewire* operations for orphan nodes. There is also an orphan node on the original path, which is called crack node x_{crack}. Crack nodes can be iterated through the surrounding nodes and select the most qualified near node as parent, generating a collision avoidance path to new root by backward recursion, so that the broken part of original path due to dynamic obstacles can be re-connected. Fig. 12 shows the operation of Algorithm 7.

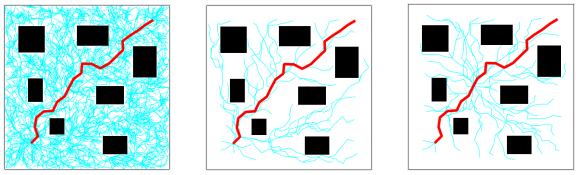
Algorithm 7 RePlanning

$X_{orphan} \leftarrow \emptyset$
for *each* $x_{node} \subset G_{simp} = (V_{simp}, E_{simp})$ **do**
  if $Parent(x_{node}) = \emptyset$ **then**
    $X_{orphan} \leftarrow X_{orphan} \cup x_{node}$
  end if
end for
for *each* $x_{orphan} \subset X_{orphan}$ **do**
  $Neighbour(G_{simp} = (V_{simp}, E_{simp}), x_{orphan})$
  $ChooseParent$; $Rewire$
return $G_{simp} = (V_{simp}, E_{simp})$
end for
$X_{near} \leftarrow Neighbour(G_{simp} = (V_{simp}, E_{simp}), x_{crack})$
$Parent(x_{crack}) \leftarrow \emptyset$; $Cost(x_{init}, x_{crack}) \leftarrow \infty$
for *each* $x_{near} \subset X_{near}$ **do**
  if $CollisionFree(x_{near}, x_{crack}) \wedge Cost(x_{init}, x_{near})$
$+Distance(x_{near}, x_{crack}) < Cost(x_{init}, x_{crack})$ **then**
    $Parent(x_{crack}) \leftarrow x_{near}$
    $Cost(x_{init}, x_{crack}) \leftarrow Cost(x_{init}, x_{near})$
$+Distance(x_{near}, x_{crack})$
  end if
end for
Backward Recurse to Generate Replan Path

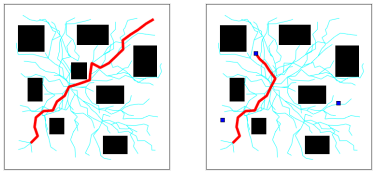
After the execution of Algorithms 6 and 7, a re-planned path is generated. The re-planned path is approximately optimized and its proportion in the whole path is relatively small.

Therefore, overall quality of entire path is still very high and the entire path is still close to original one. In fact, Algorithm 6 can be conducted in advance. Thus, in case of emergency, only Algorithm 7 needs to be executed. Because there are far fewer nodes in the simplified tree, the processing time is very short, which makes this algorithm to be called rapid re-planning algorithm. In addition, based on the tree which is updated and simplified by Algorithm 6, a rapid path re-planning algorithm for emergency landing can be derived. All nodes in the tree are optimized and each of their cumulative costs cannot be less in current environment. Therefore, as long as the algorithm traverse all near nodes of vertiports in configuration space, selecting the most qualified near node as the parent node of a vertiport, then the approximate optimal path can be obtained through backward recursion from the new root node to the most suitable vertiport for emergency landing.

B. Simulation and Results



(a) Original path and tree (b) Simplified tree for initial root (c) Simplified tree for new root



(d) Re-planning based on updated tree (e) Re-planning for emergency landing

Fig. 13. Tree updating and Path re-planning

V. Conclusion

UAM platforms in urban airspace need to consider environmental, operational and even social factors, such as static and dynamic geofences, micro-weather patterns, CNS quality-of-service, noise and visual pollution, privacy as well as energy status of aerial vehicles. According to the defined problems above, this paper describes an RRT* based approach for generating energy-efficient path in various wind conditions, conforming to environmental and operational limitations, while finding the most suitable vertiport for mid-flight landing for battery changing or emergency situations. Compared with the RRT*, this method can generate higher quality energy-efficiency paths with shorter processing time and it has a unique operation of obtaining vertiport for mid-flight landing. The impact of different wind conditions is investigated by utilizing the simulation studies. However, this method assumes constant forward velocity and thrust, as well as attack of angle, which makes it unable to generate accurate energy-efficient path in practical. Therefore, we are planning to improve the algorithm for solving this problem in future work. We want to focus on RRT* based rapid path re-planning method for emergency landing or avoiding dynamic geofences. Reinforcement learning might be another powerful method to solve this path planning problem and we will also focus on this subject as future work.

Acknowledgements

This work is supported, in parts, by the Engineering and Physical Sciences Research Council [Grant Number: EP/V026763/1].

References

[1] Edward Xu. The Future of Transportation: White Paper on Urban Air Mobility Systems. EHang, 2020.
[2] Bauranov A, Rakas J. Designing airspace for urban air mobility: A review of concepts and approaches[J]. Progress in Aerospace Sciences, 2021, 125: 100726.
[3] Vascik P D, Hansman R J. Constraint identification in on-demand mobility for aviation through an exploratory case study of los angeles[C]//17th AIAA Aviation Technology, Integration, and Operations Conference. 2017: 3083.
[4] Vascik P D, Balakrishnan H, Hansman R J. Assessment of air traffic control for urban air mobility and unmanned systems[J]. 2018.
[5] Murray C W A, Ireland M, Anderson D. On the response of an autonomous quadrotor operating in a turbulent urban environment[C]//AUVSI's unmanned systems conference. 2014.
[6] Yedavalli P, Mooberry J. An Assessment of Public Perception of Urban Air Mobility (UAM)[J]. Airbus UTM: Defining Future Skies, 2019: 2046738072.1580045281-1681120550.
[7] Intaratep N, Alexander W N, Devenport W J, et al. Experimental study of quadcopter acoustics and performance at static thrust conditions[C]//22nd AIAA/CEAS Aeroacoustics Conference. 2016: 2873.
[8] Kwon H, Kim J, Park Y. Applying LSA text mining technique in envisioning social impacts of emerging technologies: The case of drone technology[J]. Technovation, 2017, 60: 15-28.
[9] Villasenor J. Observations from above: unmanned aircraft systems and privacy[J]. Harv. JL and Pub. Pol'y, 2013, 36: 457.
[10] Al Haddad C, Chaniotakis E, Straubinger A, et al. Factors affecting the adoption and use of urban air mobility[J]. Transportation research part A: policy and practice, 2020, 132: 696-712.
[11] Pang B, Ng E M, Low K H. UAV Trajectory Estimation and Deviation Analysis for Contingency Management in Urban Environments[C]//AIAA AVIATION 2020 FORUM. 2020: 2919.
[12] Logan M J, Bird E, Hernandez L, et al. Operational Considerations of Small UAS in Urban Canyons[C]//AIAA Scitech 2020 Forum. 2020: 1483.
[13] Radio Technical Commission for Aeronautics. Minimum Aviation System Performance Standards for Automatic Dependent Surveillance Broadcast (ADS-S)[M]. RTCA, Incorporated, 2002.
[14] Vascik P D, Hansman R J. Evaluation of key operational constraints affecting on-demand mobility for aviation in the Los Angeles basin: ground infrastructure, air traffic control and noise[C]//17th AIAA Aviation Technology, Integration, and Operations Conference. 2017: 3084.
[15] Rezende R N, Barros E, Perez V. General aviation 2025-A study for electric propulsion[C]//2018 Joint Propulsion Conference. 2018: 4900.
[16] Yu X, Zhang Y. Sense and avoid technologies with applications to unmanned aircraft systems: Review and prospects[J]. Progress in Aerospace Sciences, 2015, 74: 152-166.
[17] Huang H, Savkin A V. Energy-efficient decentralized navigation of a team of solar-powered UAVs for collaborative eavesdropping on a mobile ground target in urban environments[J]. Ad Hoc Networks, 2021, 117: 102485.
[18] Karaman, S. and Frazzoli, E., 2011. Sampling-based algorithms for optimal motion planning. The international journal of robotics research, 30(7), pp.846-894.
[19] Johnson W. Helicopter theory[M]. Courier Corporation, 2012.
[20] Seddon J M, Newman S. Basic helicopter aerodynamics[M]. John Wiley and Sons, 2011.
[21] Johnson W. Rotorcraft aeromechanics[M]. Cambridge University Press, 2013.

https://dspace.org/

An RRT* based method for dynamic mission balancing for urban air mobility under uncertain operational conditions

Lou, Junlin; Yuksek, Burak; Inalhan, Gokhan; Tsourdos, Antonios

2021-11-15
IEEE
https://doi.org/10.1109/DASC52595.2021.9594424
Lou J, Yuksek B, Inalhan G, Tsourdos A. (2021) An RRT* based method for dynamic mission balancing for urban air mobility under uncertain operational conditions. In: 2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC), 3-7 October 2021, San Antonio, TX USA

Downloaded from a DSpace repository. https://dspace.org/