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
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 sight 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 geofences 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;
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 charging 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]:

\[ E = P t \approx \left( \sum_{n=1}^{n}(P_i + P_o)m + P_p \right)_{V_{c/d}=0} + \sum_{m=1}^{m}(P_c/d)m \] (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]:

\[ E \approx n \left( \frac{kT^2}{2pAV_f}B + \frac{1}{8}C_{do}N_cR(1 + 4.6\mu^2)\rho(\omega R)^3 \right) + \frac{1}{2}f\rho V_f^4 + \frac{nT_f V_{c/d}}{\tan \alpha} \] (2)

where \( k \) is non-uniform flow correction, \( T_f \) is forward thrust, \( \rho \) 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, $\mu$ is rotor advance ratio, $\omega$ is rotor rotation speed, $f$ is equivalent flat-plate drag area, $\alpha$ 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 have a ratio, which is called Air/Ground ratio in this paper. When there is uniform and constant X-axis positive direction of x-axis.

When the angle between direction of wind and positive X-axis is $\sigma$, 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:

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

Time-varying wind environment can also be created:

$$V_{wind} = a \sin(bt_{cu}) + l$$

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

1: \( V \leftarrow x_{\text{init}}; \quad E \leftarrow \emptyset; \)
2: \( \text{verti}_k \in \text{Vertiports} \)
3: \( \text{PassVerti}(\text{verti}_k) \leftarrow k \)
4: \( \text{for } i \leftarrow 1 : m \text{ do} \)
5: \( \text{for } k \leftarrow 1 : \text{size(Vertiports)} \text{ do} \)
6: \( \text{Samplingbias}(\text{verti}_k) \)
7: \( \text{end for} \)
8: \( \text{if rand < biasprob then} \)
9: \( x_{\text{simp}} \leftarrow \text{Samplingbias}_i \)
10: \( \text{else} \)
11: \( x_{\text{simp}} \leftarrow \text{SamplingFree}_i \)
12: \( \text{end if} \)
13: \( x_{\text{nearest}} \leftarrow \text{Nearest}(G = (V, E), x_{\text{simp}}) \)
14: \( x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{simp}}) \)
15: \( \text{if ObstacleFree}(x_{\text{nearest}}, x_{\text{new}}) \text{ then} \)
16: \( x_{\text{near}} \leftarrow \text{Neighbour}(G = (V, E), x_{\text{new}}) \)
17: \( \text{PassVerti}(x_{\text{near}}) \leftarrow 0 \)
18: \( V \leftarrow V \cup x_{\text{new}} \)
19: \( x_{\text{min}} \leftarrow x_{\text{nearest}} \)
20: \( c_{\text{min}} \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{nearest}}) + \text{Cost}(x_{\text{nearest}}, x_{\text{new}}) \)
21: \( \text{for each } x_{\text{near}} \subset X_{\text{near}} \text{ do} \)
22: \( \text{if CollisionFree}(x_{\text{near}}, x_{\text{new}}) \text{ and } \text{Cost}(x_{\text{near}}, x_{\text{new}}) < c_{\text{min}} \text{ then} \)
23: \( x_{\text{min}} \leftarrow x_{\text{near}} \)
24: \( c_{\text{min}} \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{near}}) + \text{Cost}(x_{\text{near}}, x_{\text{new}}) \)
25: \( \text{end if} \)
26: \( \text{end for} \)
27: \( E \leftarrow E \cup (x_{\text{min}}, x_{\text{near}}) \)
28: \( \text{for each } x_{\text{near}} \subset X_{\text{near}} \text{ do} \)
29: \( \text{if CollisionFree}(x_{\text{near}}, x_{\text{new}}) \text{ and } \text{Cost}(x_{\text{init}}, x_{\text{near}}) < \text{Cost}(x_{\text{init}}, x_{\text{new}}) \text{ then} \)
30: \( x_{\text{new}} \leftarrow \text{ChooseParent}(x_{\text{init}}, x_{\text{near}}) \)
31: \( E \leftarrow E \cup (x_{\text{new}}, x_{\text{near}}) \)
32: \( \text{if CollisionFree}(x_{\text{new}}, x_{\text{near}}) \text{ then} \)
33: \( \text{Cost}(x_{\text{near}}, x_{\text{new}}) = \text{Cost}(x_{\text{init}}, x_{\text{new}}) \)
34: \( \text{Parent}(x_{\text{near}}) = x_{\text{new}} \)
35: \( \text{Cost}(x_{\text{init}}, x_{\text{near}}) = \text{Cost}(x_{\text{init}}, x_{\text{new}}) \)
36: \( E \leftarrow E \cup (x_{\text{new}}, x_{\text{near}}) \)
37: \( \text{Propagate Cost Changes to Leaves} \)
38: \( \text{end if} \)
39: \( \text{end for} \)
40: \( \text{return } G = (V, E) \)
41: \( \text{end for} \)
42: \( \text{Save paths to Vertiports} \)
43: \( \text{Generate path to Goal} \)

The proposed algorithm is given in Algorithm 1. Here, it is executed until \( m \text{th} \) 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 \( \text{PassVerti} \) to nodes on the tree, which makes the matrix of tree data have one more column. After \( m \text{th} \) sampling, algorithm 2 and algorithm 3 will be executed.

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

- \( \text{SamplingFree} \): A procedure returns a new state \( x_{\text{simp}} \), which is generated randomly.
- \( \text{Samplingbias} \): A procedure returns a new state \( x_{\text{simp}} \), which is near one of vertiports.
- \( \text{Nearest} \): A procedure returns the nearest node of \( x_{\text{simp}} \) in tree.
- \( \text{Steer} \): A procedure returns a new node \( x_{\text{new}} \) steering from \( x_{\text{nearest}} \) to \( x_{\text{simp}} \) at a certain distance.
- \( \text{ObstacleFree} \): A procedure checks if there is collision in steering or not.

- \( \text{Neighbour} \): A procedure returns nodes in tree whose Euclidean distance to \( x_{\text{new}} \) is less than a predefined value.
- \( \text{PassVerti} \): A label used for marking a node whether it is on a branch connected with a vertiport and defining which vertiport is connected.
- \( \text{CollisionFree} \): A procedure tests if collision occurs on the straight-line segment between two nodes.
- \( \text{Propagate Cost Changes to Leaves} \): A procedure iterates through children, then recurse to update their costs until no any child found in tree data.
- \( \text{Cost} \): 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 \( \text{ChooseParent} \) and \( \text{Rewire} \) operations. These two procedures require to calculate energy consumption cost through coordinates of \( x_{\text{near}} \) and \( x_{\text{new}} \) many times. At this time, these variables can be derived by Eq.(14)(15)(16)(17):

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

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

\[
\theta = \arccos \left( \frac{\Delta x^2}{\Delta x^2 + \Delta y^2} \right) \tag{16}
\]

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

where \( \Delta x \), \( \Delta y \) and \( \Delta z \) are coordinates differences of a pair of nodes in X, Y, Z axes, \( \beta \) is angle between vehicle flight direction and horizontal plane, \( \omega_0 \) 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 \( \text{Samplingbias} \), 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

1: for $k \leftarrow 1$: size(Vertiports) do
2: $X_{\text{near}} \leftarrow \text{Neighbour}(\text{Verti}_k)$
3: $\text{Parent}(\text{Verti}_k) \leftarrow \emptyset$; $\text{Cost}(\text{init}, \text{Verti}_k) \leftarrow \infty$
4: for each $x_{\text{near}} \in X_{\text{near}}$ do
5: if CollisionFree$(x_{\text{near}}, \text{Verti}_k)$ ∧
6: Cost$(x_{\text{init}}, x_{\text{near}}) + \text{Cost}(x_{\text{near}}, \text{Verti}_k) < \text{Cost}(x_{\text{init}}, \text{Verti}_k)$ then
7: $\text{Cost}(x_{\text{init}}, \text{Verti}_k) \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{near}}) + \text{Cost}(x_{\text{near}}, \text{Verti}_k)$
8: $\text{Parent}(\text{Verti}_k) \leftarrow x_{\text{near}}$
9: end if
10: return $\text{Cost}(x_{\text{init}}, \text{Verti}_k)$

Algorithm 3 Generate path to Goal

1: for $i \leftarrow m + 1$: $n$ do
2: for $k \leftarrow 1$: size(Vertiports) do
3: $\text{Samplingbias}(\text{Verti}_k)$
4: end for
5: if $\text{rand} < \text{biasprob}$ then
6: $x_{\text{simp}} \leftarrow \text{Samplingbias}_i$
7: else
8: $x_{\text{simp}} \leftarrow \text{SamplingFree}_i$
9: end if
10: $x_{\text{nearest}} \leftarrow \text{Nearest}(G = (V, E), x_{\text{simp}})$
11: $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x_{\text{simp}})$
12: if OffsetFree$(x_{\text{nearest}}, x_{\text{new}})$ then
13: $X_{\text{near}} \leftarrow \text{Neighbour}(G = (V, E), x_{\text{new}})$
14: $\text{PassVerti}(x_{\text{new}}) \leftarrow 0$
15: $V \leftarrow V \cup x_{\text{new}}$
16: $x_{\text{min}} \leftarrow x_{\text{nearest}}$
17: $c_{\text{min}} \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{nearest}}) + \text{Cost}(x_{\text{nearest}}, x_{\text{new}})$
18: end for
19: $\text{Rewire Variant}$
20: end if
21: return $G = (V, E)$
22: end for
23: $X_{\text{near}} \leftarrow \text{Neighbour}(G = (V, E), x_{\text{goal}})$
24: $\text{Parent}(x_{\text{goal}}) \leftarrow \emptyset$; $\text{Cost}(x_{\text{init}}, x_{\text{goal}}) \leftarrow \infty$
25: for each $x_{\text{near}} \in X_{\text{near}}$ do
26: if CollisionFree$(x_{\text{near}}, x_{\text{goal}})$ ∧
27: Cost$(x_{\text{init}}, x_{\text{near}}) + \text{Cost}(x_{\text{near}}, x_{\text{goal}}) < \text{Cost}(x_{\text{init}}, x_{\text{goal}})$ then
28: $\text{Parent}(x_{\text{goal}}) \leftarrow x_{\text{near}}$
29: $\text{Cost}(x_{\text{init}}, x_{\text{goal}}) \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{near}})$
30: end if
31: end for
32: $\text{Backward Recurse to Generate Final Path}$

Algorithm 4 ChooseParent Variant

1: if CollisionFree$(x_{\text{init}}, x_{\text{new}})$$\land$
2: Distance$(x_{\text{init}}, x_{\text{new}}) < \text{Threshold}$ then
3: $\text{Parent}(x_{\text{new}}) \leftarrow \text{Verti}_k$
4: if PassVerti$(x_{\text{new}}) < 0$ then
5: $\text{Cost}(x_{\text{init}}, x_{\text{new}}) \leftarrow \text{Cost}(x_{\text{init}}, \text{Verti}_k)$
6: $\text{Cost}(x_{\text{init}}, x_{\text{new}}) \leftarrow \text{Cost}(x_{\text{init}}, \text{Verti}_k)$
7: end if
8: $\text{end if}$
9: return $\text{Cost}(x_{\text{init}}, x_{\text{new}})$

Algorithm 5 Rewrite Variant

1: for each $x_{\text{near}} \in X_{\text{near}}$ do
2: if CollisionFree$(x_{\text{init}}, x_{\text{new}}) \land \text{PassVerti}(x_{\text{new}}) > 0$$\land$$x_{\text{near}} \notin \text{Vertiports}$ then
3: if PassVerti$(x_{\text{new}}) = 0$ then
4: $\text{Cost}(x_{\text{init}}, x_{\text{new}}) \leftarrow \infty$
5: end if
6: $\text{end if}$
7: if Cost$(x_{\text{init}}, x_{\text{new}}) + \text{Cost}(x_{\text{near}}, x_{\text{new}}) < c_{\text{min}}$ then
8: $c_{\text{min}} \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{new}}) + \text{Cost}(x_{\text{near}}, x_{\text{new}})$
9: $\text{PassVerti}(x_{\text{new}}) \leftarrow \text{PassVerti}(x_{\text{new}})$
10: end if
11: end if
12: return $c_{\text{min}}$
13: end if
14: $\text{Rewire Variant}$
15: $\text{Propagate Cost and PassVerti Changes to Leaves}$
16: end if
17: $\text{end if}$
18: if CollisionFree$(x_{\text{new}}, x_{\text{near}}) \land \text{PassVerti}(x_{\text{near}}) = 0$ then
19: if PassVerti$(x_{\text{new}}) = 0$ then
20: if $\text{PassVerti}(x_{\text{near}}) < \text{Threshold}$ then
21: $\text{Parent}(x_{\text{new}}) \leftarrow x_{\text{new}}$
22: $\text{Cost}(x_{\text{init}}, x_{\text{new}}) \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{new}})$
23: $\text{Cost}(x_{\text{init}}, x_{\text{near}}) \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{new}})$
24: $\text{Cost}(x_{\text{init}}, x_{\text{new}}) \leftarrow \text{Cost}(x_{\text{init}}, x_{\text{new}})$
25: $\text{Propagate Cost Changes to Leaves}$
26: end if
27: $\text{end if}$

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

Algorithm 5 is a variant of $\text{Rewire}$ operation. It will inevitably update connection as well as $\text{Passverti}$ attribute if $\text{Passverti}$ attribute of $x_{\text{new}}$ is not 0 and $\text{Passverti}$ attribute of $x_{\text{near}}$ is equal to 0. Otherwise it will not update connection. If both of $x_{\text{near}}$ and $x_{\text{new}}$ 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, $\text{Passverti}$ 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 flew 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](image)

![Fig. 3. An example of planning](image)

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 will 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](image)

![Fig. 5. No Wind](image)

![Fig. 6. Week positive X-axis wind](image)

![Fig. 7. Strong positive X-axis wind](image)

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
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 Verti2, 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 Verti2 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.

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

1: \( G = (V, E) \) $\rightarrow$ tree data
2: \( \text{ConfigSpace} = Lim_x \times Lim_y \)
3: \( x_{near} \leftarrow \text{PathData} \)
4: \( x_n \subset X \)
5: \( V_{simp} \leftarrow x_n; E_{simp} \leftarrow \emptyset \)
6: for \( i = 1 : n \) do
7: \( x_{near} \leftarrow \text{Neighbour}(G = (V, E), x_i) \)
8: for each \( x_{near} \subset X_{near} \) do
9: if \( \text{CollisionFree}(x_i, x_{near}) \wedge \)
10: Distance \( (x_i, x_{near}) \leq \text{Cost}(x_{i-1}, x_{near}) + \text{Cost}(x_{i-1}, x_i) \) then
11: \( \text{Parent}(x_{near} \leftarrow x_i \)
12: \( \text{Cost}(x_{near}) \leftarrow \text{Distance}(x_i, x_{near}) \)
13: Propagate Cost Changes to Leaves
14: end if
15: end for
16: end for
17: for \( j = 1 : a \) do
18: for \( k = 1 : b \) do
19: \( A_{jk} \leftarrow \text{Area}[\text{Lim}_x, \text{Lim}_x, \text{Lim}_y, \text{Lim}_y] \)
20: Iterate through Node \( \subset A_{jk} \)
21: Find \( x_{jk} \) with \( \text{Cost}_{\text{min}} \)
22: Backward Recurse Get \( V, E \)
23: \( V_{simp} \leftarrow V_{simp} \cup V; E_{simp} \leftarrow E_{simp} \cup E \)
24: end for
25: end for
26: \( G_{simp} \leftarrow (V_{simp}, E_{simp}) \)
27: for each \( x_{node} \subset G_{simp} = (V_{simp}, E_{simp}) \) do
28: \( \text{ChooseParent}; \text{Rewire} \)
29: \( \text{return} G_{simp} = (V_{simp}, E_{simp}) \)
30: end for

Algorithm 7 RePlanning

1: \( x_{orphan} \leftarrow \emptyset \)
2: for each \( x_{node} \subset G_{simp} = (V_{simp}, E_{simp}) \) do
3: if \( \text{Parent}(x_{node}) = \emptyset \) then
4: \( x_{orphan} \leftarrow x_{orphan} \cup x_{node} \)
5: end if
6: end for
7: for each \( x_{orphan} \subset X_{orphan} \) do
8: \( \text{Neighbour}(G_{simp} = (V_{simp}, E_{simp}), x_{orphan}) \)
9: \( \text{ChooseParent}; \text{Rewire} \)
10: \( \text{return} G_{simp} = (V_{simp}, E_{simp}) \)
11: end for
12: \( x_{near} \leftarrow \text{Neighbour}(G_{simp} = (V_{simp}, E_{simp}), x_{crack}) \)
13: \( \text{Parent}(x_{crack}) \leftarrow \emptyset; \text{Cost}(x_{init}, x_{crack}) \leftarrow \infty \)
14: for each \( x_{near} \subset X_{near} \) do
15: if \( \text{CollisionFree}(x_{near}, x_{crack}) \wedge \text{Cost}(x_{init}, x_{near}) \)
16: Distance \( (x_{near}, x_{crack}) < \text{Cost}(x_{init}, x_{crack}) \) then
17: \( \text{Parent}(x_{crack}) \leftarrow x_{near} \)
18: \( \text{Cost}(x_{init}, x_{crack}) \leftarrow \text{Cost}(x_{init}, x_{near}) \)
19: \( \text{Distance}(x_{near}, x_{crack}) \)
20: \( \text{end if} \)
21: end for
22: \( \text{Backward Recurse to Generate Replan Path} \)

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

![Image](69x456 to 132x518)

(a) Original path and tree

![Image](107x362 to 170x425)

(b) Simplified tree for initial root

![Image](145x456 to 208x519)

(c) Simplified tree for new root

![Image](221x444 to 283x519)

(d) Re-planning based on updated tree

![Image](294x456 to 357x518)

(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 geofoences. 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