Available online at www.sciencedirect.com

ScienceDirect

Automatica 44 (2008) 696–712

automatica

www.elsevier.com/locate/automatica

Real-time path planning with limited information for autonomous unmanned air vehicles[☆]

Yoonsoo Kim[a,*], Da-Wei Gu[b], Ian Postlethwaite[b]

[a]Department of Mechanical and Mechatronic Engineering, University of Stellenbosch Private Bag X1, Matieland 7602, South Africa
[b]Department of Engineering, University of Leicester, Leicester LE1 7RH, UK

Received 5 June 2006; received in revised form 18 December 2006; accepted 9 July 2007
Available online 11 December 2007

Abstract

We propose real-time path planning schemes employing limited information for fully autonomous unmanned air vehicles (UAVs) in a hostile environment. Two main algorithms are proposed under different assumptions on the information used and the threats involved. They consist of several simple (computationally tractable) deterministic rules for real-time applications. The first algorithm uses extremely limited information (only the probabilistic risk in the surrounding area with respect to the UAV's current position) and memory, and the second utilizes more knowledge (the location and strength of threats within the UAV's sensory range) and memory. Both algorithms provably converge to a given target point and produce a series of safe waypoints whose risk is almost less than a given threshold value. In particular, we characterize a class of dynamic threats (so-called, static-dependent threats) so that the second algorithm can efficiently handle such dynamic threats while guaranteeing its convergence to a given target. Challenging scenarios are used to test the proposed algorithms.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Autonomous vehicles; Path planning; Limited data; Uncertainty; Static and dynamic threats

1. Introduction

There is an extensive literature on traditional path planning in which, for known obstacles, one searches for a collision-free shortest path connecting the starting and ending points of a vehicle. Most of the papers are motivated by robotic applications (see Latombe, 1991 for a survey), and the majority of promising path planning methods fall into three categories: roadmap, cell decomposition and potential field approaches. Firstly, roadmap approaches construct a connected graph consisting of vertices (waypoints) and edges that represent feasible routes for vehicle travelling. Methods used for graph construction include (generalized) Voronoi diagrams (Takahashi & Schilling, 1989), visibility lines (Tarjan, 1981), silhouettes (Canny, 1988), probabilistic graph expansions (Kavraki, Svestka, Latombe, & Overmars, 1996), etc. Among these methods, a class of algorithms known as probabilistic roadmap methods (PRMs) have received much attention for their simplicity and speed of convergence properties when solving problems in high-dimensional spaces. Secondly, cell decomposition approaches construct a connected graph in which a vertex corresponds to a *cell* which is part of the feasible vehicle operational space. One can further group this kind of approach into exact, approximate or probabilistic cell decompositions depending on the way the obstacles are treated (see Lingelbach, 2005 for details). Lastly, the celebrated potential field approaches, for example Kim and Khosla (1992), employ a navigation function (defined over the underlying space that has a unique global minimum at the vehicle ending point) and follow the steepest descent. The main concern in potential field methods is how to choose such a navigation function in order to avoid a vehicle being trapped at a local minimum. Some extensions of the potential field approach can be found in recent literature, e.g. the global dynamic window and the stream function approaches in

☆ This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form under the direction of Editor Berc Rüstem.

* Corresponding author. Tel.: +27 21 808 4265; fax: +27 21 808 4958.
E-mail addresses: ykim@sun.ac.za (Y. Kim), dag@le.ac.uk (D.-W. Gu), ixp@le.ac.uk (I. Postlethwaite).

0005-1098/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2007.07.023

Brock and Khatib (1999) and Waydo and Murray (2003), respectively.

All the aforementioned methods can be used for successful, or probabilistically successful, path planning in a classical sense in that they always find a collision-free path, whenever one exists, although some algorithms require a judicious choice of parameters and functions. However, direct application of these classical or modern path planning approaches appears to be less successful when the following constraints are to be satisfied simultaneously[1] : (1) the vehicle operational space $\mathbf{X}$ is large, e.g. $[0, 200] \times [0, 200] \times [0, 2]$ km; (2) $\mathbf{X}$ and the location of threats are uncertain and change in real-time; (3) sensors equipped on the vehicle have limited ranges; (4) deterministic convergence to the target point should be guaranteed whenever a feasible path is available in $\mathbf{X}$; (5) a resultant path should be within the vehicle dynamics and capability domain; (6) an algorithm must be simple enough to operate on-board. These constraints are realistic and often required for unmanned air vehicle (UAV) applications. Several UAV-related papers address path planning subject to some of these additional requirements. The associated algorithms involve, for example, mixed-integer linear programming (MILP) (Kamal, Gu, & Postlethwaite, 2005b; Richard, Schouwenaars, How, & Feron, 2002), Voronoi diagrams (Beard, McLain, Goodrich, & Anderson, 2002; Gu, Kamal, & Postlethwaite, 2004), probabilistic roadmaps (Pettersson & Doherty, 2004), evolutionary computation (Zheng, Li, Xu, Sun, & Ding, 2005), extensive local search (Kamal, Gu, & Postlethwaite, 2005a) and bouncing (Kim, Mesbahi, & Hadaegh, 2004), but as yet they need to be developed further for full consideration of all the foregoing requirements. We note that, in its full generality, the path planning problem is NP-complete (Hwang & Ahuja, 1992).

In particular, when considering requirements (4)–(6) above, we realize that there are few algorithms that utilize only limited local information and guarantee convergence, as well as involving relatively simple deterministic rules. We also note that in relation to (1), (2) and (4), even recent successful variations of PRMs, e.g. the rapid exploring random tree (RRT) approach, may struggle to find a solution for the problems having so-called *double bug traps* (LaValle, 2006; LaValle & Kuffner, 1999). This is because these methods are required to grow *trees* in many places and find difficulties in connecting them efficiently. This point will be revisited in Section 5, where we present several examples for testing algorithms. In conclusion, all these observations lead us to propose a novel and practically tractable path planning algorithm for a single UAV in this paper. The paper is organized as follows. We first identify in Section 2 the obstacles to be avoided based on the probabilistic risk of exposure to threats. This approach of calculating risk can be found in various papers and has been applied to the path planning of UAVs, e.g. Dogan (2003). We then present in Sections 3.1 and 3.3 several simple rules which are the main ingredients of the new algorithm (initially meant for fixed-altitude flights but further extended to free flights). The initial algorithms only allow *static* threats (to be explained formally in the next section) at this time. A theoretical justification for the algorithm is given in Section 3.2 and a smooth path planning scheme considered in Section 3.4 whereby a suitable value of a particular parameter is selected. The algorithms are then improved in Section 4 using more knowledge than needed for the initial algorithms. This avoids undesirable situations which might arise in the initial algorithm, in the presence of static (pop-up) threats and a limited class of dynamic threats. These results are illustrated through numerical investigations in Section 5, with conclusions in Section 6. Note that we summarize the main notation in Table 1.

[1] Although a success story on a real-time robotic application can be found in Brock and Khatib (1999) using the global dynamic window approach, the approach still seems incomplete in the sense that: (i) the model of the environment incorporating sensory data would not be sufficiently accurate to find a collision-free path, because only connectivity information about the operational space would be used for an uncertain environment; (ii) the employed local minima-free navigation function is not global in general (Koditschek & Rimon, 1990); (iii) the relationship between the width of the region over which the navigation function is computed and the corresponding computational demand is not clear.

Table 1
Notation frequently used in the paper

X	UAV operational plane
w	Point in $\mathbf{R}^3$
ρ	Constant risk threshold
$P(w)$	Probabilistic risk at w
$\mathbf{C}_w$	Cell which contains w
z	UAV altitude
β	Partitioned cell size
$P(\mathbf{C}_w)$	Probabilistic risk of $\mathbf{C}_w$
Ω	Union of Ω_z
Ω_z	Set of obstacles with an altitude of z
$\mathscr{A}_{2\mathrm{D}}$	Algorithm for fixed-altitude flights
$\mathscr{A}_{3\mathrm{D}}$	Algorithm for free flights
$\tilde{\mathscr{A}}_{3\mathrm{D}}$	Advanced algorithm for free flights
X_z	Plane with an altitude of z
Φ	Empty set
w_s	UAV's starting position
w_t	UAV's target position
$centre(\mathbf{C})$	Centre point of cell $\mathbf{C}$
$\mathscr{N}(w)$	Set of neighbouring cells around w
$\mathbf{C}_\mathrm{o}$	Obstacle cell
$\mathbf{C}_\mathrm{s}$	Safe cell
Look	Binary parameter (*Left* or *Right*)
$id(\mathbf{C})$	Unique number assigned to cell $\mathbf{C}$
$\tilde{\Omega}$	Union of Ω_z^θ
Ω_z^θ	Set of obstacles (height z and attitude θ)
$\mathscr{OC}$	Obstacle component
$\mathscr{OCR}$	Obstacle component range

2. Definition of obstacles

By "obstacle", we mean a region where a UAV is likely to fail to continue its mission. In other words, if we define $P(w)$ as the probabilistic risk of exposure at position $w \in \mathbf{R}^3$ to the sources of threat, the obstacle corresponds to the set of w's such that $P(w) > \rho$ where ρ is a nonnegative constant. Setting $\rho = 0$ leads to a perfect-risk-free path, if available. However, in the absence of such a perfect-risk-free path, we set ρ to be strictly positive and consider an optimal path $\mathbf{P}$ along which a desirable cost, e.g. time to travel, is minimized, subject to $P(w) \leqslant \rho, \forall w \in \mathbf{P}$.

There are numerous ways of modelling $P(w)$ in the literature, each of which is basically motivated by the nature of specific threat events. In particular, we note several stochastic types of risk modelling. For example, in Nilim, El Ghaoui, and Duong (2002) the main threat source is convective weather, whose dynamics has a stochastic nature, and $P(w)$ is predicted using the maximum likelihood function. Another interesting example can be found in Nikolova, Brand, and Karger (2006). The authors therein assume that some unknown risk sources, e.g. car traffic, randomly affect the vehicle travel time and the associated cost, and the travel time is captured by a Gamma distributed process. These stochastic types of risk modelling can be also found in the context of the so-called *Dynamic Travelling Repairperson Problem* (see Enright & Frazzoli, 2005), where the targets to be visited by a UAV are modelled as a homogeneous spatio-temporal Poisson process. Although these stochastic modelling approaches attract practical interest, in this paper we only consider a deterministic risk model. As will be presented shortly, we assume that our main threat sources are ground-based missile units whose resultant risk distribution is deterministically represented in many UAV applications, e.g. Gu et al. (2004). However, we note that our path planning algorithms to be proposed in this paper are not restricted to deterministic models, and the aforementioned stochastic approaches can be easily incorporated to accommodate various stochastic events at this risk modelling stage.

As a first step towards our risk modelling, we partition the slice $\mathbf{X}_z = [0,1] \times [0,1] \times z$ of the scaled operational space $\mathbf{X} = [0,1] \times [0,1] \times [0,1]$ into disjoint square cells of length $\beta (\leqslant 1)$, where $1/\beta$ is an integer.[2] Then, given a defined $P(w)$, $\forall w = (\tilde{x}, \tilde{y}, z_0) \in \mathbf{X}_{z_0}$, we wish to calculate the risk $P(\mathbf{C}_w)$ of cell $\mathbf{C}_w$ as follows:

$$P(\mathbf{C}_w) = \frac{\int_{(x,y,z_0)\in\mathbf{C}_w} P(w)\,\mathrm{d}x\,\mathrm{d}y}{\beta^2}, \tag{1}$$

where $\mathbf{C}_w = \{(x, y, z_0) | (c_x - 1)\beta \leqslant x \leqslant c_x\beta, (c_y - 1)\beta \leqslant y \leqslant c_y\beta\}$ and $c_x = \lceil \tilde{x}/\beta \rceil$, $c_y = \lceil \tilde{y}/\beta \rceil (\in \{1, 2, \ldots, 1/\beta\})$ (see Fig. 1). The formula (1) for calculating $P(\mathbf{C}_w)$ needs to be approximated for real-time applications. One candidate for this is the sum of the contributions due to the four corner points of $\mathbf{C}_w$, i.e. $P(\mathbf{C}_w) = \sum_{\alpha_1,\alpha_2} P((c_x - \alpha_1)\beta, (c_y - \alpha_2)\beta, z_0)/4$, where $\alpha_1, \alpha_2 \in \{0, 1\}$.[3]

As noted before, we consider the principal source of danger as M enemy missile units with short ($R_s = 7\,\mathrm{km}$), medium

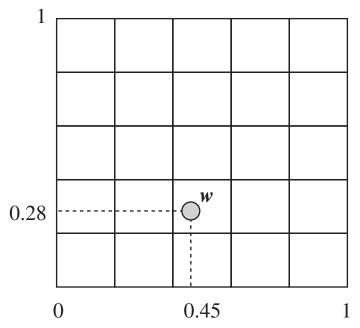

Fig. 1. A partition of the slice $\mathbf{X}_{z_0}$ of the operation space with $\beta = 0.2$: if $w = [0.45,\ 0.28,\ z_0]^{\mathrm{T}}$, then $c_x = \lceil \tilde{x}/\beta \rceil = \lceil 0.45/0.2 \rceil = 3$, $c_y = \lceil \tilde{y}/\beta \rceil = \lceil 0.28/0.2 \rceil = 2$, and thus $\mathbf{C}_w = \{(x, y, z_0) \,|\, 0.4 \leqslant x \leqslant 0.6, 0.2 \leqslant y \leqslant 0.4\}$; note that $\mathbf{C}_w$ contains w.

(25 km), or long (65 km) ranges. Then, as found in Gu et al. (2004), we calculate $P(w)$ from the following formula:

$$P(w) = 1 - \prod_{i=1}^{M} (1 - P_i(w)), \tag{2}$$

where $P_i(w) = (1 - \mathit{Step}\ (d, R_{\{\mathrm{s,m,l}\}}, k_1)) \times \mathit{Step}\ (d, 0.1 \times R_{\{\mathrm{s,m,l}\}}, k_2) \times \mathit{Step}\ (\sin^{-1}(z_0/d), \gamma, k_3)$; $\mathit{Step}\ (a, b, c) = (1 + (a-b)/\sqrt{c^2 + (a-b)^2})/2$; d is the distance from w to the ith missile; γ is the lowest coverage angle of the radars (assumed to be 0.17 rad); and k_1, k_2, k_3 are the softness parameters of the *Step* function with the values of 5, 1, 0.1, respectively. P_i represents the probabilistic risk solely due to the ith missile unit and is a nonlinear function of the distance d and the altitude z_0 of a UAV with respect to the ith missile unit. The first two terms of P_i state that P_i increases as d does likewise, but decreases if d becomes too small. The last term says that P_i is small if a UAV stays below the radar coverage area. Overall, P_i forms a mushroom-shaped distribution of the probability with respect to d and z_0.

Given a proper value of $\rho \geqslant 0$,[4] we then have the follow ing set $\mathbf{\Omega}$ of defining obstacles:

$$\mathbf{\Omega} = \bigcup_z \mathbf{\Omega}_z, \tag{3}$$

where $\mathbf{\Omega}_{z_0} = \{w \in \mathbf{X}_{z_0} \mid P(\mathbf{C}_w) > \rho\}$. The risk $P(w)$ (or $P(\mathbf{C}_w)$) is often time-varying as the source of danger is likely to be, which implies that the (possibly huge) risk distribution map must be updated in real-time. This naturally implies that the family of path planning algorithms which require global or a large amount of local information can fail to yield satisfactory performance in real-time situations. In this regard, we first assume that if a UAV is located at w, it has the capability of recognizing in real-time whether or not each of its neighbouring cells, within a sphere $\mathbf{S}$ of radius 3β centred at w, is an obstacle. In addition, it is assumed that all obstacles in the UAV operational range are *static*. By a *static* obstacle, we mean that

[2] One may be curious about why the whole operational space $\mathbf{X}$ is not partitioned with three-dimensional cubic cells from the start. This is, as explained later, because we have found a direct partition of the three-dimensional space to be intractable for our problem. We also note that the way of decomposing the operational space and defining the obstacles is very similar to that adopted in approximate cell decomposition methods (Lingelbach, 2005).

[3] The average risk representation for $P(\mathbf{C}_w)$ can be dangerous in the presence of a thin *wall* of high risk. In this case, one has to consider the maximum risk contained in the associated cells, along with the average risk. However, in this paper we do not pursue this direction because the risk formula (2) to be used does not lead to this case.

[4] The parameter ρ should be appropriately selected such that the total flight time does not exceed a certain limit. This point will be revisited in Section 3.4.

once the obstacle is known to be dangerous, it stays dangerous. We also assume that pop-up threats do not appear in such a way that the current UAV location becomes dangerous. The algorithms $\mathscr{A}_{2\mathrm{D}}$ and $\mathscr{A}_{3\mathrm{D}}$ in Sections 3.1 and 3.3 are based on this assumption. We then consider more practical assumptions and change the algorithm accordingly in Section 4. It is further assumed in Section 4 that a UAV is capable of detecting the location and strength of threats within an observable range much larger than 3β, and of handling a limited type of *dynamic* obstacles while guaranteeing the convergence of the algorithm.

3. Local path planning algorithm for static threats

Typical path planning schemes which are based on only local information consist of two phases. The first corresponds to a so-called optimization phase: one tries to find an optimal direction along which an underlying objective, e.g. distance to a target point, is minimized or maximized. This can be readily done by solving a mathematical program or performing an extensive search. In the second phase, one must allow the UAV to escape from a region where the optimization step leads to an infeasible movement of the vehicle. However, due to the non-convex nature of the problem involved in this second phase, it is not clear when one can switch back to the first phase.

In this section, we propose a simple approach to path planning that resolves the transition issue and is also suitable for fully autonomous UAV applications. In order to meet the challenging requirements of such applications, we develop the algorithm so that it is: (i) based on only local information; (ii) convergent to a target point; (iii) used for designing a smooth path; (iv) simple and consequently fast. In the first optimization phase, the proposed algorithm (denoted by $\mathscr{A}$) follows the classical simple idea of heading directly towards a target point to reduce flight time. However, as $\mathscr{A}$ enters the second phase, i.e. the path for the UAV becomes *unsafe*, we turn our attention to the cell, as defined earlier. At each time step when we determine the next direction to proceed, we seek a *desirable* (to be explained) cell among neighbouring cells around the current position. Then, the centre (denoted by *centre*($\mathbf{C}$)) of the desired cell $\mathbf{C}$ serves as a reference point (next waypoint) from which a smooth path can be designed, and thus enables a UAV to manoeuvre to meet restricted flight conditions (see Section 3.4). The transition from the second to the first phase occurs when the minimization of the distance to the target point is possible (safe again), and the current cell is being visited for the first time for this transition purpose. Fig. 2 illustrates how $\mathscr{A}$ produces waypoints w_i's. We here remark that algorithm $\mathscr{A}$ is not a myopic one-*step*-ahead-type algorithm, but a one-*cell*-ahead-type algorithm. The cell size β is chosen based on sensor and UAV characteristics as well as on the degree of approximation of the original threat model.

In what follows, we first develop an algorithm for a fixed-altitude flight for which $\mathbf{\Omega} = \mathbf{\Omega}_z$ with a fixed z. We then modify the initial algorithm to be suitable for free flight in a three-dimensional environment without the same altitude flight constraint.

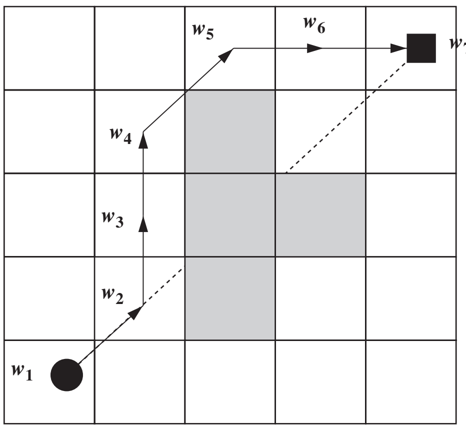

Fig. 2. Typical waypoints, w_i's, generated from the algorithm $\mathscr{A}$: w_1 and w_7 are given initial and target points, respectively; the intervals $[w_1, w_2]$ and $[w_5, w_7]$ correspond to the first phase of $\mathscr{A}$, and the interval $[w_2, w_5]$ the second phase; the shaded cells denote obstacles.

3.1. Fixed-altitude flight and algorithm $\mathscr{A}_{2\mathrm{D}}$

Let us begin a more detailed presentation by clearly defining the problem we are interested in. With the terminology and the definition of obstacles in Section 2, a typical path planning problem at a fixed altitude can be described as follows:

$\mathscr{P}_{2\mathrm{D}}$: Given UAV starting and target positions $w_\mathrm{s}, w_\mathrm{t} \in \mathbf{X}_{z_0} \cap \mathbf{\Omega}^\mathrm{c}$, where $\mathbf{X}_{z_0} = [0,1] \times [0,1] \times z_0$, $\rho, \beta > 0$ and $\mathbf{\Omega}^\mathrm{c}$ is the complement of $\mathbf{\Omega}$ defined by (3), find a sequence of (unknown a priori) n waypoints, $\{w_l\}_{l=1}^n$, such that $w_1 = w_\mathrm{s}$, $w_n = w_\mathrm{t}$, $v_j \notin \mathbf{\Omega}$, where $v_j = t w_j + (1-t) w_{j+1} (0 \leqslant t \leqslant 1)$ for $j = 1, 2, \ldots, n-1$ $(n \geqslant 2)$, and $\|w_j - w_{j+1}\| \geqslant (1 - \sqrt{2}/2)\beta$ for $j = 1, 2, \ldots, n-2$ $(n > 2)$. The condition of $\|w_j - w_{j+1}\| \geqslant (1-\sqrt{2}/2)\beta$, where $\|\cdot\|$ denotes the Euclidean 2-norm, is imposed to facilitate the design of a smooth path (see Section 3.4). Note that the value of β is chosen to be the same as the size of the cells in our algorithm below. We assume that cells $\mathbf{C}_{w_\mathrm{s}}$ and $\mathbf{C}_{w_\mathrm{t}}$ are safe, i.e. $\mathbf{C}_{w_\mathrm{s}} \cap \mathbf{\Omega} = \mathbf{C}_{w_\mathrm{t}} \cap \mathbf{\Omega} = \Phi$, the empty set. We define the distance between a cell $\mathbf{C}$ and w_t as $\|centre(\mathbf{C}) - w_\mathrm{t}\|$, and the *desirable* cell $\mathbf{C}_\mathrm{s} = \mathrm{argmin}_{\mathbf{C} \in \mathscr{N}(w_k)} \|centre(\mathbf{C}) - w_\mathrm{t}\|$, where $\mathscr{N}(w_k)$ is the set of safe neighbouring cells around w_k. The neighbouring cells around w_k are the eight cells numbered from 1 to 8 in Fig. 4.[5] The algorithm $\mathscr{A}_{2\mathrm{D}}$, for fixed-altitude flights, can now be stated as follows:

Algorithm $\mathscr{A}_{2\mathrm{D}}$.
Initialization: Let an index $k := 1$, a set $\mathbf{\Gamma} := \Phi$, an obstacle cell $\mathbf{OC} := 0$ and the first waypoint $w_1 := w_\mathrm{s}$. Partition the operational space into disjoint square cells of length β and assign a unique number $id(\mathbf{C})$ to each cell $\mathbf{C}$.

```
Phase I
Step I-1:  Define the line segment $\overrightarrow{w_k w_\mathrm{t}}$ joining $w_k$ and $w_\mathrm{t}$.
Step I-2:  If $\|\overrightarrow{w_k w_\mathrm{t}}\| \leqslant \beta$, define $\tilde{w} := w_\mathrm{t}$;
           else define $\tilde{w} (\in \overrightarrow{w_k w_\mathrm{t}})$ such that $\|\overrightarrow{w_k \tilde{w}}\| = \beta$.
```

[5] We note that the distance between w_{k-1} and modified w_k (reset to $centre(\mathbf{C}_{w_k})$ in Step I-3) is at least $(1 - \sqrt{2}/2)\beta$.

```
Step I-3:   If $\overrightarrow{w_k\tilde{w}} \cap \mathbf{\Omega} = \Phi$;
                if $\|\overrightarrow{w_k\tilde{w}}\| < \beta$, terminate the algorithm;
                else set $w_{k+1} := \tilde{w}$; $k := k+1$; goto
                Step I-1;
            else
                find the desirable cell $\mathbf{C}_s$ among
                neighbouring cells around $w_k$;
                $w_k$:=centre($\mathbf{C}_{w_k}$); $w_{k+1}$:=centre($\mathbf{C}_s$);
                define $\mathbf{C}_o$ as the first unsafe cell which
                intersects $\overrightarrow{w_k w_t}$;
                set $\mathbf{OC} := id(\mathbf{C}_o)$ and invoke
                Which Direction To Go to obtain Look;
                goto Phase II with $k := k + 1$.
Phase II
Step II-1:  Find the first cell $\mathbf{C}_n (\neq \mathbf{C}_{w_k})$ from $\mathbf{C}_{w_k}$
            which intersects $\overrightarrow{w_k w_t}$.
Step II-2:  If $\mathbf{C}_n \cap \mathbf{\Omega} = \Phi$ and $id(\mathbf{C}_n) \notin \mathbf{\Gamma}$,
                set $w_{k+1}$:=centre($\mathbf{C}_n$); $\mathbf{\Gamma}$:=$\mathbf{\Gamma} \cup id(\mathbf{C}_n)$;
                $\mathbf{OC} := 0$;
                goto Phase I with $k := k + 1$;
            else
                execute the Obstacle Following routine
                to update $\mathbf{OC}$ and obtain $w_{k+1}$;
                goto Step II-1 with $k := k + 1$.
```

Subroutines

Which Direction To Go: Given w_k, w_{k+1}, w_t and $\mathbf{C}_o$, this routine returns *Look*; if $\overrightarrow{w_k w_{k+1}}$ can be obtained up to scaling by rotating $\overrightarrow{w_k w_t}$ (by less than 180°) with respect to w_k anti-clockwise (respectively, clockwise), set *Look* := *Right* (respectively, *Look* := *Left*); see for illustration Fig. 3.

Obstacle Following: Given **OC** and *Look*, this routine updates and returns **OC** and w_{k+1}; by performing an anti-clockwise (if *Look* = *Right*) or clockwise (if *Look* = *Left*) search starting from **OC**, find the first safe neighbouring cell $\mathbf{C}_s$ and the unsafe cell $\mathbf{C}_o$ immediately before $\mathbf{C}_s$; set $\mathbf{OC} := id(\mathbf{C}_o)$ and $w_{k+1} :=$ *centre*($\mathbf{C}_s$); see for illustration Fig. 4.

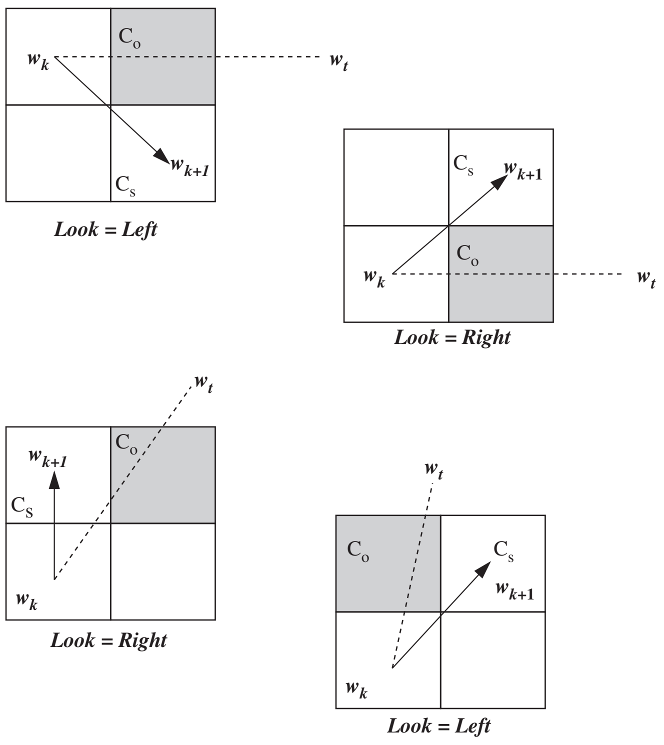

Fig. 3. Illustrations to show the determination of the flag *Look*.

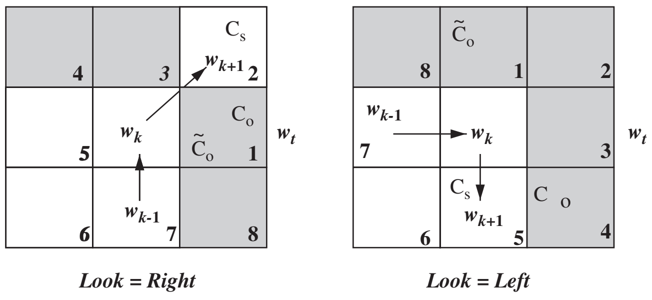

Fig. 4. Choosing the next waypoint w_{k+1} in **Phase II** based on the flag *Look*: at w_k, the search is performed on the order of the cells numbered from 1 (current $\mathbf{OC} = \tilde{\mathbf{C}}_o$ in each figure) to 8 for each case; the cell $\mathbf{C}_s$ is the first safe cell found by the search and **OC** is now updated to the obstacle cell ($\mathbf{C}_o$ in each figure) immediately before $\mathbf{C}_s$ in the context of each search.

3.2. Convergence of algorithm $\mathscr{A}_{2D}$

This section is dedicated to a theoretical justification of the proposed algorithm $\mathscr{A}_{2D}$. Because the algorithm employs the variable $\mathbf{\Gamma}$ to remember the cells where **Phase II** ends, the following convergence proof holds only when obstacles introduced in the operational space are static.

Proposition 1. *Suppose the path planning problem $\mathscr{P}_{2D}$ is feasible, i.e. a solution does exist. Then a sequence of (safe) waypoints generated from the proposed algorithm $\mathscr{A}_{2D}$ converges, i.e. the final waypoint of the sequence is the same as the target point w_t.*

Proof. According to **Phase I** of $\mathscr{A}_{2D}$, a sequence $\{w_l\}$ of waypoints generated from $\mathscr{A}_{2D}$ converges to the target point w_t unless trapped at a local point or in an infinite loop. Clearly, $w_j \neq w_{j+1}$ $(j \neq t)$, for one can always choose w_{j+1} as w_{j-1}, if no other safe waypoints exist at w_j. Together with the condition of $\|w_j - w_{j+1}\| \geqslant (1 - \sqrt{2}/2)\beta$, it is therefore not possible that $\{w_l\}$ ends at points other than w_t unless $\{w_l\}$ contains infinite loops.

In order to see if the existence of an infinitely repeating subsequence in $\{w_l\}$ is possible, consider such a subsequence $\{w_m\}$ $(\subset \{w_l\})$ and its corresponding sequence $\{\mathbf{C}_{w_m}\}$ of cells. Let us denote by $\mathbf{C}^{\mathrm{I}}$ the class of cells visited by the action of **Phase I** or the transition from **Phase II** to **Phase I** and, similarly, by $\mathbf{C}^{\mathrm{II}}$ the class of cells visited by the action of **Phase II** or the transition from **Phase I** to **Phase II**. First, in view of the optimization process in **Phase I**, one can readily see that

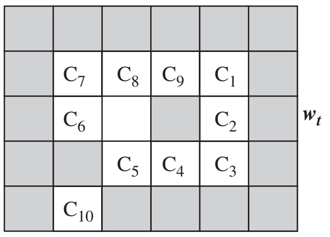

Fig. 5. An example of $\{\mathbf{C}_m\} \subseteq \mathbf{C}^{\mathrm{II}}$: $\{\mathbf{C}_m\} = \mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_9, \mathbf{C}_1, \ldots$.

there exists at least one $\mathbf{C}_i \in (\mathbf{C}^{\mathrm{II}} \cap \{\mathbf{C}_{w_m}\})$ where the transition from **Phase I** to **Phase II** occurs. Moreover, one can further deduce that the whole sequence $\{\mathbf{C}_{w_m}\}$ is contained in $\mathbf{C}^{\mathrm{II}}$, i.e. a UAV makes a repeat flight along the same boundary of obstacles. In fact, suppose $\mathbf{C}_j \in (\mathbf{C}^{\mathrm{I}} \cap \{\mathbf{C}_{w_m}\})$ for contradiction. This immediately implies the existence of $\mathbf{C}_k \in (\mathbf{C}^{\mathrm{II}} \cap \{\mathbf{C}_{w_m}\})$ where the transition from **Phase II** to **Phase I** occurs. Noting that the next cell (waypoint) after $\mathbf{C}_k$ is repeatedly chosen as $\mathbf{C}_{k+1}$, we can conclude $\mathbf{C}_{k+1} \in \mathbf{C}^{\mathrm{II}}$; otherwise, $\mathbf{C}_{k+1}$ would not be selected when $\mathbf{C}_k$ was visited twice (note that $\mathbf{C}_k$ is remembered as one of the cells where the transition from **Phase II** to **Phase I** occurs, i.e. $\mathbf{C}_k \in \mathbf{\Gamma}$, when being visited for the first time). Since $\mathbf{C}_j \neq \mathbf{C}_k$ or $\mathbf{C}_j \neq \mathbf{C}_{k+1}$, we can iteratively apply the same argument to contradict $\mathbf{C}_j \in \mathbf{C}^{\mathrm{I}}$, and thus to conclude $\{\mathbf{C}_{w_m}\} \subseteq \mathbf{C}^{\mathrm{II}}$.

We then proceed to show that $\{\mathbf{C}_{w_m}\} \subseteq \mathbf{C}^{\mathrm{II}}$ is prohibited by the feasibility of the problem. As illustrated in Fig. 5, suppose that the infinite loop in question is $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_9, \mathbf{C}_1, \ldots$. We note that the flag *Look* is set to *Left* in this example. The feasibility assumption allows the existence of a cell $\mathbf{C}_{10}$ through which a UAV can enter or exit the infinite loop, because w_{t} is outside the block of obstacles. Clearly, if w_{t} was inside the block of obstacles, the UAV would not get into the infinite loop. In fact, the algorithm must select $\mathbf{C}_{10}$, instead of $\mathbf{C}_6$, as the next cell at $\mathbf{C}_5$, governed by the obstacle-following rule in **Phase II** (see Fig. 4). Together with the boundedness of the operational space (and thus the number of total cells), we are led to the desired conclusion. □

3.3. Free flight (three-dimensional flight paths) and the algorithm $\mathscr{A}_{3\mathrm{D}}$

It would have been ideal if the algorithm $\mathscr{A}_{2\mathrm{D}}$ had been developed for free flight scenarios in the first place. The difficulty with this is that for the three-dimensional case, the binary flag *Look* in the obstacle-following step of $\mathscr{A}_{2\mathrm{D}}$ is insufficient to capture the additional degree of freedom in the movement of the UAV. For this purpose, one would need to introduce another variable such as *Up* or *Down* in order to follow the boundary of three-dimensional obstacles in a desirable manner.

However, for the sake of simplicity (and convergence), we propose an approach which directly uses $\mathscr{A}_{2\mathrm{D}}$ with minor modifications. As opposed to the definition of $\mathbf{\Omega}$, where we only consider obstacles on the planes $\mathbf{X}_z$ parallel to the ground, we now have to define another set $\tilde{\mathbf{\Omega}}$ of obstacles on the planes not necessarily parallel to the ground and therefore need an additional parameter θ. To this end, we consider a two-dimensional plane $\mathbf{X}_z^{\theta}(w_1, w_2, w_3)$ passing through any specified three points $w_1, w_2, w_3 \in \mathbf{X} = [0,1] \times [0,1] \times [0,1]$. The superscript θ of $\mathbf{X}_z^{\theta}$ represents a unique rotation of $\mathbf{X}_z = [0,1] \times [0,1] \times z$ with the (sequence of) rotational angle(s) θ, i.e. the differential *attitude* of $\mathbf{X}_z^{\theta}$ with respect to $\mathbf{X}_z$. Recalling the partition introduced in Section 2, one can still construct a square cell $\mathbf{C}_w$ $(\subset \mathbf{X}_z^{\theta})$ to which $w (\in \mathbf{X}_z^{\theta})$ belongs. This construction leads to another set of obstacles, which is

$$\tilde{\mathbf{\Omega}} = \bigcup_{z,\theta} \mathbf{\Omega}_z^{\theta}, \tag{4}$$

where $\mathbf{\Omega}_z^{\theta} = \{w \in \mathbf{X}_z^{\theta} \mid P(\mathbf{C}_w) > \rho\}$.

The path planning problem $\mathscr{P}_{2\mathrm{D}}$ can be modified accordingly as follows.

$\mathscr{P}_{3\mathrm{D}}$: Given UAV staring and target positions $w_{\mathrm{s}}, w_{\mathrm{t}} \in \mathbf{X} \cap \tilde{\mathbf{\Omega}}^{\mathrm{c}}$, where $\mathbf{X} = [0,1] \times [0,1] \times [0,1]$, $\rho, \beta > 0$ and $\tilde{\mathbf{\Omega}}^{\mathrm{c}}$ is the complement of $\tilde{\mathbf{\Omega}}$ defined in (4), find a sequence $\{w_l\}_{l=1}^{n}$ of waypoints such that $w_1 = w_{\mathrm{s}}$, $w_n = w_{\mathrm{t}}$, $v_j \notin \tilde{\mathbf{\Omega}}$, where $v_j = t w_j + (1-t) w_{j+1} (0 \leqslant t \leqslant 1)$ for $j = 1, 2, \ldots, n-1$ $(n \geqslant 2)$, and $\|w_j - w_{j+1}\| \geqslant (1 - \sqrt{2}/2)\beta$ for $j = 1, 2, \ldots, n-2$ $(n > 2)$.

The algorithm $\mathscr{A}_{3\mathrm{D}}$ that attempts to solve $\mathscr{P}_{3\mathrm{D}}$ follows the basic principles adopted in $\mathscr{A}_{2\mathrm{D}}$. The difference is that $\mathscr{A}_{3\mathrm{D}}$ utilizes multiple two-dimensional planes $\mathbf{X}_z^{\theta}$, as opposed to a single plane $\mathbf{X}_z$. It is initiated by selecting an operational plane $\mathbf{X}_{z_1}^{\theta_1}$ passing through the two points w_{s} and w_{t} and another intermediate point w_{int}, on which $\mathscr{A}_{2\mathrm{D}}$ is executed as a subroutine of $\mathscr{A}_{3\mathrm{D}}$. The point w_{int} is chosen such that $\mathbf{X}_{z_1}^{\theta_1}$ has at least a trivial path joining w_{s} and w_{t}. For example, if $w_{\mathrm{int}} = (x_0, y_0, z_0)$ has a large value of z_0, a UAV may increase its altitude to find a feasible path, as well as staying on $\mathbf{X}_{z_1}^{\theta_1}$.[6] At each instance of finding the best safe cell $\mathbf{C}_{\mathrm{s}}$ (see the transition condition in **Phase I** of $\mathscr{A}_{2\mathrm{D}}$) at the current waypoint w_k, we seek such a $\mathbf{C}_{\mathrm{s}}$ among $\mathbf{C}_{w_k}(z_1, \theta_1)$'s neighbouring cells, contained in $\mathbf{X}_{z_1}^{\theta_1}$ as well as $\mathbf{X}_{z_1+\beta}^{\theta_1}$ and $\mathbf{X}_{z_1-\beta}^{\theta_1}$. Here, $\mathbf{X}_{z_1+\beta}^{\theta_1}$ and $\mathbf{X}_{z_1-\beta}^{\theta_1}$ denote the planes which are parallel and have a distance of β with respect to $\mathbf{X}_{z_1}^{\theta_1}$ (see Fig. 6). When $\mathbf{C}_{\mathrm{s}}$ is selected, one could consider neighbouring *cubic* cells around w_k to accommodate the risk between planes. For example, instead of checking $P(\mathbf{C}_1)$ through $P(\mathbf{C}_{26})$ in Fig. 6, check $P(\mathbf{C}_1)$ through $P(\mathbf{C}_8)$, and further $P(\mathbf{C}_{ij})$ and $P(\mathbf{C}_{ik})$, where $i = 0, 1, \ldots, 8$, $j = 9, \ldots, 17$, $k = 18, \ldots, 26$, and $\mathbf{C}_{mn}$ is a cubic cell whose top and bottom square cells are $\mathbf{C}_m$ and $\mathbf{C}_n$. The risk $P(\mathbf{C}_{mn})$ may be then calculated as $\{P(\mathbf{C}_m) + P(\mathbf{C}_n) + P(\mathit{centre}(\mathbf{C}_{mn}))\}/3$. If $P(\mathbf{C}_{mn})$ is the best, then set $\mathbf{C}_{\mathrm{s}} := \mathbf{C}_n$.

The next operational plane switched to is $\mathbf{X}_{z_2}^{\theta_2}$, passing through $w_{k+1} = \mathit{centre}(\mathbf{C}_{\mathrm{s}})$, w_{t} and w_{int}, unless $\mathbf{C}_{\mathrm{s}} \subset \mathbf{X}_{z_1}^{\theta_1}$. Again, w_{int} is selected to guarantee the existence of a feasible path joining w_{k+1} and w_{t} in $\mathbf{X}_{z_2}^{\theta_2}$. As $\mathbf{X}_{z_2}^{\theta_2}$, after being partitioned, may have a different set of obstacles from $\mathbf{X}_{z_1}^{\theta_1}$, we

[6] We note that w_{int} can be easily identified in the UAV applications we are interested in where all sources of threat are ground-based, and thus the probabilistic risk is typically inversely proportional to the flying altitude (see Eq. (2)).

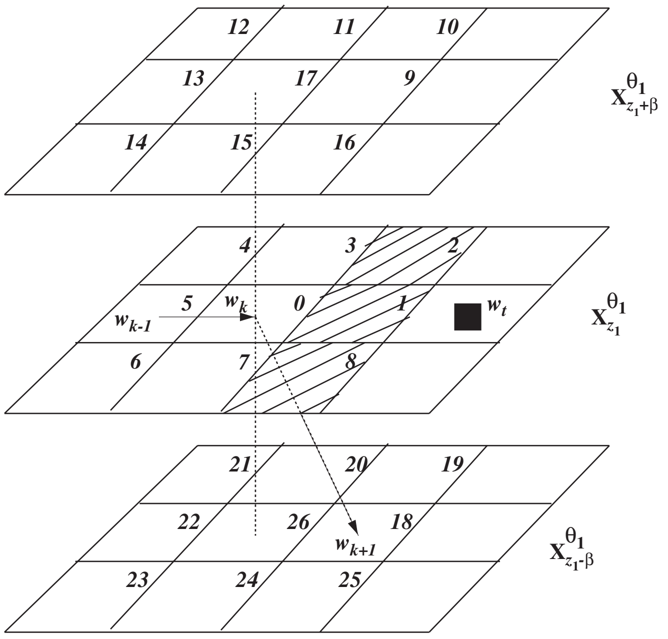
12
11
10
13
17
9
$\mathbf{X}^{\theta_1}_{z_1+\beta}$
14
15
16
4
3
2
5
w_k
0
1
w_t
w_{k-1}
$\mathbf{X}^{\theta_1}_{z_1}$
6
7
8
21
20
19
22
26
18
w_{k+1}
$\mathbf{X}^{\theta_1}_{z_1-\beta}$
23
24
25

Fig. 6. A waypoint selection in case of a free flight: the 18th cell is chosen as the best safe cell among 26 neighbouring cells around w_k; the next operational plane will be $\mathbf{X}^{\theta_2}_{z_2}$ passing through w_{k+1}, w_t and w_{int} for some θ_2 and z_2, where w_{int} can be any point which ensures a feasible path to w_t from w_{k+1} on $\mathbf{X}^{\theta_2}_{z_2}$; in this example, $w_{\text{int}} = w_k$.

therefore need to identify a new cell $\mathbf{C}_{w_{k+1}}$ on $\mathbf{X}^{\theta_2}_{z_2}$ (not on $\mathbf{X}^{\theta_1}_{z_1}$), set $\tilde{w} = centre(\mathbf{C}_{w_{k+1}})$ and check if $\mathbf{C}_{w_{k+1}}$ is safe, i.e. $\mathbf{C}_{w_{k+1}} \cap \mathbf{\Omega}^{\theta_2}_{z_2} \neq \Phi$. If not, the next best $\mathbf{C}_{\text{s}}$ should be checked until $\mathbf{C}_{w_{k+1}} \subset \mathbf{X}^{\theta_2}_{z_2}$ becomes safe, where we eventually reset $w_{k+1} := \tilde{w}$. Note that no plane change occurs, i.e. $z_2 = z_1$ and $\theta_2 = \theta_1$, if no safe $\mathbf{C}_{w_{k+1}} \notin \mathbf{X}^{\theta_1}_{z_1}$ is found. This change of planes continues until $\mathscr{A}_{\text{3D}}$ converges. Fig. 6 exemplifies waypoint generation using $\mathscr{A}_{\text{3D}}$. We note that the condition $v_j \notin \tilde{\mathbf{\Omega}}$ in $\mathscr{P}_{\text{3D}}$ may not be exactly satisfied with the waypoints generated by $\mathscr{A}_{\text{3D}}$. Instead, $\mathscr{A}_{\text{3D}}$ solves $\mathscr{P}_{\text{3D}}$ with $\tilde{\mathbf{\Omega}}$ replaced by $\hat{\mathbf{\Omega}} = \bigcup_{z,\theta \in \mathbf{V}} (\mathbf{\Omega}^{\theta}_{z})$, where $\mathbf{V}$ is the set of (z, θ) such that $\mathbf{X}^{\theta}_{z}$ is used during the execution of $\mathscr{A}_{\text{3D}}$. In addition, in order to ensure the guaranteed convergence of $\mathscr{A}_{\text{3D}}$, an array variable, such as $\mathbf{\Gamma}$ in **Phase II** of $\mathscr{A}_{\text{2D}}$, must be used to store points where the plane change occurs and the involved plane identifiers (z and θ), which can then be used to prohibit a repeat transition between the same pair of planes at the same transition point. We then have the following proposition on the convergence of $\mathscr{A}_{\text{3D}}$. A formal description of the foregoing three-dimensional extension can be found in Section 4.1.

Proposition 2. *Under the assumption that the intermediate point w_{int} can be easily selected, whenever the plane change occurs, say from $w_k \in \mathbf{X}^{\theta_i}_{z_i}$ to $w_{k+1} \in \mathbf{X}^{\theta_j}_{z_j}$, to guarantee a feasible path ($\subset \mathbf{X}^{\theta_j}_{z_j}$) from w_{k+1} to the target point w_{t}, a sequence of (safe) waypoints generated from the proposed algorithm $\mathscr{A}_{\text{3D}}$ converges.*

Proof. In view of Proposition 1, note that the feasible path guaranteed by the assumption can be actually detected by algorithm $\mathscr{A}_{\text{2D}}$. The constraint that a repeat plane transition between the same pair of planes at the same transition point is prohibited, implies the convergence of $\mathscr{A}_{\text{3D}}$ immediately. □

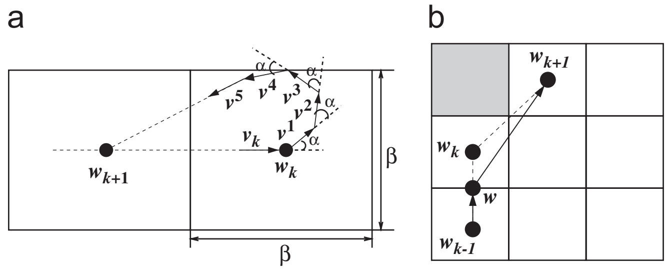

Fig. 7. Smooth transition from one to another waypoint. (a) A smooth transition from w_k to w_{k+1} taking into consideration UAV dynamics: the magnitude of intermediate velocities v^i's $(i = 1, 2, 3, 4)$ is the same as $V_{\min}$. (b) Another smooth transition from w_{k-1} to w_{k+1} taking the UAV dynamics into consideration.

3.4. Smooth path planning incorporating UAV dynamics: selection of β

A sparse partition (with large β) of the operational space leads to simple waypoint tracking, although the resultant path becomes conservative due to a possibly larger set of obstacles. More specifically, the value of β is often a function of $V_{\min}$ and α in UAV applications, where $V_{\min}$ is the minimum magnitude of the velocity of a UAV and α is the maximum turning angle. As depicted in Fig. 7(a), suppose a UAV approaches w_k with velocity v_k and its next waypoint has been chosen as w_{k+1}. We note that this is the worst situation in terms of the angle between w_k and w_{k+1}. In an ideal situation, the sequence $v^1, v^2, \ldots$ of intermediate velocities is preferable to a smooth transition to w_{k+1}. If a UAV is capable of changing its heading each Δt (s), then one can readily show that a safe flight is still guaranteed, provided $\beta \geqslant \beta_{\min}$, where $\beta_{\min} = 2V_{\min}\Delta t \sum_{i=1}^{n} \sin(n\alpha)$, for the greatest integer n satisfying $n\alpha \leqslant \pi$. For instance, when $V_{\min} = 50\,\text{m/s}$ and $\Delta t = 1\,\text{s}$, the values of β (km) are 1.1430, 0.5671 and 0.3732, respectively, for α (deg) equal to 10, 20 and 30. Moreover, one can further smooth a path if an obstacle cell is in a favourable location with respect to waypoints. Suppose a UAV is pointing towards a waypoint $w_k = centre(\mathbf{C}_{w_k})$. Assuming that a UAV is capable of checking the safety of all the neighbouring cells, within a sphere of radius 3β centred at the current position w, the UAV can calculate the next waypoint w_{k+1} immediately after it enters $\mathbf{C}_{w_k}$. Then, if the line segment $\overrightarrow{ww_{k+1}}$ does not cross any obstacle cell, it is desirable to point towards w_{k+1} directly, not via w_k (see Fig. 7(b)), which yields a smoother as well as faster path.

4. Utilizing more knowledge: dynamic threats and algorithm $\widetilde{\mathscr{A}}_{\text{3D}}$

4.1. Assumptions and algorithm description

Although the proposed algorithm has desirable convergence, limited use of information and memory may often lead to undesirably long flight times (see Fig. 10(a) for an illustration).

In this section, we improve the proposed algorithm in terms of flight time and its handling of dynamic threats by utilizing more information. To this end, we first introduce some new terminology. Given a two-dimensional operational plane and a partition over the plane, we define an *obstacle component* as a set of all *connected* static or *static-dependent* threats and the corresponding *obstacle component range* as the union of the obstacles due to all the connected static or static-dependent threats. By *connectedness* of two distinct threats Th_i and Th_j at i and j (for short, $Th_i \leftrightarrow Th_j$), we mean that there exists a continuous path $\mathbf{P}$ joining i and j such that every cell $\mathbf{C}_w$ containing a point w on $\mathbf{P}$ is dangerous (i.e. $P(\mathbf{C}_w) > \rho,\ \forall w \in \mathbf{P}$). By $Th_i \leftrightarrow \mathscr{OC}_k$, where Th_i is a threat and $\mathscr{OC}_k$ is an obstacle component, we mean that Th_i is connected to a threat(s) Th_j in $\mathscr{OC}_k$. A dynamic threat Th_d is called *static-dependent* on an obstacle component $\mathscr{OC}_i$ (or multiple obstacle components $\mathscr{OC}_j$ and $\mathscr{OC}_k$) if $Th_d \in \mathscr{OC}_i$ (or $Th_d \in \mathscr{OC}_j$ and $Th_d \in \mathscr{OC}_k$) at all time instances. We note that if the range of a dynamic threat straddles two obstacle components $\mathscr{OC}_1$ and $\mathscr{OC}_2$, by the definition of obstacle component, we consider $\mathscr{OC}_1$ and $\mathscr{OC}_2$ disjoint unless the dynamic threat is static-dependent on both of the obstacle components.The improved algorithm, denoted by $\widetilde{\mathscr{A}}_{3D}$, is going to update the obstacle component information whenever a new threat or threat's movement is sensed. We say that a UAV (or a waypoint) is *engaged* with a threat or an obstacle component, when the UAV follows (or the waypoint is chosen for the UAV to follow) the boundary of the obstacle due to the threat or the obstacle component.

We now introduce the following assumptions which facilitate a proof of the theoretical convergence of $\widetilde{\mathscr{A}}_{3D}$ as well as guaranteeing that the pointwise risk level is less than a pre-defined threshold value ρ throughout the UAV flight[7] :

(A) a UAV is capable of detecting the location and strength of threats within its observable range and capable of memorizing (and being informed of) the location and strength of already-detected threats;
(B) the UAV's observable range is larger than the range of any threat introduced in the operational space;
(C) pop-up or dynamic threats appear or behave slowly such that the UAV can fly from the current cell $\mathbf{C}$ to a safe neighbouring cell before $\mathbf{C}$ becomes dangerous due to the threats; this safety-related assumption could be removed if one does not require strict-obstacle-avoidance (i.e. $P(w) \leqslant \rho$ for some $\forall w \in \mathbf{P}$, where $\mathbf{P}$ is a chosen flight path); if the current UAV position becomes dangerous due to the UAV's short vision or the appearance of a pop-up threat, then the UAV can simply increase its altitude or move in a *good* direction until it finds a safe point where the main algorithm can be resumed; such a safe point always exists if all sources of threat have a risk characteristic expressed by formula (2);

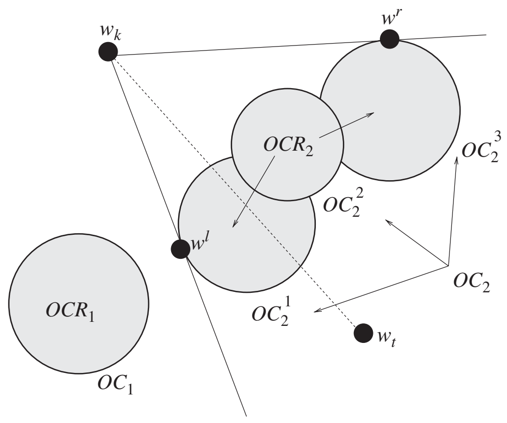

Fig. 8. Threat avoidance manoeuvre: since $\|\overrightarrow{w^l w_t}\| < \|\overrightarrow{w^r w_t}\|$, the UAV points towards w^l instead of w_t, unless w^l is on the boundary of the operating space where no feasible path exists and thus the UAV moves towards the other edge w^r; note that there are two obstacle components $\mathscr{OC}_1$ and $\mathscr{OC}_2$ and their corresponding ranges are $\mathscr{OCR}_1$ and $\mathscr{OCR}_2$; the second component $\mathscr{OC}_2$ has three connected threats $\mathscr{OC}_2^1$, $\mathscr{OC}_2^2$ and $\mathscr{OC}_2^3$.

(D) the ranges associated with dynamic threats are disjoint, i.e. no cell is within an obstacle region associated with more than one dynamic threat;
(E) there always exists a feasible path joining the UAV position and the target position at each time step;
(F) if $Th_d \leftrightarrow \mathscr{OC}_i$ (or $Th_d \leftrightarrow \mathscr{OC}_j$ and $Th_d \leftrightarrow \mathscr{OC}_k$) for more than fixed T time units, where Th_d is a dynamic threat and $\mathscr{OC}_i$ (or $\mathscr{OC}_j$ and $\mathscr{OC}_k$) is (are) an obstacle component(s), then Th_d is static-dependent on $\mathscr{OC}_i$ (or both $\mathscr{OC}_j$ and $\mathscr{OC}_k$); here, one time unit is the time needed for the UAV to generate a single waypoint.

Having these assumptions, the key ideas of $\widetilde{\mathscr{A}}_{3D}$ are the following:

(1) as a UAV is capable of detecting the location of threats within its sensor range, it is desirable to avoid the threats, if necessary, before it actually reaches them; therefore when $\overrightarrow{w_k w_t}$, where w_k and w_t are the current UAV and target positions, intersects any detected threat, the UAV points towards the closest *edge* of the corresponding obstacle component (including the intersected threat), instead of w_t, in terms of the line of sight angle (see Fig. 8);
(2) the transition from **Phase II** to **Phase I** of $\mathscr{A}_{2D}$ is now allowed only if the whole line segment joining w_k and w_t does not intersect the ranges of all the previously and currently engaged obstacle components; an array variable $\mathscr{COMP}$ stores the already engaged components in $\widetilde{\mathscr{A}}_{3D}$ (see Step II-1 in $\widetilde{\mathscr{A}}_{3D}$);
(3) the binary flag *Look* defined in Step I-3 of algorithm $\mathscr{A}_{2D}$ is set based on the first idea; for the example shown in Fig. 8, we set *Look* := *Left*, since $\overrightarrow{w_k w^l}$ can be found up to scaling by rotating the vector $\overrightarrow{w_k w_t}$ with respect to w_k clockwise; if w^r was the closest one, we would set *Look* := *Right* associated with an anti-clockwise search; note again that if

[7] Some of the following assumptions may not be valid in practice. However, we note that algorithm $\widetilde{\mathscr{A}}_{3D}$ shows successful performance even for numerous scenarios in which some of the assumptions fail (see Section 5.2).

w^l was outside the operational space, we would choose w^r and subsequently *Look* := *Right* (see the *Finding Closest Edge* routine in $\tilde{\mathscr{A}}_{3D}$); one may imagine a situation in which *Look* is flipped infinitely many times by repeatedly invoking the *Finding Closest Edge* routine when $\mathscr{OCR}_k$ changes in a pathological manner; however, this is not practically possible due to the assumption (C);

(4) when a UAV comes to know that it is going to meet the boundary of the operational plane in **Phase II**, we switch the obstacle-following direction from clockwise (anti-clockwise) to anti-clockwise (clockwise) (see Step II-3 in $\tilde{\mathscr{A}}_{3D}$);

(5) in **Phase II**, a UAV is not engaged with a dynamic threat unless the dynamic threat is static-dependent on the currently engaged obstacle component; in particular, this idea will be realized by the so-called *Passive Threat Avoidance* routine (see the end of algorithm $\tilde{\mathscr{A}}_{3D}$).

If the *Passive Threat Avoidance* procedure has been repeated to generate the last *T* waypoints, then the underlying dynamic threat is considered static-dependent on the currently engaged obstacle component by the assumption (F), and the UAV is subsequently commanded to actively get around the dynamic (now static-dependent) threat rather than to passively avoid it (see Steps II-2 and II-3 in $\tilde{\mathscr{A}}_{3D}$). We note that no phase change occurs when a UAV meets a dynamic threat in **Phase I** (see Step I-3-3 in $\tilde{\mathscr{A}}_{3D}$).

As a result, the ideas (1)–(5) can significantly reduce the UAV's total flight time by avoiding the undesirable *bouncing*s between **Phase I** and **Phase II** observed in algorithm $\mathscr{A}_{2D}$ or $\mathscr{A}_{3D}$ and furthermore efficiently handle dynamic threats while still guaranteeing the convergence of the algorithm. We now state the improved algorithm $\tilde{\mathscr{A}}_{3D}$:

Algorithm $\tilde{\mathscr{A}}_{3D}$.

Initialization: Set an index $k := 1$, variables $\mathbf{\Gamma} := \Phi$, $\mathbf{OC} := 0$, $\mathscr{COMP} := \Phi$ and the first waypoint $w_1 := w_s$. Define an operational plane $\mathbf{X}_z^\theta$ such that $w_s, w_{\text{int}}, w_t \in \mathbf{X}_z^\theta$, where $w_{\text{int}} = (x_0, y_0, z_0)$ is an arbitrary point with a large z_0. Partition $\mathbf{X}_z^\theta$ into disjoint square cells of length β and assign a unique number $id(\mathbf{C})$ to each cell $\mathbf{C}$. Define the number T_{ahead} of looking-ahead steps to be used for determining a binary variable *Look* and the maximum number T_{pta} of executions of the *Passive Threat Avoidance* routine, and set the associated counters t_{ahead} and t_{pta} to 0.

Update the obstacle component map: Construct the set of current obstacle components $\mathscr{OC} := \bigcup_j \mathscr{OC}_j := \bigcup_j \{\mathscr{OC}_j^1, \mathscr{OC}_j^2, \ldots\}$, where $\mathscr{OC}_x \cap \mathscr{OC}_y = \Phi (x, y = 1, 2, \ldots, x \neq y)$, based on the current UAV vision and update whenever any new threat or its movement is sensed or informed throughout the algorithm. Check which obstacle component or threat is currently engaged, i.e. $w_k \in \mathscr{OCR}_k$, where $\mathscr{OCR}_k$ is the range associated with $\mathscr{OC}_k$ and $\mathscr{OCR} := \bigcup_k \mathscr{OCR}_k$.

Phase I

```
Step I-1:   Define line segment $\overrightarrow{w_k w_t}$ joining $w_k$ and $w_t$.
Step I-2:   If $\|\overrightarrow{w_k w_t}\| \leqslant \beta$, define $\tilde{w}_1 := w_t$;
            else define $\tilde{w}_1 (\in \overrightarrow{w_k w_t})$ such that $\|\overrightarrow{w_k \tilde{w}_1}\| = \beta$.
Step I-3:   If $\overrightarrow{w_k \tilde{w}_1} \cap \mathscr{OCR} = \Phi$,
              if $\|\overrightarrow{w_k w_t}\| < \beta$, terminate the algorithm;
              if $\overrightarrow{w_k w_t} \cap \mathscr{OCR} \neq \Phi$,
                identify the first $\mathscr{OC}_k$ such that
                $\overrightarrow{w_k w_t} \cap \mathscr{OC}_k \neq \Phi$;
                execute Finding Closest Edge to obtain
                $\tilde{w}_t \in \mathscr{OCR}_k$ and Look;
                define $\tilde{w}_2 (\in \overrightarrow{w_k \tilde{w}_t})$ such that $\|\overrightarrow{w_k \tilde{w}_2}\| = \beta$;
                if $\overrightarrow{w_k \tilde{w}_2} \cap \mathscr{OCR} = \Phi$, set $w_{k+1} := \tilde{w}_2$;
                else set $w_{k+1} := \tilde{w}_1$;
              else set $w_{k+1} := \tilde{w}_1$;
              $k := k + 1$; goto Step I-1;
            else
Step I-3-1:   Execute Finding Desirable Cell to obtain $\mathbf{C}_s$;
              reset $w_k := centre(\mathbf{C}_{w_k})$; $w_{k+1} := centre(\mathbf{C}_s)$;
Step I-3-2:   If $(w_k, w_{k+1}) \in \mathbf{\Gamma}$,
                goto Step I-3-1; find the next desirable $\mathbf{C}_s$;
              else $\mathbf{\Gamma} := \mathbf{\Gamma} \cup (w_k, w_{k+1})$; find $\tilde{z}$ and $\tilde{\theta}$ such
                  that $w_{k+1}, w_{\text{int}}, w_t \in \mathbf{X}_{\tilde{z}}^{\tilde{\theta}}$;
                  partition $\mathbf{X}_{\tilde{z}}^{\tilde{\theta}}$ into disjoint square cells
                  of length $\beta$; identify $\tilde{w}, \mathbf{C}_s$ such that
                  $\tilde{w} = centre(\mathbf{C}_s)$, $w_{k+1} \in \mathbf{C}_s$, $\mathbf{C}_s \subset \mathbf{X}_{\tilde{z}}^{\tilde{\theta}}$;
                  set $z := \tilde{z}$ and $\theta := \tilde{\theta}$;
Step I-3-3: If $\mathbf{C}_s \cap \mathscr{OCR} \neq \Phi$,
              goto Step I-3-1; find the next desirable $\mathbf{C}_s$
              and follow the same procedure thereafter;
            else $w_{k+1} := \tilde{w}$;
              if $w_{k+1}$ is engaged with a dynamic threat,
                goto Step I-1 with $k := k + 1$;
              else goto Phase II with $k := k + 1$.
```

Phase II

```
Step II-1:  Identify the currently engaged threat $Th_k$
            and obstacle component $\mathscr{OC}_j$;
            set $\mathscr{COMP} := \mathscr{COMP} \cup \mathscr{OCR}_j$;
            if $\overrightarrow{w_k w_t} \cap \mathscr{COMP} = \Phi$,
                set $\mathbf{OC} := 0$; goto Step I-2.
Step II-2:  If $Th_k$ is static, $t_{\text{pta}} := 0$;
            else
                if $t_{\text{pta}} \leqslant T_{\text{pta}}$, $t_{\text{pta}} := t_{\text{pta}} + 1$;
                    execute Passive Threat Avoiding to
                    obtain $w_{k+1}$;
                    goto Step II-1 with $k := k + 1$;
                else $t_{\text{pta}} := 0$; $\mathscr{OC}_j := \mathscr{OC}_j \cup Th_k$.
Step II-3:  If Look is undefined yet,
                set Look:=Left; execute Which Way To Go
                to obtain DeadEnd, PhaseIpossible,
                $t_{\text{ahead}}$ and $w(t_{\text{ahead}})$;
                set $t_{\text{ahead}}^l := t_{\text{ahead}}$, $w^l := w(t_{\text{ahead}})$ and
                PhaseIpossibleL := PhaseIpossible;
                if DeadEnd = 1, Look := Right;
```

```
    else
      execute Which Way To Go to
      obtain DeadEnd, PhaseIpossible,
      $t_{\text{ahead}}$ and $w(t_{\text{ahead}})$;
      set $t^r_{\text{ahead}} := t_{\text{ahead}}$, $w^r := w(t_{\text{ahead}})$
      and PhaseIpossibleR :=
        PhaseIpossible;
      if PhaseIpossibleL = 1,
        if PhaseIpossibleR = 1 and
          $t^r_{\text{ahead}} < t^l_{\text{ahead}}$, Look := Right;
      else
        if PhaseIpossibleR = 1,
          Look := Right;
        else
          if $\|\overrightarrow{w^r w_t}\| < \|\overrightarrow{w^l w_t}\|$,
            Look := Right;
  else
    execute Which way to go to
    obtain DeadEnd;
    if DeadEnd = 1,
      flip Look (i.e. Look := Left if Look=
      Right, or Look := Left otherwise);
  execute Obstacle Following to obtain $\tilde{w}$;
  goto Step II-1 with $w_{k+1} := \tilde{w}$ and $k := k + 1$.
```

Subroutines

Finding Closest Edge: Given k, w_k, w_t, $\mathcal{OCR}_k$ and $\mathcal{OCR}$, this routine returns $\widetilde{w}_t \in \mathcal{OCR}_k$ and *Look*; find two *edges* w^r and w^l of $\mathcal{OCR}_k$ (see Fig. 8 for the definition of the two *edges*); if $\mathbf{C}_{w^r}$ (resp., $\mathbf{C}_{w^l}$) is one of the boundary cells of the operational space and $\mathbf{C}_{w^r}$ (resp., $\mathbf{C}_{w^l}$) $\cap \mathcal{OCR} \neq \Phi$, return $\widetilde{w}_t := w^l$ and *Look* := *Left* (resp., $\widetilde{w}_t := w^r$ and *Look* := *Right*); Otherwise, if $\|\overrightarrow{\widetilde{w}_t w^r}\| \leqslant \|\overrightarrow{\widetilde{w}_t w^l}\|$ (resp., $\|\overrightarrow{\widetilde{w}_t w^r}\| > \|\overrightarrow{\widetilde{w}_t w^l}\|$), return $\widetilde{w}_t := w^r$ and *Look* := *Right* (resp., $\widetilde{w}_t := w^l$ and *Look* := *Left*).

Finding Desirable Cell: Given w_k and X^θ_z, this routine returns $\mathbf{C}_s = \text{argmin}_{\mathbf{C} \in \mathcal{N}(w_k)} \|centre(\mathbf{C}) - w_t\|$, where $\mathcal{N}(w_k)$ is the set of safe neighbouring cells around w_k (see for illustration Fig. 6).

Which Way To Go: Given k, w_k, w_t, T_{ahead}, $\mathcal{COMP}$, $\mathcal{OCR}$ and *Look*, this routine returns two indicators *DeadEnd* and *PhaseIpossible* (both initially set to zero), t_{ahead} (the time steps elapsed after entering this routine) and $w(t_{\text{ahead}})$ (the waypoint associated with t_{ahead}); first, set the internal clock $t_{\text{ahead}} := 1$, $w(t_{\text{ahead}}) := w_k$ and invoke the *Obstacle Following* routine with $w_k := w(t_{\text{ahead}})$; if the returned $\tilde{w}$ from the *Obstacle Following* routine belongs to one of the boundary cells of the operational space, then find another boundary cell $\mathbf{C}_B$ immediately next to $\mathbf{C}_{\tilde{w}}$ as well as $\|\overrightarrow{centre(\mathbf{C}_B) w_t}\| < \|\overrightarrow{\tilde{w} w_t}\|$; if $\mathbf{C}_B \cap \mathcal{OCR} \neq \Phi$, terminate the routine with returning *DeadEnd* := 1, t_{ahead} and $w(t_{\text{ahead}}) := \tilde{w}$; if $\overrightarrow{\tilde{w} w_t} \cap \mathcal{COMP} = \Phi$, then terminate the routine with returning *PhaseIpossible* := 1, t_{ahead} and $w(t_{\text{ahead}}) := \tilde{w}$; otherwise, $t_{\text{ahead}} := t_{\text{ahead}} + 1$, $w(t_{\text{ahead}}) := \tilde{w}$ and repeat this routine until $t_{\text{ahead}} = T_{\text{ahead}}$.

Obstacle Following: Given w_k, *Look* and **OC**, this routine returns the next waypoint $\tilde{w}$ on the boundary of the current obstacle component range and updated **OC**; if **OC** = 0, then set **OC** to the first unsafe cell which intersects $\overrightarrow{w_k w_t}$; by performing an anti-clockwise (if *Look* = *Right*) or clockwise (if *Look* = *Left*) search starting from **OC**, find the first safe neighbouring cell $\mathbf{C}_s \in \mathbf{X}^\theta_z$ and the obstacle cell $\mathbf{C}_o \in \mathbf{X}^\theta_z$ immediately before $\mathbf{C}_s$ (see Fig. 4); return $\mathbf{OC} := id(\mathbf{C}_o)$ and $\tilde{w} := centre(\mathbf{C}_s)$.

Passive Threat Avoidance: Given w_k, Th_k and $\mathcal{OC}_j$, this routine returns the next waypoint w_{k+1}; if w_k is safe, $w_{k+1} := w_k$ (this assignment does not mean that a UAV stays at the current waypoint, i.e. the actual corresponding control action would be a turn around the current waypoint within the current cell, see Section 3.4); otherwise, $w_{k+1} := centre(\mathbf{C}_s)$, where $\mathbf{C}_s$ ($\subset \mathbf{X}^\theta_z$) is a safe neighbouring cell around w_k but still engaged with a static threat ($\in \mathcal{OC}_j$); in order to guarantee the existence of such a w_{k+1}, we need the assumptions (C)–(E).

4.2. Convergence of algorithm $\tilde{\mathscr{A}}_{3\text{D}}$

We first consider the simple case where all the introduced threats in the operational space are static.

Proposition 3. *Suppose that the assumptions* (A) *and* (B) *in Section* 4.1 *and the assumption of Proposition* 2 *hold. Then*, *if all the* (*finite number of*) *threats introduced in the UAV operational space are static*, *a sequence of* (*safe*) *waypoints generated from the proposed algorithm* $\tilde{\mathscr{A}}_{3\text{D}}$ *converges.*

Proof. By the reasoning given in the proof of Proposition 2, it is sufficient to provide a proof only for the case where the UAV operational plane **X** is two-dimensional. If the UAV does not meet any obstacle component, i.e. the algorithm never enters **Phase II**, then there is nothing to prove. Suppose the UAV is engaged with obstacle components in the order of $\{\mathcal{OC}\} := \mathcal{OC}_1, \mathcal{OC}_2, \ldots$, where $\mathcal{OC}_i \neq \mathcal{OC}_{i+1}$, $i = 1, 2, \ldots$, as it proceeds to the target w_t. We note that $\mathcal{OC}_j \subset \mathcal{OC}_{j+1}$ if the UAV is currently engaged with $\mathcal{OC}_j$ and detects a threat which is connected to some threat already in $\mathcal{OC}_j$. We then define $\mathbf{S}_j := \{w \mid \overrightarrow{w w_t} \cap (\bigcup_{i=1}^{j} \mathcal{OCR}_i) = \Phi\}$.

We note that all the points ($\in \mathcal{OCR}_j$) where **Phase II** ends and **Phase I** starts, belong to $\mathbf{S}_j$. The main steps in proving the convergence are to show that for any i, j: (1) $\mathbf{S}_j$ is not empty; (2) after the UAV is engaged with $\mathcal{OC}_j$, it reaches a point in $\mathbf{S}_j$ in finite time units; (3) $\mathcal{OC}_i \neq \mathcal{OC}_j$ $(i < j)$. If these can be shown, then the convergence follows from the finiteness of the number of threats (thus obstacle components). The first assertion is clear by the problem feasibility. For the second assertion, suppose the UAV is engaged with $\mathcal{OC}_j$. This means that the UAV basically keeps following the boundary of $\mathcal{OCR}_j$ in a fixed direction (clockwise or anti-clockwise), unless it reaches the boundary of **X** where the direction is reversed (see Step II-3 in $\tilde{\mathscr{A}}_{3\mathrm{D}}$). However, the direction reversal is allowed at most once, since the problem is feasible. Noting that $\mathscr{OCR}_j$ consists of a finite number of threats, the UAV then should be able to reach a point in $\mathbf{S}_j$ in finite time units after at most one direction reversion. The third assertion is implied by the rule employed in Step II-1 in $\tilde{\mathscr{A}}_{3\mathrm{D}}$. In fact, the UAV is disengaged from $\mathscr{OC}_j$ at w_k only if the whole line segment $\overrightarrow{w_k w_{\mathrm{t}}}$ is not blocked by any so far detected threat, i.e. $\overrightarrow{w_k w_{\mathrm{t}}} \cap (\bigcup_{i=1}^{j} \mathscr{OCR}_i) = \Phi$ ($\overrightarrow{w_k w_{\mathrm{t}}} \cap \mathscr{COMP} = \Phi$ in Step II-1 of $\tilde{\mathscr{A}}_{3\mathrm{D}}$). Hence, the UAV cannot be re-engaged with the same obstacle component thereafter. This completes the proof. □

We now proceed with a convergence proof for the case involving dynamic threats.

Proposition 4. *Suppose that all the assumptions* (A)–(F) *in Section* 4.1 *and the assumption of Proposition* 2 *hold. Then, as long as the number of threats introduced in the UAV operational space is finite, a sequence of* (*safe*) *waypoints generated from the proposed algorithm* $\tilde{\mathscr{A}}_{3\mathrm{D}}$ *converges.*

Proof. We again provide a proof only for the two-dimensional case. We recall and prove the three assertions used to prove Proposition 3. We stress that $\{\mathscr{OC}\}$ is now the sequence of obstacle components consisting of static or static-dependent threats. We note that the set $\mathbf{S}_j$ possibly changes as $\mathscr{OCR}_j$ does. The first assertion clearly holds by the assumption (E).

For the second assertion, we first note that the UAV is not transferred to **Phase II** when it meets a single dynamic threat in **Phase I**. Instead, the UAV avoids the threat by finding the best safe (*desirable*) cell, as described in Step I-3-1 of $\tilde{\mathscr{A}}_{3\mathrm{D}}$. This scheme allows the UAV to avoid the threat by the assumption (C), unless a static threat Th_{s} ($\in \mathscr{OC}_j$) is newly detected (then the UAV is now engaged with $\mathscr{OC}_j$).[8] Algorithm $\tilde{\mathscr{A}}_{3\mathrm{D}}$ then allows the UAV to follow the same obstacle following principle as for $\mathscr{A}_{2\mathrm{D}}$ or $\mathscr{A}_{3\mathrm{D}}$ and to reach a point in $\mathbf{S}_j$, unless the UAV detects another new dynamic threat Th_d (connected to $\mathscr{OC}_j$) and has to follow the boundary of the threat's range (note that the UAV is still engaged with $\mathscr{OC}_j$). Once the UAV meets Th_d and avoids it passively for at most a fixed finite time unit (T_{pta} in $\tilde{\mathscr{A}}_{3\mathrm{D}}$) by Step II-2 of $\tilde{\mathscr{A}}_{3\mathrm{D}}$, it can determine whether Th_d is static-dependent or not. If Th_d turns out to be static-dependent, the UAV is then engaged with $\mathscr{OC}_{j+1} \equiv Th_d \cup \mathscr{OC}_j$ and eventually reaches a point in $\mathbf{S}_{j+1}$ in finite time units by Step II-3 of $\tilde{\mathscr{A}}_{3\mathrm{D}}$, unless no further threats are detected. If Th_d was not static-dependent, the UAV would clearly reach a point in $\mathbf{S}_{j+1}$ by passively avoiding it in finite time units, unless no further threats were detected.

The third assertion holds by the same reasoning given in the proof of Proposition 3, unless a dynamic threat Th_d acts as a bridge between two obstacle components $\mathscr{OC}_i$ and $\mathscr{OC}_j$, which results in allowing the UAV to visit the same obstacle component again. However, by the assumption (F) and the *Passive Threat Avoidance* routine, the UAV would not be engaged with Th_d unless Th_d is static-dependent on both of the two components. If Th_d turns out to be static-dependent on both $\mathscr{OC}_i$ and $\mathscr{OC}_j$, the UAV treats $\mathscr{OC}_i$, $\mathscr{OC}_j$ and Th_d as one obstacle component $\mathscr{OC}_k$ and thus $\mathscr{OC}_k \neq \mathscr{OC}_i \neq \mathscr{OC}_j$. This completes the proof. □

5. Numerical examples

In this section, we present test examples to show the feasibility and efficiency of the proposed algorithms. We first test the initial algorithm $\mathscr{A}_{3\mathrm{D}}$ with static threats, and then test $\tilde{\mathscr{A}}_{3\mathrm{D}}$ with static, pop-up and dynamic threats. For the purpose of visualization, we only consider a single UAV (point-mass) flying with a constant speed of 50 m/s at a fixed altitude (2 km) over various hostile regions where the pointwise risk is quantified (estimated) through the formula (2).[9] We note that the two-dimensional scenario is more challenging than the three-dimensional scenario in the sense that the former allows for a UAV to choose a path among much more options than the latter and, as a consequence, is likely to be a nonchallenging problem for the UAV.

5.1. Scenarios with static threats

For extensive simulations, 100 sets of data including the number ($\in$ [5, 10]), the location, and the strength (range) of threats (missiles) have been randomly generated. In each of these cases, the fixed starting and target points of the UAV are safe. The operational space, the starting and target points of the UAV, the threshold of being an obstacle, and the size of cells are chosen as $\mathbf{X}=[0, 200]\times[0, 200]\times 2$ (km), $w_{\mathrm{s}}=(20, 20, 2)$ (km) and $w_{\mathrm{t}}=(180, 180, 2)$ (km), $\rho=0.08$, and $\beta=2$ km, respectively, for every simulation. For the simulation with each set of data, we measure the following performance indices: (a) the peak risk during the course of the simulation; (b) the maximum time to compute a waypoint; (c) the logarithm of the total flight length.

We first note from Fig. 9(a) that the proposed algorithm converges to the target point while maintaining the pointwise risk almost less than $\rho = 0.08$. Slight violations of the threshold limit are due to the approximation of the risk formula (1). The 50th and the 65th cases were found to be infeasible, and the corresponding measured quantities are set to zero to indicate no solution generated, for convenience. Fig. 9(b) illustrates the maximum computational cost of generating waypoints in each scenario, and shows that the proposed algorithm produces a waypoint in negligible time using a personal computer equipped

[8] We note that the UAV cannot meet another dynamic threat by the assumption (D).

[9] We are here only interested in finding safe waypoints. Although the parameter β in Section 3.4 can be chosen to facilitate following the determined waypoints, we are not actually synthesizing a controller for a UAV with specific dynamics at this time.

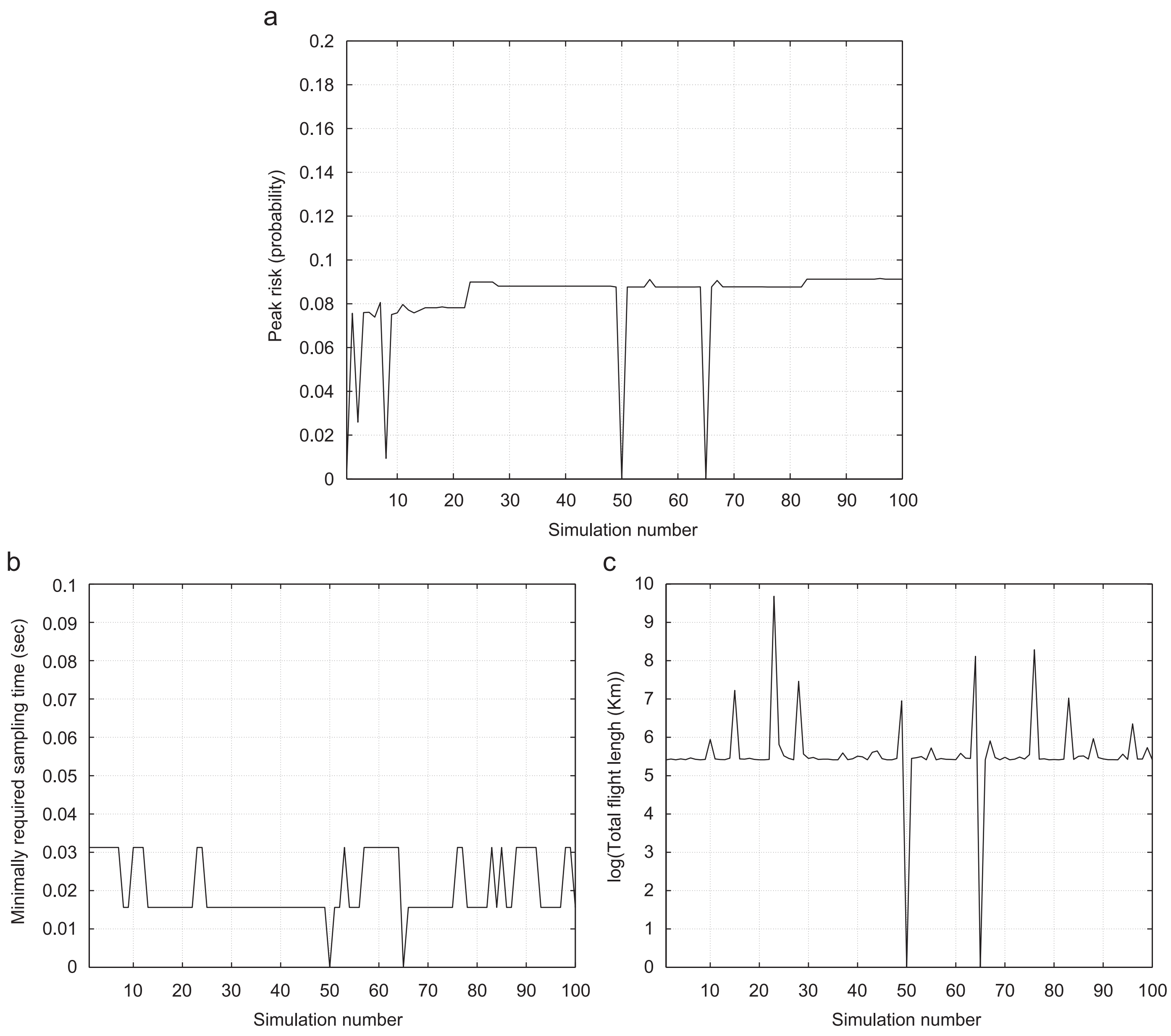

Fig. 9. The results for the UAV travelling along the 100 paths generated from algorithm $\tilde{\mathscr{A}}_{3D}$. (a) The peak risks. (b) The maximum time to compute a waypoint. (c) The total flight length.

with an Intel(R) Pentium 4 CPU 3.40 GHz. Fig. 9(c) shows the total flight length for each scenario. We note that most of the cases show reasonable performance, but some, especially the 23rd, require a large amount of time to converge. To consider this point further, the flight path generated from the algorithm with the 23rd set of data is presented in Fig. 10(a). In the figure, 10 missiles, marked as "x", are deployed in the operational space. The outermost circles around the locations of the missiles represent the set of points w, where $P(w) = \rho = 0.08$. As noticed in the figure, there exists a feasible path along which the UAV can reach the target point. Unfortunately, however, the UAV first misses the path, returning to it only after a great deal of time has elapsed. We note that this undesirable situation seems inevitable to a vehicle which has only local information on the environment but pursues a safe path. One possible remedy for this would be to dynamically change (increase, in this example) the value of ρ, which subsequently results in changing the size of the set $\mathbf{\Omega}$ of (3) of defining obstacles, so that an alternative path could be obtained (see Fig. 10(b)).

5.2. *Scenarios with static, pop-up and dynamic threats*

As already mentioned before, $\mathscr{A}_{3D}$ has a critical drawback in terms of its undesirable convergence behaviour, as illustrated in Fig. 10(a), because of its limited use of information and

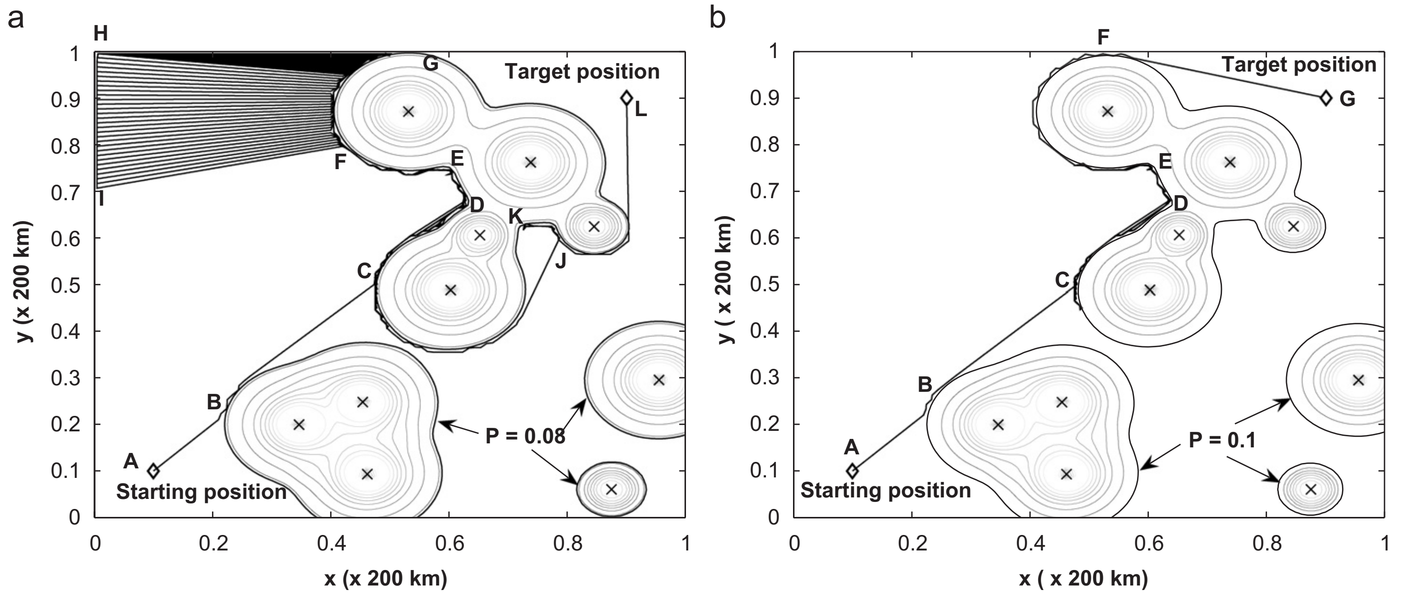

Fig. 10. The flight paths for the 23rd case. (a) When $\rho = 0.08$: The path starts from "A" and ends at "L", via "B", "C", "D", "E", "F", "G", "H", "G", "I", "F", "E", "D", "C", "J", "K" and "J ". Note that the path from "G" to "I" involves numerous transitions between **Phase I** and **Phase II**. As the UAV approaches "F" from "I", it finally turns its heading towards "E", not "G", because at "F" this choice minimizes the distance to the target point. (b) When $\rho = 0.1$: As opposed to the previous case, there exists another safe path passing through "F" as a result of increasing the value of ρ.

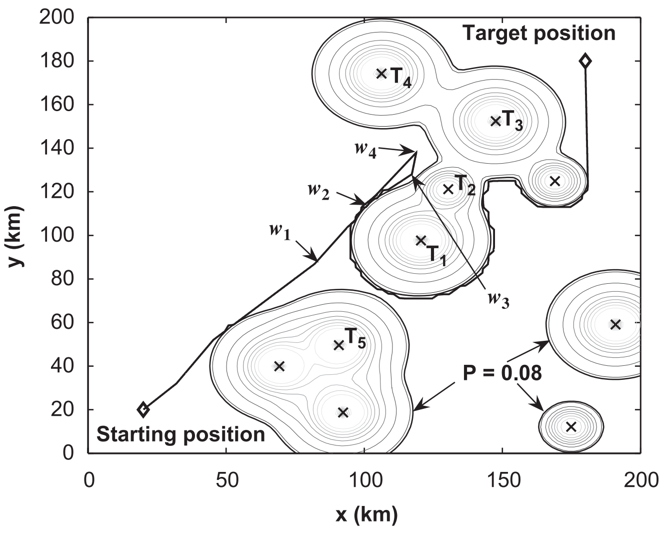

Fig. 11. Feasible path (solid black line) finding via $\tilde{\mathscr{A}}_{3D}$ when $\rho = 0.08$.

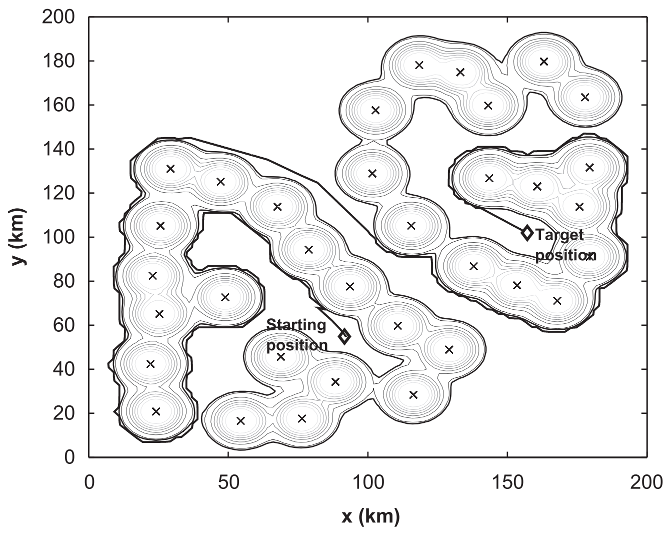

Fig. 12. An example having a *double bug trap* and a feasible path (solid black line) found via $\tilde{\mathscr{A}}_{3D}$ when $\rho = 0.08$.

memory. In this section, we demonstrate how $\tilde{\mathscr{A}}_{3D}$ can improve $\mathscr{A}_{3D}$ in terms of total flight time and handling dynamic threats by utilizing more information and memory. To begin with, we revisit the 23rd case identified in the previous section. As opposed to Fig. 10(a), Fig. 11 shows that with $\tilde{\mathscr{A}}_{3D}$ the UAV can find a feasible path faster than using $\mathscr{A}_{3D}$ with no need for increasing the value of ρ. When the UAV detects the threat T_1 at w_1, it first misses the feasible channel between T_1 and T_5 as before to move towards the closest edge w_2 of the range of T_1. The UAV lands on the range of T_1 at w_2, detects T_2 and follows the boundary of its range just for a while, after which it points towards the target point until reaches w_3, where T_3 is detected. At w_3 (respectively, w_4), the UAV moves towards the closest edge of the obstacle component range associated with T_1, T_2 and T_3 (respectively, T_1, T_2, T_3 and T_4). Note that at w_4, the UAV moves back to reach the closest edge nearby w_2 and subsequently finds the previously missed feasible path, rather than chooses the other edge associated with T_4 whose whole range is not within the operational space. Another example shown in Fig. 12 particularly addresses so-called a "double bug trap", as mentioned in Section 1, and is meant to be a hard problem for the standard rapid exploring random tree (RRT)

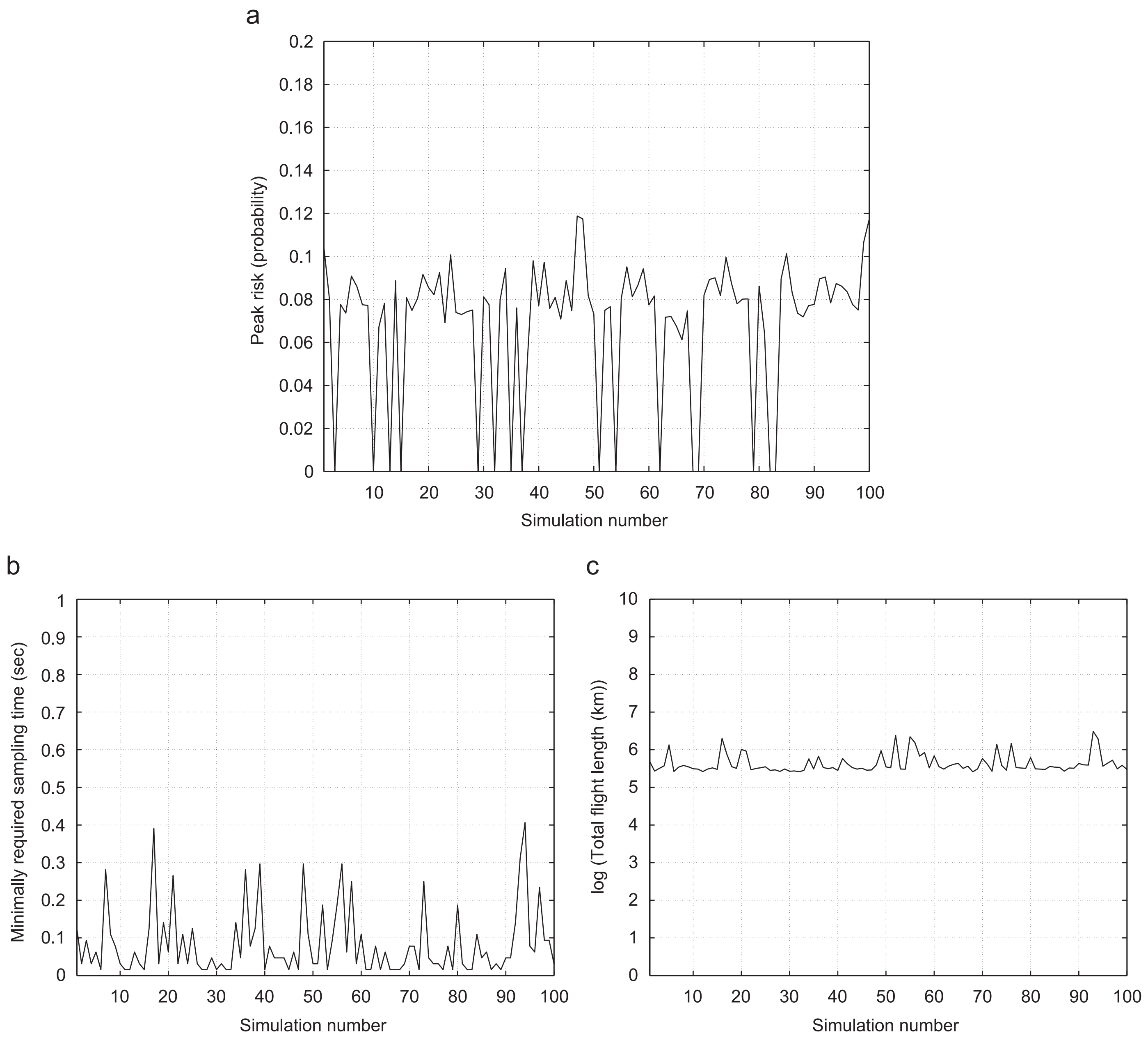

Fig. 13. The results for the UAV travelling along the 100 paths generated from algorithm $\tilde{\mathscr{A}}_{3\mathrm{D}}$. (a) The peak risks. (b) The maximum time to compute a waypoint. (c) The total flight length.

approach. However, as illustrated in the figure, $\tilde{\mathscr{A}}_{3\mathrm{D}}$ quickly finds a solution with no trouble.

We now proceed with testing $\tilde{\mathscr{A}}_{3\mathrm{D}}$ for other various scenarios. We use the same parameters (the operational space, the starting and target points of the UAV, the threshold of being an obstacle and the size of cells) as before, but assign the new parameters T_{ahead} and T_{pta} of $\tilde{\mathscr{A}}_{3\mathrm{D}}$ to 1 and 5, respectively. We then randomly create 100 different sets of data including the number, the moving direction (no direction for static and pop-up threats), the location and the strength of static, pop-up and dynamic threats. We assume that the UAV is capable of detecting threats within 40 km and the maximum range of threats is 7 km or 25 km. The numbers of static, pop-up and dynamic threats are limited up to 5, 2 and 3, respectively, and all the threats are generated such that the assumptions (A)–(C) (but not necessarily (D)–(F)) in Section 4.1 are strictly satisfied. Each dynamic threat is modelled such that it moves in a fixed random direction and returns to the original position with a constant speed of 5 m/s, and repeats the movement with a period of 4800 s. Pop-up (static) threats are generated with a probability of 1/(2000 s), and are placed at the midpoint of the current UAV and its target positions. As a result, one can observe from Figs. 13(a)–(c) that (i) the overall total flight length has been maintained relatively small for all the 100 cases (no undesirable peaks are observed any more); (ii) it requires slightly more computational power to generate a waypoint than $\mathscr{A}_{3\mathrm{D}}$

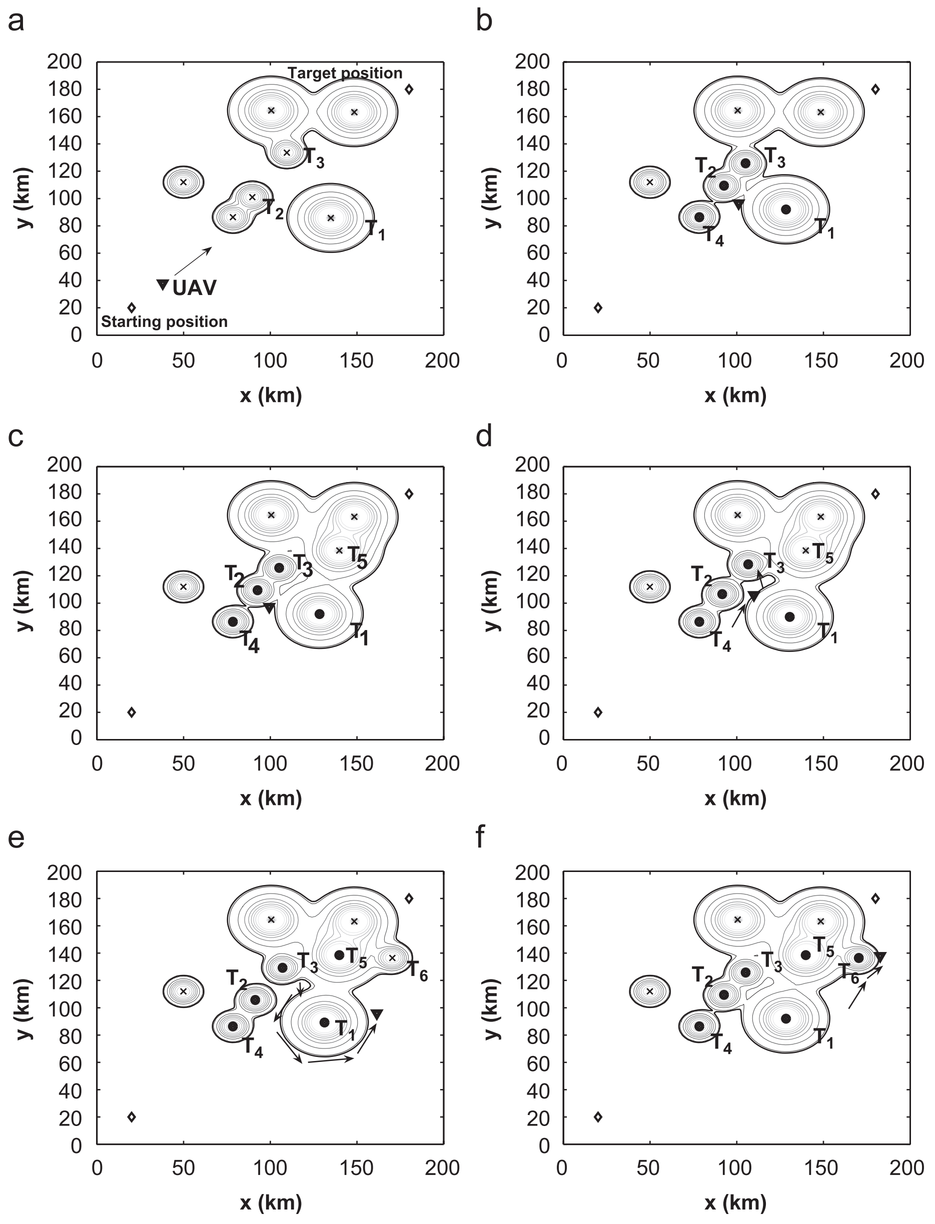

Fig. 14. The positions of the UAV and threats at various time instances for the 20th case. (a) At 508 s. (b) At 2324 s. (c) At 2532 s. (d) At 3156 s. (e) At 6304 s. (f) At 7328 s.

(note that the maximum time to compute a waypoint is less than 0.4 s)[10] ; (iii) the pointwise risks are still maintained at almost less than $\rho = 0.08$ as seen in Fig. 9(a); the figure also indicates that many solution paths have zero peak risk.

We now demonstrate one challenging scenario (the 20th case) from the above 100 data sets to show how $\tilde{\mathscr{A}}_{3\mathrm{D}}$ copes with dynamic and pop-up threats. Fig. 14(a) shows the positions of the UAV and threats at 508 s. The solid black circles in each of Figs. 14(b)–(f) represent the threats that have been detected by the UAV up until the given time instance. We note that threats T_1, T_2 and T_3 are dynamic and they move towards almost the same point, as seen in Fig. 14(b), which results in blocking the passage which was feasible at 508 s. At 2532 s, a pop-up threat T_5 appears and the UAV keeps approaching the intersection area of the ranges associated with T_1 and T_2. The UAV stays close to the intersection area for a while, and eventually passes through the passage between T_1 and T_2 when the threats start

[10] The waypoint computation time is closely related to algorithm implementation issues. For example, in our actual algorithm implementation computing the edges w^r and w^l in Fig. 8 is preceded by the time-consuming procedure in which one sweeps the area $\mathscr{OCR}_2$ by rotating vector $\overrightarrow{w_k w_t}$ clockwise as well as anti-clockwise with respect to w_k. Here, the number of points or lines used for this purpose compromises between the computation time and the exactness of w^r and w^l. If a much faster sampling time was required, one could use simple geometry to find the edges $\widetilde{w}_t$. A similar argument can be also applied to checking the connectedness of threats when the obstacle component map needs to be updated. Another possible way of reducing the computation time is to use a database memorizing the safety status of static cells. This avoids the unnecessary procedure, which is actually included in our algorithm implementation, of repeatedly checking the safety of the same cell.

moving back to where they were at 508 s (see Fig. 14(d)). After $t = 3156$ s, the UAV detects T_5 and flies to the left with respect to the vector to the target by invoking *Finding Closest Edge* routine (note that the UAV has detected T_1–T_5 so far, and T_1, T_2 and T_3 do not belong to the obstacle component to which T_5 belongs, for T_1, T_2 and T_3 are neither static nor static-dependent), and subsequently meets T_3 and starts following the boundary of the range associated with T_5 and then T_1. At some time between 3156 and 6304 s, a new pop-up threat T_6 appears and the UAV reaches the same point as at 3156 s and eventually reaches the target after avoiding T_6. We note that the algorithm has been able to generate the sequence of safewaypoints even after the assumption (D) in Section 4.1 failed at 2324 s.

6. Concluding remarks and future work

We have introduced new real-time path planning schemes which use limited information for fully autonomous UAVs. We proposed two algorithms of which one uses extremely limited information (the probabilistic risk of the surrounding cells with respect to the current UAV position) and memory, and the other utilizes more knowledge and memory (the locations and strengths of threats within the UAV's visible range). Both algorithms provably converge to a given target point and produce safe waypoints whose risk is almost less than a given threshold value. In particular, we have characterized a class of dynamic threats (so-called, static-dependent threats) which can be efficiently handled by algorithm $\tilde{\mathscr{A}}_{3D}$ whilst guaranteeing convergence to a given target. As illustrated via extensive numerical examples, more information leads to faster convergence, and some of the assumptions made in the paper may not affect the algorithm convergence. Finally, we note that the algorithms cannot yet guarantee the UAV's total flight length less than a certain desirable limit, although a smart choice of the risk threshold value ρ can partially achieve this purpose.

Acknowledgements

This research has been supported by the UK Engineering and Physical Sciences Research Council and BAE Systems.

References

Beard, R., McLain, T., Goodrich, M., & Anderson, E. (2002). Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, *18*, 911–922.

Brock, O., & Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *IEEE international conference on robotics and automation* (pp. 341–346), Detroit, May 1999.

Canny, J. (1988). *The complexity of robot motion planning*. Boston: MIT Press.

Dogan, A. (2003). Probabilistic approach in path planning for uavs. In *IEEE international symposium on intelligent control* (pp. 608–613), Houston, October 2003.

Enright, J. J., & Frazzoli, E. (2005). Uav routing in a stochastic, time-varying environment. In *IFAC World Congress*, Prague, July 2005.

Gu, D.-W., Kamal, W. A., & Postlethwaite, I. (2004). A uav waypoint generator. In *AIAA 1st intelligent systems technical conference*, Chicago, September 2004.

Hwang, Y. K., & Ahuja, N. (1992). Gross motion planning—a survey. *ACM Computing Surveys*, *24*(3), 219–291.

Kamal, W. A., Gu, D.-W., & Postlethwaite, I. (2005a). A decentralized probabilistic framework for the path planning of autonomous vehicles. In *IFAC World Congress*, Prague, July 2005.

Kamal, W. A., Gu, D.-W., & Postlethwaite, I. (2005b). Milp and its application in flight path planning. In *IFAC World Congress*, Prague, July 2005.

Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*(4), 566–580.

Kim, J.-O., & Khosla, P. K. (1992). Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, *8*(3), 338–349.

Kim, Y., Mesbahi, M., & Hadaegh, F. Y. (2004). Multiple-spacecraft reconfigurations through collision avoidance, bouncing, stalemate. *Journal of Optimization Theory and Applications*, *122*(2), 323–343.

Koditschek, D. E., & Rimon, E. (1990). Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, *11*(1), 412–442.

Latombe, J. C. (1991). *Robot motion planning*. Boston: Kluwer Academic Publishers.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge: Cambridge University Press.

LaValle, S. M., & Kuffner, J. J. (1999). Randomized kinodynamic planning. In *IEEE international conference on robotics and automation* (pp. 473–479), Detroit, May 1999.

Lingelbach, F. (2005). *Path planning using probabilistic cell decomposition*. Ph.D. thesis, KTH Signaler Sensorer och System, Stockholm, 2005.

Nikolova, E., Brand, M., & Karger, D. R. (2006). Optimal route planning under uncertainty. In *International conference on automated planning and scheduling*, Cumbria, June 2006.

Nilim, A., El Ghaoui, L., & Duong, V. (2002). Robust dynamic routing of aircraft under uncertainty. In *Digital avionics systems conference*, Irvine, October 2002.

Pettersson, P. O., & Doherty, P. D. (2004). Probabilistic roadmap based path planning for an autonomous unmanned aerial vehicle. In *Workshop on connecting planning and theory with practice*, Whistler, June 2004.

Richard, A., Schouwenaars, T., How, J. P., & Feron, E. (2002). Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *AIAA Journal of Guidance Control and Dynamics*, *25*(4), 755–764.

Takahashi, O., & Schilling, R. J. (1989). Motion planning in a plane using generalized Voronoi diagrams. *IEEE Transactions on Robotics and Automation*, *5*(2), 143–150.

Tarjan, R. E. (1981). A unified approach to path problems. *Journal of the Association for Computing Machinery*, *28*(3), 577–593.

Waydo, S., & Murray, R. M. (2003). Vehicle motion planning using stream functions. In *IEEE international conference on robotics and automation*, Taipei, May 2003.

Zheng, C., Li, L., Xu, F., Sun, F., & Ding, M. (2005). Evolutionary route planner for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, *21*(4), 609–620.

Yoonsoo Kim was born in Goheung, Republic of Korea in 1974. He received his BSc, MSc and PhD in Aerospace Engineering from Inha University (Republic of Korea), the University of Minnesota (USA) and the University of Washington (USA), in 1999, 2001 and 2004, respectively.

His main research focus is to resolve several outstanding challenges arising in coordinated control for distributed systems. He received the 2003 Graduate Research Award from the AIAA (American Institute of Aeronautics and Astronautics) pacific-northwest chapter in recognition of his PhD research. From 2004 to 2007, he was a Post-doctoral Research Associate in the Control and Instrumentation Research Group of the

Department of Engineering at the University of Leicester. He then joined the University of Stellenbosch in South Africa as Senior Lecturer in 2007.

Da-Wei Gu graduated from Department of Mathematics, Fudan University, Shanghai, China, in 1979, and received MSc degree in Applied Mathematics from Shanghai Jiao Tong University, China, in 1981, and PhD degree in Control System Theory from Department of Electrical Engineering, Imperial College of Science and Technology, London, UK, in 1985. During 1981–1982, he was a Lecturer in Shanghai Jiao Tong University. He was a Postdoctoral Research Assistant in Department of Engineering Science, Oxford University, UK, from 1985 to 1989. In 1989, he was appointed to a University Lectureship in Department of Engineering at Leicester University, UK, and has since been promoted to a Senior Lecturer and to a Reader at Leicester.
His current research interests include robust and optimal control, optimization algorithms, numerical perturbation analysis and autonomous control, with applications in aerospace and manufacturing industry.
He has over 230 research publications, including two books. He is a Chartered Engineer.

Ian Postlethwaite was born in Wigan, England in 1953. He received a First Class BSc (Eng) degree from Imperial College, London University in 1975 and the PhD degree from Cambridge University in 1978.
From 1978 to 1981, he was a Research Fellow at Cambridge University and spent six months at General Electric Company, Schenectady, USA. In 1981, he was appointed to a Lectureship in Engineering Science at Oxford University. In 1988, he moved to a Chair of Engineering at the University of Leicester where he was Head of Department from 1995 to 2004 and is now a Pro-Vice-Chancellor. He has held visiting research positions at the Australian National University and the University of California at Berkeley.
His research involves theoretical contributions to the field of robust multivariable control and the application of advanced control system design to engineering systems. He is a Fellow of the Royal Academy of Engineering, the IEEE, the IET and the InstMC. In 1991, he received the IEE FC Williams premium; in 2001, he was awarded the Sir Harold Hartley Medal of the InstMC; and in 2002, he received a best paper prize from the IFAC Journal of Control Engineering Practice. He is a co-author with Sigurd Skogestad of Multivariable Feedback Control (Wiley, 1996 and 2005).