Robotics and Autonomous Systems 62 (2014) 142–150

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Flight path planning for unmanned aerial vehicles with landmark-based visual navigation

Luitpold Babel *

Institut für Mathematik und Informatik, Fakultät Betriebswirtschaft, Universität der Bundeswehr München, D-85579 Neubiberg, Germany

HIGHLIGHTS

- The flight performance and the navigational capabilities are considered.
- The airspace is discretized by a network depending on the UAV characteristics.
- The time-consuming tasks are performed in a preprocessing step.
- It is demonstrated that a flight path is calculated within few seconds.

ARTICLE INFO

Article history:
Received 6 March 2013
Received in revised form
31 October 2013
Accepted 22 November 2013
Available online 28 November 2013

Keywords:
Route planning
Kinodynamic planning
Unmanned aerial vehicles
Landmark-based visual navigation
Shortest paths in networks

ABSTRACT

In this paper we present an algorithm to determine a shortest trajectory of a fixed-wing UAV in scenarios with no-fly areas. The innovative feature is that not only the kinematic and dynamic properties, but also the navigational capabilities of the air vehicle are taken into account. We consider a UAV with landmark-based visual navigation, a technique which can cope with long-term GPS outages. A navigation update is obtained by matching onboard images of selected landmarks with internally stored geo-referenced images. To achieve regular updates, a set of landmarks must be identified which are passed by the air vehicle in a proper sequence and with appropriate overflight directions.

The algorithm is based on a discretization of the airspace by a specific network. Each path in the network corresponds to a trajectory which avoids the no-fly areas and respects the flight performance of the air vehicle. Full functionality of the navigation can be ensured by dynamically adapting the network to the environmental conditions. A shortest trajectory is then obtained by the application of standard network algorithms.

1. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly deployed in civilian and military domains. Civil applications include search and rescue, border interdiction, traffic monitoring, law enforcement, disaster and emergency management, wild fire suppression, communications relay, and many others. In the military field, UAVs play a key role within the concept of information dominance. They are widely used for intelligence, reconnaissance and surveillance missions. A survey of UAV applications is given by Sarris [1].

A UAV mission planning system is intended to assist an operator to plan and manage missions which satisfy the operational requirements while taking into account the limitations of the air vehicle, airspace control regulations, rules of engagement, etc. The main component of an automated mission planning system is the flight path algorithm. The task is to find an optimal or near-optimal route from a start point to a destination point in a complex and challenging scenario. This problem is further complicated by the need of a fast planning. Modern mission planning systems must support quick reactions to changing operational requirements, tactical considerations, or environmental conditions.

An advanced flight path algorithm should consider *differential constraints* which arise from the kinematic and the dynamic behavior of the air vehicle. Limitations of the velocity are often called *kinematic constraints* in order to distinguish them from *dynamic constraints* which refer to the acceleration capabilities. For an air vehicle, the kinematic constraints (first order derivative of motion) are usually expressed by its minimum and maximum velocity. The dynamic constraints (second order derivatives) reflect that the air vehicle is not able to instantaneously change its velocity or perform sharp turns. Motion planning problems with velocity and acceleration bounds are also known as *kinodynamic problems* (see Donald et al. [2]). Planning under differential constraints is intensively discussed in the textbook of LaValle [3].

Equally important, however, is to involve the *navigational capabilities* of the UAV. A successful mission of an autonomous air vehicle strongly depends on an accurate and reliable navigation.

* Tel.: +49 8960043267; fax: +49 8960043795.
E-mail address: luitpold.babel@unibw.de.

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.11.004

Navigation data (position, velocity, and attitude) are needed for guidance and control. They are usually obtained by inertial navigation systems which are based on measurements of the vehicle's angular velocity and acceleration. In order to correct the inevitable drift, the systems are aided by external non-inertial navigation data. These data are often provided by a satellite-based system such as the Global Positioning System (GPS). Since GPS is not always available, visual (image-based) navigation is establishing more and more as a cheap and robust additional measurement method. In a landmark-based approach, an onboard camera takes images of selected landmarks during flight. By matching these images with geo-referenced landmark data, the position and attitude of the air vehicle relative to the landmark can be estimated.

This paper investigates the problem of finding a flight path of minimal length in the horizontal plane, from a start point with release velocity vector to a destination point with approach velocity vector. The flight path must avoid no-fly zones (densely populated areas, threat regions, obstacles due to topography, etc.) and must consider the performance of the air vehicle. This means that the trajectory should be sufficiently smooth (obeying the kinematic and dynamic constraints of the air vehicle) and should ensure full functionality of the navigation system. The latter includes the identification of a set of suitable landmarks which have to be passed by the air vehicle in a proper sequence. The distances between consecutive landmarks and the overflight directions must be controlled in order to guarantee regular navigation updates. The problem is aggravated by the fact that the quality of navigation updates is strongly depending on (rapidly changing) weather conditions.

Finding a flight path which avoids no-fly zones is strongly related to the problem of finding a collision-free path of a robot in an environment with obstacles. Different techniques have been used to solve that problem, among them are mixed integer linear programs, the potential field method, cell decomposition, the roadmap method, the mass–spring–damper method, and several network-based approaches with discretizations of the space by visibility graphs, Voronoi diagrams or regular grids. We refer to the reviews of Latombe [4] and LaValle [3].

In adapting these methods to the flight path problem, some authors completely neglect the maneuverability of the air vehicle. In a number of papers, the flight performance is indeed considered, but in a rather simplified form. For instance, the network-based methods of Carlyle [5], Rippel [6] and Zabarankin [7] generate a polygonal path from a regular grid discretization of the airspace. The flight performance is restricted to the implementation of turn radius constraints. They allow consecutive edges on a path only if these edges include a sufficiently small angle. Other authors perform a path smoothing after having found a collision-free path in a first step. For instance, Bortoff [8] determines a shortest path in a network based on Voronoi polygons and achieves a smoothing of the path by a springs and masses approach. Judd and McLain [9] use a series of cubic splines to smooth the straight-line segments.

Newer approaches try to solve the flight path problem while, at the same time, accounting for no-fly zones and differential constraints. These include sampling-based methods based on rapidly exploring random trees, model predictive control methods, and mathematical programming methods which treat the route planning problem as a numerical optimization problem. A review is given by Goerzen et al. [10]. On the other hand, we are not aware of any research that also considers the navigational capabilities of the air vehicle.

In this paper we introduce a flight path algorithm which considers no-fly zones, differential and navigational constraints for a fixed-wing UAV equipped with land-mark-based visual navigation. The algorithm is based on a discretization of the airspace by a specific network. The edges of the network represent shortest obstacle-avoiding trajectories between the landmarks. Each of them respects the differential constraints which are given in the form of velocity and acceleration bounds. The generation of the trajectories adopts an idea from Babel [11]. By deleting certain edges, we obtain a subnetwork where the trajectories additionally comply with the navigational constraints. While the basic network is static, the subnetwork can be considered as dynamic in the sense that it depends on the environmental conditions of the current mission. The approach allows the application of standard network methods to find a shortest path. In order to realize a quick planning we split the algorithm into a preprocessing and an online planning phase. The preprocessing phase gathers the time-consuming preparatory work, thus allowing an online planning within a few seconds.

Section 2 briefly describes the concept of landmark-based visual navigation. Section 3 outlines how to find a shortest flight path with differential constraints in a scenario with no-fly zones. Section 4 contains the main contribution of the paper, the route planning algorithm for air vehicles with navigational constraints. Section 5 continues with implementation details. Computational results for different scenarios are discussed in Section 6. We conclude with final remarks in Section 7.

2. Landmark-based navigation

Today, most air vehicles are equipped with an inertial navigation system. The main component of an inertial navigation system is the inertial measurement unit (IMU), an electronic device which usually consists of three accelerometers and three gyroscopes. Measurements of the current rate of acceleration and changes in angular velocities (of pitch, roll and yaw) are integrated over time, thus providing an estimation of the velocity, orientation and position of the vehicle.

A major drawback of inertial navigation systems is that they suffer from integration drift. Small measurement errors of acceleration and angular velocity are integrated into increasing errors in velocity which are accumulated into still greater errors in position. This provides an ever-increasing difference between the estimated position of the air vehicle and its true position (see e.g. Britting [12], Titterton and Weston [13]).

For that reason the position must be periodically corrected by an input from some other source. This can be accomplished by a satellite-based system (such as the Global Positioning System GPS) or some other type of navigation system. An aided system fuses the inertial navigation data with the GPS data (or data from some other source), hence bounding the error and providing a higher degree of accuracy than is possible with the use of any single system (see e.g. Farrel and Barth [14], Grewal et al. [15]).

However, for UAVs with GPS-aided inertial navigation system a loss of GPS signals can be extremely problematic and, in the worst case, end up with a complete failure of the mission. Reasons for GPS outages might be that satellite services are temporarily unavailable or satellite signals are jammed. Recently, spoofing has also come into the focus of unmanned flight systems (a spoofing attack attempts to deceive a GPS receiver with manipulated GPS signals hence pretending a false position). Both jamming and spoofing have become a major concern of GPS users due to the easy availability of technology on the market (see e.g. Wright et al. [16]).

To overcome these problems, visual navigation is getting more and more common for aiding inertial navigation systems (for reviews of the applied techniques see e.g. Bonin-Font et al. [17], DeSouza and Kak [18]). Visual navigation can cope with long-term GPS outages. A significant advantage – compared to other active approaches – is that visual navigation does not send out any signals. Hence there is no danger of being detected by hostile

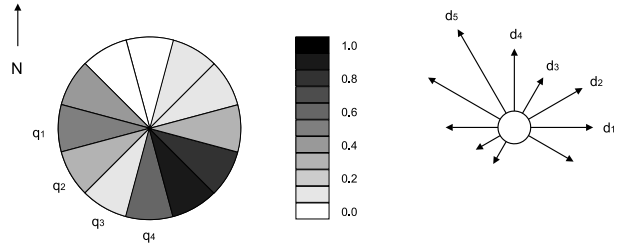

Fig. 1. Landmark rosette with quality values and associated flight distances.

forces. Moreover, it cannot be jammed since it is not guided by external signals. As a passive system it is less vulnerable to malicious actions.

In recent years, a number of unmanned aircraft systems were put into service which operate with landmark-based visual navigation. This method provides navigation updates in certain intervals. The idea is to estimate the position of the air vehicle by matching a sequence of onboard images to a geo-referenced image. The onboard images are taken from a downward looking infrared or visual camera which is mounted on the air vehicle. The geo-referenced images are stored in the onboard computer beforehand or downloaded during flight (see e.g. Cesetti et al. [19], Conte and Doherty [20]).

Landmarks can be either natural (e.g. rivers, shorelines, forests) or man-made (e.g. roads, railway tracks, crossings, significant buildings, bridges). They have a fixed and known position, relative to which the UAV can localize itself. The quality of a navigation update is depending on static and dynamic properties which includes

- the characteristics of the specific landmark (size, geometry, structure, material, texture, etc.)
- the encounter situation (overflight direction, flight altitude, camera's angle of view, etc.)
- and the environmental conditions (weather, rainfall, air humidity, temperature, visual range, etc.).

These properties are typically summarized in a landmark rosette (see Fig. 1). In the example, an approach from direction North yields quality value 0. This means that no navigation update is possible. An approach from direction South–South–East yields quality value 1 which corresponds to a perfect update. Reasons for a low quality value might be that parts of the landmark are masked (e.g. by a forest or a large building) or the perspective is extremely unfavorable.

According to the weather conditions, a landmark might have different quality values. For instance, if the navigation system is equipped with a visual camera then a high-quality landmark might become quite miserable if snowfall hides some of its characteristic features and wipes off the contrast to the background. For an infrared camera, a change of temperature can dramatically change the quality values. This provides different landmark rosettes for day and night, in sunshine or with overcast sky.

For the route planning of a UAV with landmark-based visual navigation, the intervals between two navigation updates (i.e. the distance between two consecutive landmarks on the flight path) must not be too large. Due to IMU drift, the navigation error increases with the length of the flight path. The next landmark might be outside the camera's field of view if the navigation error becomes too big and the estimated position of the air vehicle is far away from the real position. In this case the air vehicle gets lost.

We assume that the flight altitude over the landmarks is fixed. The maximal allowed flight distance $dist(L, \alpha, \omega)$ from a landmark L with overflight direction α to the next landmark depends on the quality value q of the landmark in direction α and the IMU drift, i.e.

$$dist(L, \alpha, \omega) = f(q(\alpha, \omega)).$$

The allowed flight distances associated to a landmark are illustrated in Fig. 1. Note that the quality values, and hence the flight distances, depend on the environmental conditions ω during the mission.

3. Flight path optimization in scenarios with no-fly zones

A trajectory will be called *feasible* if it avoids the no-fly areas and *flyable* if it respects the flight performance of the air vehicle. Finding a trajectory in a scenario with no-fly zones is an integral part of our flight path planning problem. The associated subproblem can be formulated as follows.

(RP) Find a feasible and flyable trajectory of minimal length from a start point P with specified velocity vector $\vec{v}_P$ to a destination point Q with velocity vector $\vec{v}_Q$.

This problem is referred to as the route planning problem (RP).

We represent the trajectory by a twice continuously differentiable parametrized curve

$$\gamma(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \tag{1}$$

in the plane with parameter range $t \in [0, T]$. The curve must comply with the start conditions

$$\gamma(0) = P, \qquad v(0) = \dot{\gamma}(0) = \vec{v}_P \tag{2}$$

and the destination conditions

$$\gamma(T) = Q, \qquad v(T) = \dot{\gamma}(T) = \vec{v}_Q. \tag{3}$$

For the acceleration in the endpoints we postulate

$$a(0) = \dot{v}(0) = \vec{0}, \qquad a(T) = \dot{v}(T) = \vec{0}. \tag{4}$$

The kinematic and dynamic capabilities of the air vehicle are modeled by the restrictions

$$v_{\min} \le |v(t)| \le v_{\max}, \quad t \in [0, T] \tag{5}$$

where $v_{\min}$ and $v_{\max}$ denote the minimal and maximal absolute velocity of the air vehicle, and

$$a_{\min} \le a_{long}(t) \le a_{\max}, \quad t \in [0, T] \tag{6}$$

$$a_{lat}(t) \le \hat{a}_{\max}, \quad t \in [0, T]. \tag{7}$$

Here $a(t)$ has been decoupled in longitudinal and lateral accelerations. $a_{\min}$ and $a_{\max}$ bound the (linear) deceleration and acceleration, respectively, and $\hat{a}_{\max}$ the perpendicular acceleration. Finally

$$\gamma(t) \notin Z_1, \ldots, Z_m, \quad t \in [0, T] \tag{8}$$

must hold where $Z_1, \ldots, Z_m$ denote the no-fly zones of the mission. The goal is to minimize the length of the trajectory, i.e.

$$\min \int_0^T |\dot{\gamma}(t)| dt.$$

For the solution of the problem (RP) we adapt an approach which has been introduced recently in [11]. The idea is to discretize the airspace by a very specific network. The network is characterized by the fact that each path in the network corresponds to a trajectory which respects the kinodynamic restrictions (5)–(7) of the air vehicle (such a trajectory is flyable) and which avoids the no-fly areas (a trajectory fulfilling (8) is feasible). This allows to apply standard network techniques, e.g. the algorithm of Dijkstra, the A*-algorithm, or any other method (see e.g. Cormen et al. [21]) to find a shortest path.

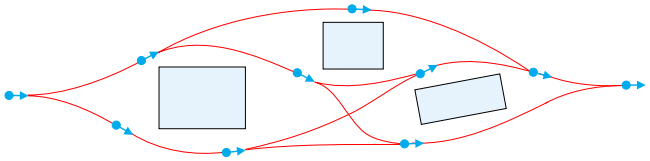

Fig. 2. Line elements with flight path segments and no-fly zones.

The generation of the network starts by randomly distributing a large number of line elements in the horizontal plane. A *line element* consists of a position in the plane together with a velocity vector whose length is equal to the nominal velocity v_0 of the air vehicle (in this sense, start and destination are also line elements with velocity vectors $\vec{v}_P$ and $\vec{v}_Q$). The second step is to connect pairs of line elements by flight path segments. These paths must be both flyable and feasible. The vertices of the network represent the line elements, and two vertices are linked by a directed edge if there is a flyable and feasible flight path segment connecting the corresponding line elements. The edge cost is defined as the length of the flight path segment. Here is a formal description of the algorithm.

```
Algorithm Find-Trajectory
Input:  Start P, destination Q, no-fly areas, flight performance data of UAV
Output: Flyable and feasible trajectory γ(t) from P to Q

CREATE vertices v_P for start P and v_Q for destination Q
V := {v_P, v_Q}, E := ∅
GENERATE line elements by randomly choosing position and direction
For each line element A
    CREATE a new vertex v_A, V := V ∪ {v_A}
end
For each pair of line elements (A, B)
    GENERATE a curve γ_AB(t) from A to B using quintic Hermite interpolation
    CHECK γ_AB(t) with respect to feasibility and flyability
    If γ_AB(t) is both feasible and flyable
        CREATE a new edge (v_A, v_B) with cost c(v_A, v_B) := ∫ |γ̇_AB(t)| dt,
        E := E ∪ {(v_A, v_B)}
    end
end
FIND a shortest path in the network G = (V, E, c) from v_P to v_Q
CONCATENATE the curves belonging to the path to a P–Q trajectory γ(t)
```

The main principle is sketched in Fig. 2. The start point of the air vehicle is located on the left, the destination point is on the right, release and approach directions are both 0°. The figure shows line elements (blue arrows) with some flyable and feasible flight path segments in a scenario with three no-fly zones.

The flight path segments are generated by an iterative procedure using quintic Hermite interpolation. Let A and B be two line elements with positions (x_A, y_A), (x_B, y_B) and velocity vectors $(v_{Ax}, v_{Ay})^T$, $(v_{Bx}, v_{By})^T$. We define the acceleration in each line element to be zero. A flight path connecting A and B must fulfill

$$x(0) = x_A, \qquad x(t_{\max}) = x_B$$
$$\dot{x}(0) = v_{Ax}, \qquad \dot{x}(t_{\max}) = v_{Bx}$$
$$\ddot{x}(0) = 0, \qquad \ddot{x}(t_{\max}) = 0$$

and

$$y(0) = y_A, \qquad y(t_{\max}) = y_B$$
$$\dot{y}(0) = v_{Ay}, \qquad \dot{y}(t_{\max}) = v_{By}$$
$$\ddot{y}(0) = 0, \qquad \ddot{y}(t_{\max}) = 0$$

where $t_{\max}$ defines the parameter range. If $x(t)$ and $y(t)$ are presumed to be polynomial functions then these six conditions require polynomials of degree 5. Let

$$x(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0.$$

For $t = 0$ we obtain

$$a_0 = x_A, \qquad a_1 = v_{Ax}, \qquad a_2 = 0.$$

The remaining coefficients are obtained for $t = t_{\max}$ as the solution of the linear system of equations

$$\begin{pmatrix} t_{\max}^3 & t_{\max}^4 & t_{\max}^5 \\ 3t_{\max}^2 & 4t_{\max}^3 & 5t_{\max}^4 \\ 6t_{\max} & 12t_{\max}^2 & 20t_{\max}^3 \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} x_B - x_A - v_{Ax}\, t_{\max} \\ v_{Bx} - v_{Ax} \\ 0 \end{pmatrix}.$$

The coefficients for $y(t)$ are calculated analogously.

The parameter range $t_{\max}$ is identified with the flight time along the curve. Since the flight time is not known in advance it is determined by an iterative procedure. The procedure starts by defining $t_{\max}$ as the flight time which is required for a straight flight (with nominal velocity v_0) from position A to position B. Each iteration provides a new curve (as described above). By numerical integration one obtains the length of this curve and – with v_0 – a new estimate of the flight time which can be used as parameter range $t_{\max}$ in the next iteration. Usually, only very few iterations are needed for a suitable parametrization of the curve. For more details we refer to [11].

Finally, the flight path segment has to be inspected with regard to flyability and feasibility. This is done by tracking the curve from the start point (i.e. $t = 0$) to the endpoint ($t = t_{\max}$) with a small time increment and checking the restrictions (5)–(8). If one of the restrictions is violated then the flight path segment is dismissed.

Due to the construction, each path in the network corresponds to a flyable and feasible trajectory. Each trajectory consists of a concatenation of flight path segments with line elements as connection points. This guarantees that the transition between adjacent flight path segments is smooth and without shock and jerk.

4. Route planning with landmark-based navigation

Let $\gamma(t)$ be a trajectory passing through a sequence of landmarks $L_1, L_2, \ldots, L_k$ (see Fig. 3). The trajectory will be called *navigable* if the flight distances between any two consecutive landmarks allow a navigation update in each landmark. In this section the route planning problem (RP) is extended to the route planning problem with landmark-based navigation.

(RP-N) Find a feasible, flyable and navigable trajectory of minimal length from a start point P with specified velocity vector $\vec{v}_P$ to a destination point Q with velocity vector $\vec{v}_Q$.

We explicitly emphasize that (RP-N) is a nontrivial task. The problem is highly complex from a combinatorial point of view. It includes

- selecting a set of suitable landmarks (from a pool of known landmarks),
- finding the optimal sequence of landmarks appearing on the trajectory,
- and choosing appropriate overflight directions.

A complete enumeration and evaluation of all subsets, sequences and directions is unrealistic. The number of possible configurations is extremely large and by far exceeds the available processing time.

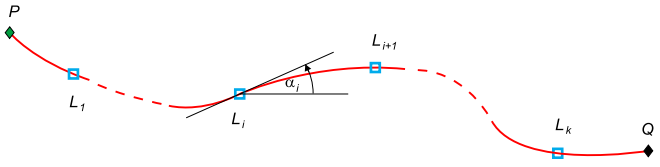

Fig. 3. Trajectory through a sequence of landmarks.

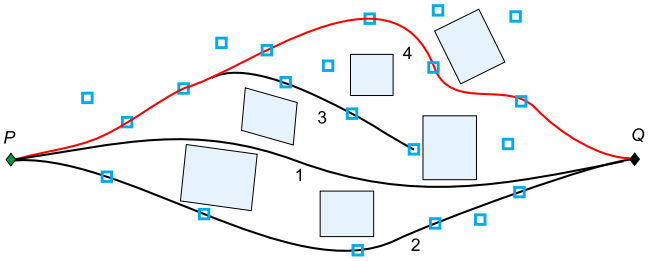

Fig. 4. Comparison of path-finding strategies.

On the other hand, naive approaches do not guarantee to find optimal solutions.

Consider the simple but illustrative scenario in Fig. 4 which allows to demonstrate different path finding strategies. Trajectory (1) is obtained by applying any conventional route planning algorithm. The route is not suitable for landmark-based navigation since there are no landmarks on the track or even close to it. The reason might be that the route leads across open water, across a desert or a region without any significant landmarks. Trajectory (2) is determined by an operator according to "best guess", i.e. a sequence of landmarks is manually selected and connected by shortest trajectories. This route is also not suitable since the flight distance between the second and third landmark is too large to allow a navigation update. Trajectory (3) is the result of a "Greedy" approach which proceeds stepwise from landmark to landmark. The method selects, from the set of all reachable landmarks, the one which is located closest to the destination. Unfortunately, this method gets trapped at a no-fly area. Trajectory (4) is the optimal solution.

Note that passing a landmark in the direction with highest quality value is not always the best strategy. The difficulty is illustrated in Fig. 5. Suppose that the left landmark has high quality values in direction North which are decreasing towards direction South. The trajectory emanating in direction North is quite long. Direction North–North–East is blocked by a no-fly area. The shortest trajectory comes from direction East–South–East. However, in this direction the quality value might be too low for a navigation update in the next landmark. Moreover, a shortest navigable trajectory between two landmarks is not necessarily part of a globally optimal solution.

We solve (RP-N) with a network-based approach. The first step is the construction of a *shortest-path-network*. Each landmark is represented by a set of vertices, one vertex for each overflight direction. The edges represent shortest trajectories between the landmarks, the edge costs are the lengths of the trajectories. The trajectories are obtained by solving a series of problems of type (RP). Here is a formal description of the algorithm.

`Algorithm Generate-Shortest-Path-Network`

Input: Landmarks $L_1, \ldots, L_m$

Output: Shortest-path-network $G = (V, E, c)$

$V := \emptyset$, $E := \emptyset$
For each landmark L_i
  For each overflight direction α
    CREATE a new vertex $v_{i,\alpha}$, $V := V \cup \{v_{i,\alpha}\}$
  end
end
For each pair of landmarks (L_i, L_j)
  For each overflight direction α of L_i
    For each overflight direction β of L_j
      FIND a feasible and flyable trajectory $\gamma(t)$ of minimum length from (L_i, α) to (L_j, β) by applying algorithm `Find-Trajectory`
      If $\gamma(t)$ exists
        CREATE a new edge $(v_{i,\alpha}, v_{j,\beta})$ with cost $c(v_{i,\alpha}, v_{j,\beta}) := \int |\dot{\gamma}(t)|\, dt$,
        $E := E \cup \{(v_{i,\alpha}, v_{j,\beta})\}$
      end
    end
  end
end

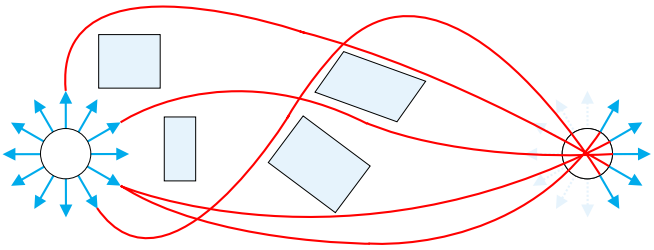

Fig. 5. Shortest trajectories between two landmarks.

Each path in this network corresponds to a flyable and feasible trajectory. The twice continuously differentiable curves $\gamma(t)$ are connected at the landmarks. The transition between two curves is smooth since the velocity is v_0 and the acceleration is zero in each endpoint (we assume nominal velocity v_0 over the landmarks).

Note that, in operational use, the air vehicle should pass the landmarks without strong maneuvers in order to keep the camera stable. This is realized by our approach since the acceleration over the landmarks is zero.

The second step is to include start and destination of the mission in the network. For this purpose, we generate trajectories from start point P to the landmarks and from the landmarks to destination point Q.

`Algorithm Connect-Endpoints`

Input: Start P, destination Q, shortest-path-network $G = (V, E, c)$

Output: Completed shortest-path-network $\tilde{G} = (\tilde{V}, \tilde{E}, c)$

CREATE vertices v_P for start P and v_Q for destination Q
$\tilde{V} := V \cup \{v_P, v_Q\}$, $\tilde{E} := E$
For each landmark L_i
  For each overflight direction α
    FIND a feasible and flyable trajectory $\gamma(t)$ of minimum length from P to (L_i, α) by applying algorithm `Find-Trajectory`
    If $\gamma(t)$ exists
      CREATE a new edge $(v_P, v_{i,\alpha})$ with cost $c(v_P, v_{i,\alpha}) := \int |\dot{\gamma}(t)|\, dt$,
      $\tilde{E} := \tilde{E} \cup \{(v_P, v_{i,\alpha})\}$
    end
    FIND a feasible and flyable trajectory $\gamma(t)$ of minimum length from (L_i, α) to Q by applying algorithm `Find-Trajectory`
    If $\gamma(t)$ exists
      CREATE a new edge $(v_{i,\alpha}, v_Q)$ with cost $c(v_{i,\alpha}, v_Q) := \int |\dot{\gamma}(t)|\, dt$,
      $\tilde{E} := \tilde{E} \cup \{(v_{i,\alpha}, v_Q)\}$
    end

17 **end**
18 **end**

In the third step all edges (trajectories) are deleted from the network which are not navigable. Let $\gamma(t)$ be a trajectory from P to Q passing the landmarks $L_1, \ldots, L_k$ at time $t_1, \ldots, t_k$ with overflight directions $\alpha_1, \ldots, \alpha_k$ (see Fig. 3). Then $\gamma(t)$ is navigable if

$$\int_{t_i}^{t_{i+1}} |\dot{\gamma}(t)|\, dt \leq dist(L_i, \alpha_i, \omega), \quad i = 1, \ldots, k$$

and

$$\int_{t_0}^{t_1} |\dot{\gamma}(t)|\, dt \leq dist(P)$$

holds, where $dist(P)$ denotes the maximal allowed flight distance from start point P to the first landmark. $dist(P)$ depends on the navigation (position) accuracy at the release. The reduced network consists of those trajectories which are not only feasible and flyable but also navigable. Now, the problem (RP-N) can be solved by computing a shortest path.

`Algorithm Find-Navigable-Trajectory`

Input: Shortest-path-network $\tilde{G} = (\tilde{V}, \tilde{E}, c)$, maximal flight distances $dist(L, \alpha, \omega)$

Output: Flyable, feasible and navigable trajectory $\gamma(t)$ from P to Q

1 **For each** edge $(v_P, v) \in \tilde{E}$
2 **If** $c(v_P, v) > dist(P)$
3 DELETE edge (v_P, v) from the network, $\tilde{E} := \tilde{E} \backslash \{(v_P, v)\}$
4 **end**
5 **end**
6 **For each** edge $(v_{i,\alpha}, v) \in \tilde{E}$
7 **If** $c(v_{i,\alpha}, v) > dist(L_i, \alpha, \omega)$
8 DELETE edge $(v_{i,\alpha}, v)$ from the network, $\tilde{E} := \tilde{E} \backslash \{(v_{i,\alpha}, v)\}$
9 **end**
10 **end**
11 FIND a shortest path in the network $\tilde{G} = (\tilde{V}, \tilde{E}, c)$ from v_P to v_Q
12 CONCATENATE the curves belonging to the path to a P–Q trajectory $\gamma(t)$

The algorithm finds a globally optimal solution provided that algorithm `Find-Trajectory` solves (RP) to optimality. This is ensured for a sufficiently fine discretization of the airspace, i.e. a sufficiently large network.

Note that a straightforward implementation of the algorithms is too slow for use in practice. In the next section we discuss implementation details which significantly reduce the running time.

5. Implementation

For the generation of the shortest-path-network, we have to discretize the 360° circle into a reasonable number k of overflight directions (line 3 of algorithm `Generate-Shortest-Path-Network`). Each landmark L_i is then represented by k vertices in the network. For each landmark pair (L_i, L_j) one has to find k^2 shortest trajectories. Since a landmark rosette is usually subdivided into 30° sectors with main directions 0°, 30°, 60°, ... (see Fig. 1) we accept these $k = 12$ directions.

Analyzing all pairs of landmarks (see line 7) would be extremely time-consuming. Fortunately, it is sufficient to examine those pairs which lie sufficiently close together. Let $d_{\max}(L_i)$ denote the maximal allowed flight distance from landmark L_i which allows a navigation update in the following landmark, i.e.

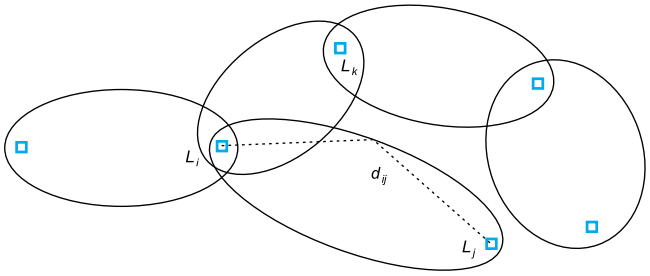

Fig. 6. Footprints of basic networks for flight path optimization.

$$d_{\max}(L_i) := \max_{\omega} \max_{\alpha} dist(L_i, \alpha, \omega).$$

$d_{\max}(L_i)$ assumes optimal environmental conditions ω and refers to the optimal overflight direction α in L_i. Obviously, a trajectory from L_i to another landmark L_j cannot be navigable if the Euclidean distance between L_i and L_j exceeds $d_{\max}(L_i)$.

Let L_j be a landmark with distance less than $d_{\max}(L_i)$. We have to solve a series of route planning problems (RP) from L_i to L_j (lines 10–11) which differ only in the release and approach directions. The positions of start and destination remain the same. The solution of these problems is based on a common network for algorithm `Find-Trajectory` (in the following referred to as *basic network*). As described in Section 3, the vertices of the network represent the (randomly distributed) line elements. The line elements are spread within an ellipse with focal points located at the landmark positions and major diameter equal to $d_{ij} = \max\{d_{\max}(L_i), d_{\max}(L_j)\}$. Obviously, each path from L_i to L_j of length smaller than d_{ij} lies completely within this ellipse. We call it the *footprint* of the network (see Fig. 6). The network can also be used to determine routes from L_j to L_i. Note that a flyable trajectory is also flyable in the reverse direction. On the other hand, this is not the case for navigability. A navigable trajectory is not always navigable in the reverse direction since navigability is depending on the quality of the originating landmark.

The shortest path calculations between L_i and L_j are performed using the algorithm of Dijkstra. This algorithm finds shortest paths from a single start vertex to a set of destination vertices (*one-to-many* problem). In order to solve our *many-to-many* problem we run Dijkstra's algorithm for every vertex belonging to L_i. We can omit vertices with release direction α where $\max_\omega dist(L_i, \alpha, \omega)$ is smaller than the Euclidean distance between L_i and L_j. In this case, no navigable trajectory exists.

The implementation of algorithm `Connect-Endpoints` uses the same ideas as described above.

For operational use of our approach, we propose a two-stage process consisting of a *preprocessing phase* and an *online planning phase*. This split is motivated by the idea to prepare whatever is possible well in advance. The remaining work should allow an extremely fast online planning. The preprocessing considers data which are known a priori including the kinematics and dynamics of the air vehicle, the position of the no-fly areas and the characteristic properties of the landmarks. This phase does not impose strong limitations on the computer running time. On the other hand, the online planning phase should allow quick reactions to changing scenarios. It considers mission data including the position of the start and destination points with release and approach conditions and the rapidly changing environmental conditions.

Preprocessing Phase

1. `Generate-Shortest-Path-Network`

Online Planning Phase

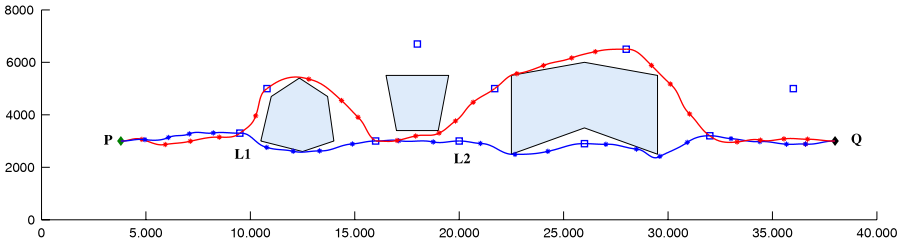

Fig. 7. Trajectories in Scenario (I) with high-quality landmarks (lower curve) and two degraded landmarks (upper curve).

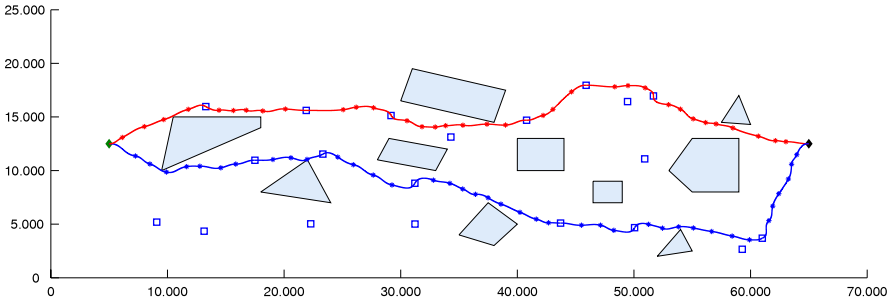

Fig. 8. Trajectories in Scenario (II) with type *N* landmarks (lower curve) and type *S* landmarks (upper curve).

2. `Connect-Endpoints`
3. `Find-Navigable-Trajectory`

The result of the preprocessing phase can be considered as part of a database for the mission planning system. The online planning phase is performed immediately before launch of the mission.

6. Computational results

The algorithms were implemented and visualized with Matlab R2012a. The calculations were performed on a standard laptop computer with 2.2 GHz Intel Core i7 CPU and 8 GB of RAM.

Scenario (I) is defined within an operational area of size 40 km × 8 km. It includes 3 no-fly areas and 10 navigation landmarks (see Fig. 7). The start point P of the mission is located on the left with release direction 0°. The destination point Q is on the right, the direction of the approach is also 0°. For the air vehicle we assume a nominal velocity of 30 m/s. The minimal and maximal velocity is 25 m/s and 35 m/s, respectively. The lateral acceleration is restricted by 20 m/s^2.

In the first test case, it is assumed that all landmarks are of the same quality. The maximal allowed flight distance to the next landmark is 7 km, independent from the overflight direction. The optimization provides the lower (blue) trajectory. In the second test case, the quality of landmarks L_1 and L_2 is degraded to a maximal flight distance of 3.5 km. This enforces a significant detour around the no-fly areas. Note that the asterisks on the trajectories indicate the connection points of the flight path segments (obtained by algorithm `Find-Trajectory`).

Scenario (II) contains 10 no-fly areas and 20 navigation landmarks in an area of 70 km × 25 km (see Fig. 8). The nominal velocity of the air vehicle is 100 m/s, minimal and maximal velocity is 80 m/s and 125 m/s, respectively. The maximal lateral acceleration is 40 m/s^2. For this scenario we implemented two different landmark types. Type N (resp. type S) represents a landmark with optimal quality in overflight direction North (South) and decreasing quality towards direction South (North).

The maximal flight distance $d_{\max}$ between two consecutive landmarks is 14 km. This value holds only if the air vehicle passes a landmark in a direction with optimal quality value. With decreasing quality value q the allowed flight distance d also gets smaller. For the following test cases we choose

$$d = d_{\max} \cdot q$$

as a distance function. For instance, a type N landmark allows a maximal flight distance of 14 km to the next landmark if it is passed in direction North (quality value $q = 1.0$). The distance is decreasing to 7 km for overflight directions East and West ($q = 0.5$). Passing a type N landmark in direction South ($q = 0.0$) does not make any sense since the associated distance is zero.

In the first test case all landmarks are of type N. If the distance from one landmark to the next landmark is large (close to $d_{\max}$) then the current landmark has to be passed in direction North or close to North. This is clearly reflected by the obtained trajectory (see the lower curve in Fig. 8). For example, the third landmark is not passed using the shortest connection from West to East but in a direction close to North since the fourth landmark is far away. In the second test case, we provided the landmarks with quality values of type S. In this case the situation is contrary. Large flight distances require overflights in direction South (see the upper curve of Fig. 8).

The shortest-path-network for Scenario (II) consists of 240 vertices and almost 6000 edges. Algorithm `Generate-Shortest-Path-Network` investigates 81 landmark pairs. Each of the associated basic networks (see Fig. 6) consists of 500 line elements which result in up to 70 000 flyable and feasible flight path segments. In both test cases, algorithm `Find-Navigable-Trajectory` removes about 60% of the edges from the shortest-path-network so that little more than 2000 edges remain. While the preprocessing requires about 30 min, the computer running time for the online planning is less than 2 s. It is worth mentioning that the quality of the solutions can be improved by increasing the number of line elements in the basic networks. The enhanced accuracy is achieved at the expense of growing preprocessing time.

Scenario (III) covers an operational area of size 100 km × 35 km (see Fig. 9). It contains 20 no-fly zones and 50 landmarks. The

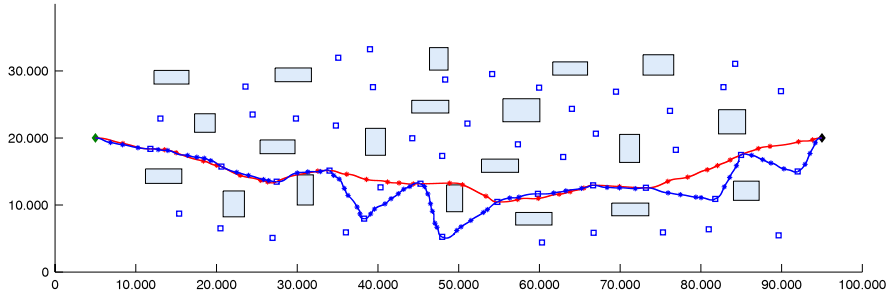

Fig. 9. Trajectories in Scenario (III) with high-quality landmarks (upper curve) and degraded landmarks (lower curve).

velocity of the UAV is restricted by 40 m/s and 60 m/s, respectively, with a nominal velocity of 50 m/s. The lateral acceleration is restricted by 15 m/s^2. In the first test case the maximal flight distance between two landmarks is 12.5 km (see the upper curve). In the second test case, the quality of all landmarks is degraded to 70% which corresponds to a maximal distance of 8.75 km. This results in a zigzag course (lower curve). The preprocessing time is about 40 min. Again, the online planning is extremely fast and requires less than 3 s.

The computational effort of our algorithm depends, in particular, on the number of landmarks in the scenario and the distance between them. The effort grows with the number of landmarks. On the other hand, the effort is reduced with increasing distance between the landmarks. The reason is that, for each landmark pair (i.e. pairs which lie sufficiently close together, see Section 5), the algorithm computes shortest trajectories for all combinations of overflight directions. These trajectories are then inserted as edges in the shortest-path-network. If the distances between the landmarks are large then the number of landmark pairs is small.

To quantify how the effort increases with the density of the landmarks, consider again Scenario (II). We added 10 landmarks by distributing them randomly within the operational area. The shortest-path-network has now 360 vertices and about 9000 edges, 129 landmark pairs are investigated. The preprocessing time increases to 45 min, the online planning requires 2 s. Doubling the number of landmarks to 40 provides a network with 480 vertices and more than 13 000 edges, with 186 landmark pairs to investigate. The preprocessing requires more than 1 h, the online planning less than 3 s.

7. Conclusion

In this paper we presented a network-based method to solve the flight path planning problem for fixed-wing UAVs in scenarios with obstacles. The main contribution is that not only the flight performance, but also the navigational capabilities of the air vehicle are taken into account. We considered UAVs equipped with landmark-based visual navigation, a system which is less vulnerable to hostile acts than GPS, since it is not guided by external signals.

The computational complexity of the flight path problem usually requires some trade-off between accuracy and computation time. However, practical applications often ask for high-quality solutions while at the same time stipulating modest processing times. Our algorithm performs all time-consuming tasks in a preprocessing phase well in advance. This allows an extremely quick online planning.

It is a common assumption in many papers that fixed-wing UAVs fly with constant altitude. This seems reasonable in certain applications such as searching an unknown terrain where the sensor coverage region and sensor properties should be the same everywhere. Another reason might be that an air vehicle achieves its maximal range if the flight altitude is kept constant. Climbing and diving is extremely fuel consuming, so that these maneuvers are avoided as far as possible.

On the other hand, our approach can easily be extended to the 3D case. We let the UAV pass the landmarks in an altitude which guarantees full functionality of the onboard camera and best possible navigation updates. The line elements representing the landmarks get an additional height information and the flight paths between the landmarks are generated in the three-dimensional airspace. For instance, one might be interested in finding a shortest trajectory over hilly terrain or, if the air vehicle is exposed to unknown threats, realize a low level flight to benefit from terrain masking effects (see [11]).

There are unresolved issues, for example concerning the robustness of the approach. An unforeseen loss of visibility of a landmark while following a pre-computed path could require modifying the path to a nearby landmark which allows a navigation update. This implies a re-planning of the trajectory during flight. Another challenge is to consider changing no-fly areas, e.g. due to popup threats or unforeseen obstructions detected during flight. These are interesting areas of future research.

References

[1] Z. Sarris, Survey of UAV applications in civil markets, STN ATLAS-3Sigma AE and Technical University of Crete, DPEM, 73100 Chania, Crete, Greece, 2001.
[2] B.R. Donald, P.G. Xavier, J. Canny, J. Reif, Kinodynamic planning, J. ACM 40 (1993) 1048–1066.
[3] S.M. LaValle, Planning Algorithms, Cambridge University Press, 2006.
[4] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Boston, 1991.
[5] W.M. Carlyle, J.O. Royset, R.K. Wood, Routing military aircraft with a constrained shortest-path algorithm, Mil. Oper. Res. 14 (3) (2009) 31–52.
[6] E. Rippel, A. Bar-Gill, N. Shimkin, Fast graph-search algorithms for general-aviation flight trajectory generation, J. Guid. Control Dyn. 28 (4) (2005) 801–811.
[7] M. Zabarankin, S. Uryasev, R. Murphey, Aircraft routing under the risk of detection, Nav. Res. Logist. 53 (8) (2006) 728–747.
[8] S.A. Bortoff, Path planning for UAVs, in: Proc. Am. Control Conf., Chicago, Illinois, 2000, pp. 364–368.
[9] K.B. Judd, T.W. McLain, Spline based path planning for unmanned air vehicles, in: AIAA Guid. Navig. Control Conf. Exhib., Montreal, Canada, 2000.
[10] C. Goerzen, Z. Kong, B. Mettler, A survey of motion planning algorithms from the perspective of autonomous UAV guidance, J. Intell. Robot. Syst. 57 (2010) 65–100.
[11] L. Babel, Three-dimensional route planning for unmanned aerial vehicles in a risk environment, J. Intell. Robot. Syst. 71 (2) (2013) 255–269.
[12] K.R. Britting, Inertial Navigation Systems Analysis, John Wiley & Sons, New York, 1971.
[13] D.H. Titterton, J.L. Weston, Strapdown Inertial Navigation Technology, Institution of Electrical Engineers, Stevenage, UK, 2004.
[14] J. Farrell, M. Barth, The Global Positioning System and Inertial Navigation, McGraw-Hill, New York, 1998.
[15] M.S. Grewal, L.R. Weill, A.P. Andrews, Global Positioning Systems, Inertial Navigation and Integration, John Wiley & Sons, New York, 2007.

[16] D. Wright, L. Grego, G. Gronlund, The Physics of Space Security, Cambridge MA, 2005.
[17] F. Bonin-Font, A. Ortiz, G. Oliver, Visual navigation for mobile robots: a survey, J. Intell. Robot. Syst. 53 (3) (2008) 263–296.
[18] G.N. DeSouza, A.C. Kak, Vision for mobile robot navigation: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2) (2002) 237–267.
[19] A. Cesetti, E. Frontoni, A. Mancini, A. Ascani, P. Zingaretti, S. Longhi, A visual global positioning system for unmanned aerial vehicles used in photogrammetric applications, J. Intell. Robot. Syst. 61 (2011) 157–168.
[20] G. Conte, P. Doherty, Vision-based unmanned aerial vehicle navigation using geo-referenced information, J. Adv. Signal Process. (2009) 18. Article ID 387308.
[21] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, third ed., MIT Press, Cambridge, 2009.

Luitpold Babel studied mathematics and received his diploma in 1987 and his doctorate degree in 1990, both from the Technical University Munich, Germany. Following research positions at the Old Dominion University, Norfolk, USA and University Graz, Austria, he achieved his habilitation in 1997. The main focus of his work was on problems of combinatorial optimization and algorithmic graph theory. From 1999–2006 he worked in the missile industry where he joined research and development teams of different subsidiaries of the companies EADS and MBDA. Since 2007 he is a full professor of mathematics and computer science at the University of the German Federal Armed Forces (Universität der Bundeswehr) in Munich, Germany. His research interests include mathematical methods in defense analyses, route planning of autonomous systems, and mission planning for guided missiles and unmanned air vehicles.