JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 33, No. 2, March–April 2010

Smooth Flight Trajectory Planning in the Presence of No-Fly Zones and Obstacles

Massimiliano Mattei* and Luciano Blasi†
Second University of Naples, Aversa 81031, Italy

DOI: 10.2514/1.45161

This paper describes a novel procedure to generate continuously differentiable optimal flight trajectories in the presence of arbitrarily shaped no-fly zones and obstacles having a fixed position in time. The operational flight scenario is first discretized with a finite dimensional grid of positions-directions pairs. A weighted and oriented graph is then defined for which the nodes are the earlier mentioned grid points and for which the arcs correspond to minimum length trajectories compliant with obstacle avoidance constraints. Arcs are obtained via solving convex quadratic programming optimization problems that can also account for geometrical constraints such as trajectory curvature limitations. The problem of finding an optimal trajectory be tween two nodes of the so-called core paths graph is then solved via a minimum cost path search algorithm. In a real-time application perspective, the generation of the core paths graph is computationally cumbersome. Moreover, the aircraft position and direction rarely coincide with one of the graph nodes. However, if the graph is built offline and stored, the definition of an optimal trajectory connecting any points of the space domain requires a reduced computational effort. The particular case of piecewise polynomial trajectories minimizing a flight path's length, compliant with constraints on curvature and flight-path angles, is fully developed. Two- and three-dimensional examples are discussed to show the applicability as well as the effectiveness of the technique.

Nomenclature

$C(\theta_{E_0,E_f})$	=	cost related to the admissible flight trajectory θ_{E_0,E_f}
$d_2(\cdot,\cdot)$	=	Euclidean distance function on $\mathbb{R}^n$
$d'(\cdot,\cdot)$	=	metric on $\mathbb{R}^n$ for the position vectors
$d''(\cdot,\cdot)$	=	metric on $\mathbb{R}^n$ for the velocity vectors
$E(t) = [s(t), \dot{s}(t)]$	=	pair of position and tangent on the s curve
$E_f = (s_f, \dot{s}_f)$	=	pair of position and tangent at the trajectory ending point
$E_i = (s_i, \dot{s}_i)$	=	pair of position s_i and direction $\dot{s}_i$ vectors
$E_0 = (s_0, \dot{s}_0)$	=	pair of position and tangent at the trajectory starting point
$f_\delta(t)$	=	value of the parameterized trajectory function in t
$K_{E_1}^{\Delta,\Sigma}$	=	Σ-flight controllability set to E_1
N_{CPG}	=	set of CPG nodes
$s(t)$	=	value of the trajectory function in t
$s(\cdot)$	=	trajectory function
$T^{\Delta,\Sigma}$	=	optimal Δ-connection set
t	=	independent parameter for the $s(t)$ trajectory curve definition
$(x(t), y(t), z(t))$	=	position vector on the trajectory curve
$(x(t_f), y(t_f), z(t_f))$	=	trajectory ending point position vector
$(x(t_0), y(t_0), z(t_0))$	=	trajectory starting point position vector
Δ	=	operational space
δ	=	parameter vector used for the finite dimensional parameterization of the trajectory segments
$\partial\Delta$	=	boundary of the operational space
$\Theta_{E_0,E_f}^{\Delta}$	=	set of admissible flight trajectories from E_0 to E_f in Δ
$\Theta_{E_0,E_f}^{\Delta,\Sigma}$	=	set of Σ-admissible flight trajectories from E_0 to E_f in Δ
θ_{E_0,E_f}	=	admissible flight trajectory between E_0 and E_f
$\Xi_{E_1}^{\Delta,\Sigma}$	=	Σ-flight reachability set from E_1
Σ	=	set of flight trajectory properties
Υ_{CPG}	=	set of CPG arcs
Ω	=	set of δ parameters

Received 28 April 2009; revision received; 1 September 2009; accepted for publication 5 October 2009.

*Full Professor, Department of Aerospace and Mechanical Engineering, Via Roma 29. Senior Member AIAA.

†Assistant Professor, Department of Aerospace and Mechanical Engineering, Via Roma 29.

I. Introduction

CIVIL and military flight missions require a careful resources optimization to maximize efficiency and effectiveness, as well as to reduce risks and operating costs. To this end, the definition of an optimal flight trajectory consistent with mission objectives, operational scenario, as well as vehicles dynamics and performance, certainly plays an important role. Objectives are often specified in terms of regions to fly over, desired flight altitudes on targets, and definition of possible loads to be released on targets. The operational scenario provides different constraints depending on takeoff and landing zones, the presence of no-fly zones or high risk zones, the presence of mountains or adverse flight conditions, minimum/maximum distance from base stations or possible cooperating vehicles. Finally, constraints related to the specific aircraft used can include maximum climb angles/rates, maximum and minimum speed, minimum turning radius, flight range, and endurance.

During the last decades much work has been carried out on the trajectory optimization problem for flying, marine, and terrestrial vehicles. The variational formulation is probably the most natural and rigorous one for this class of problems. However, the possibility to solve complex problems with variational methods is very poor. Many papers deal with direct and indirect numerical methods based on the solution of a nonlinear programming (NLP) problem [1–9]. Indirect methods try to satisfy optimality necessary conditions deriving from the application of the Pontryagin's maximum principle. Direct methods are based on the discretization of state and input variables sets to convert the functional problem into a NLP problem. Unfortunately NLP turns out to be quite burdensome for many practical applications.

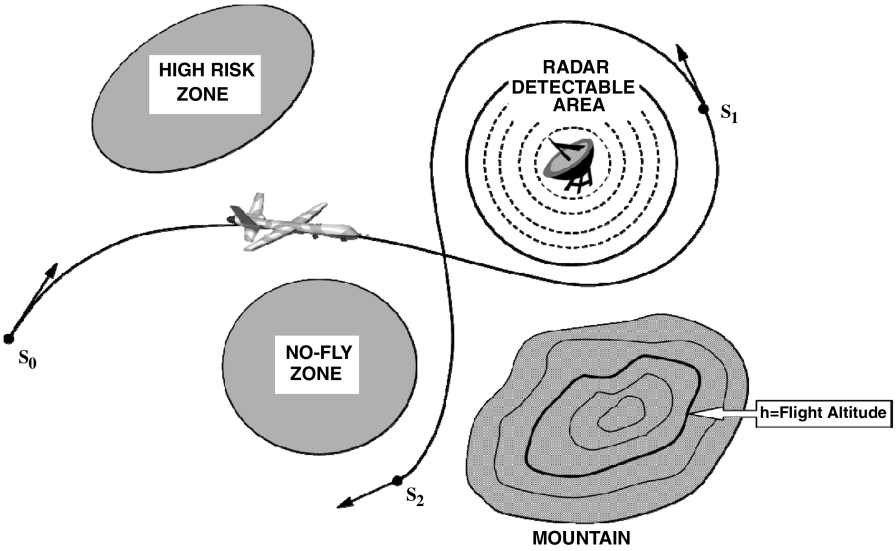

Fig. 1 Flight trajectory in the presence of mission constraints.

By means of some approximation, feasible trajectories can be generated following a purely geometrical approach, based on topological techniques creating a sequence of way points. This sequence can derive from probabilistic or potential methods [10–12] which are very popular in mobile robotics applications [13]. In the same field, several papers deal with inverse kinematics based approaches for holonomic and nonholonomic systems accounting for obstacle avoidance and energy minimization [14]. Geometrical approaches have been proposed for the generation of flight trajectories using circular arcs combined with straight connection paths [15,16]. A method for generating up to 4-D trajectories has been recently proposed in [17].

More sophisticated approaches take into account flight dynamics [18–22]: some of them are based on the so-called motion primitives, i.e., the trajectory is modeled as a sequence of trim conditions and maneuvers; others are based on a trajectory smoothing. With higher computational effort, it is possible to apply model-based predictive techniques also dealing with constraint on states/inputs [23–25]. Under simplifying assumptions, the predictive problem can be formulated in terms of mixed integer linear programming (MILP) problems that can be efficiently solved with branch and bound methods. MILP allows the inclusion of logical expressions into the optimization process to model decision making.

Because of the complexity and variety of the problem, the use of nonconventional nature-inspired optimization techniques has also been investigated. The literature reports examples of hybrid soft computing and genetic techniques applied to the definition of space and atmospheric vehicle trajectories [26–29].

This paper deals with the generation of optimal flight trajectories compliant with mission constraints resulting from no-fly zones or obstacles (see Fig. 1). A no-fly zone is an area which aircraft are not permitted to fly over, due to the presence of military restrictions (e.g., armed enemies, radar detectability) or civil restrictions mainly due to safety reasons (e.g., densely populated areas, adverse weather conditions zones). Mountains and buildings are examples of obstacles within natural and urban environments, respectively.

Constraints on flight paths, turn radius, climb, and descent angles are also taken into account in the proposed algorithm. The trajectory optimization problem is converted into a minimum cost path search within a weighted and oriented graph for which the arcs and weights are obtained via solving convex quadratic programming optimization problems. A core paths graph (CPG), containing a discrete set of admissible connection paths, is first generated. Once the graph is available, the proposed procedure allows defining an optimal trajectory connecting any points of the space domain other than CPG nodes with a reduced computational effort.

The particular case of piecewise polynomial trajectories minimizing a flight path's length is fully developed. Two- and three-dimensional examples are discussed to show the applicability as well as the effectiveness of the proposed technique. The possibility of having a feedback real-time implementation of the trajectory generation algorithm is also discussed.

The paper is organized as follows. In Sec. II the flight trajectory optimization problem within a constrained environment is precisely stated. Sec. III describes the proposed CPG algorithm. The case of piecewise polynomial flight trajectories is developed in Sec. IV. Numerical examples in 2-D and 3-D spaces are discussed in Sec. IV. Some conclusions are finally drawn.

II. Mathematical Problem Formulation

Let us consider a space domain $\Delta \subseteq \mathbb{R}^n$, possibly nonconnected, in 2-D or 3-D ($n = 2$ or 3) for which the boundary $\partial\Delta$ is determined by the presence of no-fly zones and obstacles (see Fig. 2). A parametric curve $s(\cdot)$: $t \in [t_0, t_f] \rightarrow s(t) = (x(t), y(t), z(t))^T \in \Delta$ is defining a trajectory segment in Δ, where t is the independent parameter, $(x, y, z)^T$ is the vector of space coordinates in a given Cartesian reference system, and $(x(t_0), y(t_0), z(t_0))^T$ and $(x(t_f), y(t_f), z(t_f))^T$ are the starting and ending points of the segment, respectively. With $E(\tilde{t}) = (s(\tilde{t}), \dot{s}(\tilde{t}))$ we denote the pair of position and tangent ($\dot{s} = \mathrm{d}s/\mathrm{d}t$) to the trajectory at $t = \tilde{t}$. With a slight abuse of notation $E_k = (s_k, \dot{s}_k)$, $k = 1, \ldots, N$, denotes an element of a discrete set of position-direction pairs in the operational domain. $(s_0, \dot{s}_0)$ and $(s_f, \dot{s}_f)$ denotes the point-tangent pairs of a flight trajectory segment at the starting and ending points, respectively.

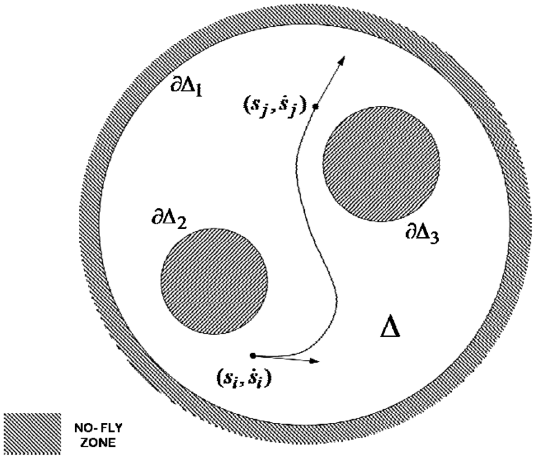

Fig. 2 Example of 2-D Δ-space, $\partial\Delta = \partial\Delta_1 \cup \partial\Delta_2 \cup \partial\Delta_3$-boundary, and Δ-admissible trajectory connecting discrete points $E_i = (s_i, \dot{s}_i)$ to $E_j = (s_j, \dot{s}_j)$ $[s_i = s(t_0), \dot{s}_i = \dot{s}(t_0), s_j = s(t_f), \dot{s}_j = \dot{s}(t_f)]$.

Definition 1 (flight trajectory): A flight trajectory from $E_0 = (s_0, \dot{s}_0)$ to $E_f = (s_f, \dot{s}_f)$ in Δ is a continuously differentiable parametric oriented curve θ_{E_0,E_f} in the independent variable t, connecting s_0 to s_f with assigned first derivatives (tangents) $\dot{s}_0$ and $\dot{s}_f$ at the extremes:

$$\theta_{E_0,E_f} \triangleq \Big\{ s(\cdot) \in C^1_{[t_0,t_f]} \colon s(t_0) = s_0, s(t_f) = s_f, \dot{s}(t_0) = \dot{s}_0, \dot{s}(t_f) = \dot{s}_f \Big\} \tag{1}$$

Definition 2 (Δ-compatibility): A pair of position-direction vectors $E_i = (s_i, \dot{s}_i)$ is Δ-compatible if there exists an $\bar{\alpha} > 0$ such that $s_i + \alpha \dot{s}_i \in \Delta, \forall\ \alpha \in [0, \bar{\alpha})$.

Definition 3 (flight trajectory admissibility in Δ): A flight trajectory connecting Δ-compatible E_0 and E_f is admissible in Δ if it is fully contained in $\Delta\ \forall\ t \in [t_0, t_f]$. The set of admissible flight trajectories from E_0 to E_f in Δ is denoted as

$$\Theta^{\Delta}_{E_0,E_f} \triangleq \Big\{ \theta_{E_0,E_f} \colon s(t) \in \Delta \quad \forall\ t \in [t_0, t_f] \Big\} \tag{2}$$

It is obvious by definition that admissible flight trajectories are composed of $E(t)$, which are Δ-compatible $\forall\ t \in [t_0, t_f]$.

A more restrictive definition of admissibility is needed for next developments.

Definition 4 (flight trajectory Σ-admissibility in Δ): Given a set Σ of desired flight trajectory properties and two Δ-compatible E_0 and E_f, a flight trajectory is Σ-admissible in Δ if it is admissible and satisfies Σ properties. The set of Σ-admissible flight trajectories from E_0 to E_f in Δ is denoted as

$$\Theta^{\Delta,\Sigma}_{E_0,E_f} \triangleq \Big\{ \theta_{E_0,E_f} \in \Theta^{\Delta}_{E_0,E_f} \colon s_{[t_0,t_f]}(\cdot) \text{ satisfies } \Sigma \Big\} \tag{3}$$

Examples of flight trajectory properties that will be used in the numerical examples are:

1) Σ_1: bounds on the second derivatives; $|\ddot{s}(t)| \le \bar{\ddot{s}}\ \forall\ t \in [t_0, t_f]$. This property is used to limit the trajectories curvature radius.

2) Σ_2: bounds on the distance between s_0 and s_f; $d'(s_0, s_f) \le \bar{d}$, $d'(\cdot,\cdot)$ being a metric on $\mathbb{R}^n$.

3) Σ_3: bounds on the difference between $\dot{s}_0$ and $\dot{s}_f$; $d''(\dot{s}_0, \dot{s}_f) \le \bar{\dot{d}}$, $d''(\cdot,\cdot)$ being a metric on $\mathbb{R}^n$.

4) Σ_4: finite dimensional parameterization of the trajectory segments curves; $s(t) = f_\delta(t)$ depends on a parameter vector $\delta \in \Omega \subseteq \mathbb{R}^p$. Polynomials are examples of possible families of connecting curves.

Compliance with Σ properties may compromise the possibility of reaching one point of the space from any another.

Definition 5 (Σ-flight controllability and reachability): Given two Δ-compatible E_1 and E_2, E_1 is Σ-flight controllable to E_2 in Δ (and conversely E_2 is Σ-flight reachable from E_1) if $\Theta^{\Delta,\Sigma}_{E_1,E_2}$ is a nonempty set. The set of all the E_i Σ-flight controllable (Σ-flight reachable) to (from) E_1 is called Σ-flight controllability (Σ-flight reachability) set to (from) E_1 and is denoted by $K^{\Delta,\Sigma}_{E_1}$ ($\Xi^{\Delta,\Sigma}_{E_1}$).

Finally, a cost related to admissible flight trajectories θ_{E_0,E_f} is defined. This can be computed on the basis of a nonnegative function:

$$C(\cdot) \colon \theta_{E_0,E_f} \in \Theta^{\Delta,\Sigma}_{E_0,E_f} \to C(\theta_{E_0,E_f}) \in \mathbb{R}^+_0 \tag{4}$$

Cost functions can depend on the path length, flight altitude, fuel consumption, risk for the aircraft or for the people under the flying area, radar detectability, distance from fixed or mobile base stations, or any other variable concerning the specific flight mission. They are generally integral functions with possible terminal costs having the following structure

$$C(\theta_{E_0,E_f}) = \int_{t_0}^{t_f} G(s, \dot{s}, \ddot{s}, t)\mathrm{d}t + \beta_f(s(t_f), \dot{s}(t_f), \ddot{s}(t_f), t_f) \tag{5}$$

We can now precisely state the flight trajectory optimization problem for which the solution is the main objective of the paper.

Problem 1: (search of the optimal flight trajectory connecting E_0 to E_f): Given a space domain Δ, a set Σ of desired properties, two Δ-compatible E_0 and E_f, with E_0 Σ-flight controllable to E_f, and a cost function $C(\cdot)$, find the Σ-admissible optimal (minimum cost) flight trajectory $\theta^*_{E_0,E_f} \in \Theta^{\Delta,\Sigma}_{E_0,E_f}$ solving the following minimization problem:

$$\min_{\theta_{E_0,E_f} \in \Theta^{\Delta,\Sigma}_{E_0,E_f}} C(\theta_{E_0,E_f}) \tag{6}$$

The optimal flight trajectory connecting E_0 to E_f is then defined as

$$\theta^*_{E_0,E_f} = \arg \min_{\theta_{E_0,E_f} \in \Theta^{\Delta,\Sigma}_{E_0,E_f}} C(\theta_{E_0,E_f}) \tag{7}$$

The optimization problem (6) is generally complex to be solved; $\Theta^{\Delta,\Sigma}_{E_0,E_f}$ is an infinite dimensional space, the cost function can be nonlinear and nonconvex, and also mathematical constraints implied by $\theta_{E_0,E_f} \in \Theta^{\Delta,\Sigma}_{E_0,E_f}$ are generally nonlinear and nonconvex.

The following definition helps identify the entire flight connectivity of Δ space.

Definition 6 (optimal Δ-connections set): The set of all the Σ-admissible optimal flight trajectories connecting each Δ-compatible E to each other is called the optimal Δ-connections set:

$$T^{\Delta,\Sigma} \triangleq \{\theta^*_{E_i,E_j} = \arg \min_{\theta_{E_i,E_j} \in \Theta^{\Delta,\Sigma}_{E_i,E_j}} C(\theta_{E_i,E_j}) \colon \quad \forall\ E_i \quad \text{and} \quad E_j\ \Delta - \text{compatible}\} \tag{8}$$

It is worth notice that, given a target point E_f, the knowledge of $T^{\Delta,\Sigma}$ would in principle allow the optimal flight trajectory to be expressed E_f as a feedback control law because, at each time instant, the optimal trajectory to E_f can be extracted from $T^{\Delta,\Sigma}$, assuming that $E_j = E_f$ and E_i is equal to the current measured $E = (s, \dot{s})$.

III. Core Paths Graph Algorithm

A number of assumptions and approximations are now introduced to convert the minimum search problem (6), arising in problem 1, into a minimum cost path search over a graph.

Definition 7 (Δ-core paths graph): A Δ-CPG is a discrete approximation of the optimal Δ-connections set. Nodes are obtained choosing a suitable N-ple of Δ-compatible E_i. Arcs are Σ-admissible optimal flight trajectories connecting nodes. N_{CPG} and Υ_{CPG} denote the sets of CPG nodes and arcs, respectively, (see Fig. 3).

It is worth notice that CPG generation is strongly affected by the following choices:

1) Number and position of Δ-compatible $E_i = (s_i, \dot{s}_i)$ composing N_{CPG}.

2) Σ properties including a possible parameterization of the Σ-admissible curves (Σ_4 property).

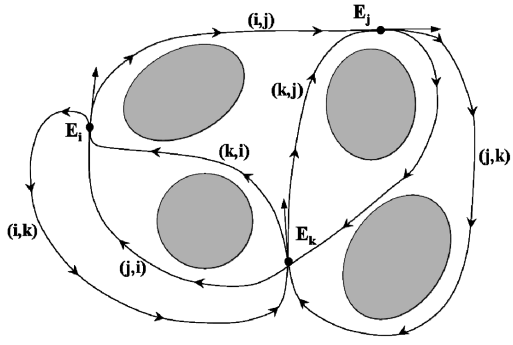

Fig. 3 Trajectories composing a CPG with 3 nodes (E_i, E_j, E_k) and 6 arcs [(i,j), (i,k), (k,i),, (k,j), (j,i), (j,k) flight trajectories].

3) Cost function for the definition of optimal trajectories.

Depending on Σ properties, each node is not necessarily Σ-flight reachable from each other. This implies that CPG can be non-complete (i.e., a simple graph in which not every pair of distinct nodes is connected by an arc). Noncompleteness can be desirable in practice because it causes a reduction of the computational burden related to calculations on graphs.

If Σ_4 property is required, trajectory curves are forced to depend on a finite dimensional parameter vector δ:

$$\theta^f_{E_0,E_f}(\delta) \triangleq \{s(t) = f_\delta(t)\colon t \in [t_0, t_f], \delta \in \Omega \subseteq \mathbb{R}^p, E(t_0) = E_0, E(t_f) = E_f\} \tag{9}$$

where Ω is the set of parameters ensuring Σ-admissible flight trajectories:

$$\Omega \triangleq \left\{\delta\colon \theta^f_{E_0,E_f}(\delta) \in \Theta^{\Delta,\Sigma}_{E_0,E_f}\right\} \tag{10}$$

With this parameterization the infinite dimensional problem (6) is converted into a finite dimensional one, and the calculation of CPG weights $W_{i,j}$ is obtained via solving the following problem

$$W_{i,j} = \min_{\delta\in\Omega} C(\theta^f_{E_i,E_j}(\delta)) \quad \forall\ E_i \in N_{\mathrm{CPG}} \quad \forall\ E_j \Sigma\text{-flight reachable from } E_i \tag{11}$$

Convexity properties and nonlinearities of problem (11) depend on choices (1–3). Examples are discussed in the next section in the light of the proposed practical problems.

Once a CPG has been built, the optimal flight trajectory between two nodes of the graph can be readily calculated by using one of the minimum cost path search algorithms available in the literature: Dijkstra's algorithm is the most popular algorithm solving the shortest path problem in the presence of nonnegative arc path costs [30].

To compute optimal flight trajectories between points $E_0 = (s_0, \dot{s}_0)$ and $E_f = (s_f, \dot{s}_f)$ which do not belong to the CPG, an enlargement of the CPG to include these points in N_{CPG} is first required. New arcs connecting E_0 to its Σ-flight controllability set on N_{CPG} and new arcs connecting E_f to its Σ-flight reachability set on N_{CPG} must then be added to Υ_{CPG}.

The following algorithm to compute flight trajectories connecting two points of the operational space with assigned flight directions is composed of an offline and an online part. In the offline part the flight connectivity on the CPG is determined. The online part assumes that a measurement of the position and velocity vector of the aircraft is available and that a target position and direction is assigned; it computes optimal connections of these two pairs of velocity and direction with the CPG; it finally solves the minimum cost path search problem on the enlarged CPG.

The possibility of implementing the proposed procedure in real time depends on how fast the online part of the algorithm can be performed compared with the control system sampling period. It is possible to evaluate an upper bound on computational times for both the CPG enlargement and the shortest path calculation on graph. This guarantees a finite-time response of the procedure.

A. Core-Paths-Graph Trajectory Optimization Algorithm: Offline Calculations and CPG Computation

1) Choose N suitable $E_i = (s_i, \dot{s}_i)$, i.e., the N_{CPG} set of nodes. A simple criterion can be a uniform distribution of points and directions with increased density in the neighborhood of obstacles and no-fly zones and in the regions in which the largest changes in flight direction are expected. An incremental procedure to evaluate the benefit due to the addition of more nodes can be readily implemented.

2) Choose a cost function related to path length, flight altitude, fuel consumption, risk, radar detectability, distance from fixed or mobile base stations. Build the Υ_{CPG} set of arcs; connect each node E_i to each E_j which is Σ-flight reachable from E_i solving problem (11). Store graph weights into a sparse matrix.

B. Core-Paths-Graph Trajectory Optimization Algorithm: Online Calculations

1) For inclusion of a (possibly time varying) terminal point E_f (if $E_f \notin N_{\mathrm{CPG}}$), enlarge CPG including node E_f and arcs connecting each Σ-flight controllable node $E_j \in N_{\mathrm{CPG}}$ to E_f.

2) For inclusion of a (possibly measured) starting point E (if $E \notin N_{\mathrm{CPG}}$), enlarge CPG including node E (pair of measured position and velocity vectors) and arcs connecting every Σ-flight reachable node $E_j \in N_{\mathrm{CPG}}$ from E.

3) Compute the optimal trajectory from E to E_f solving a minimum cost path problem over the enlarged CPG.

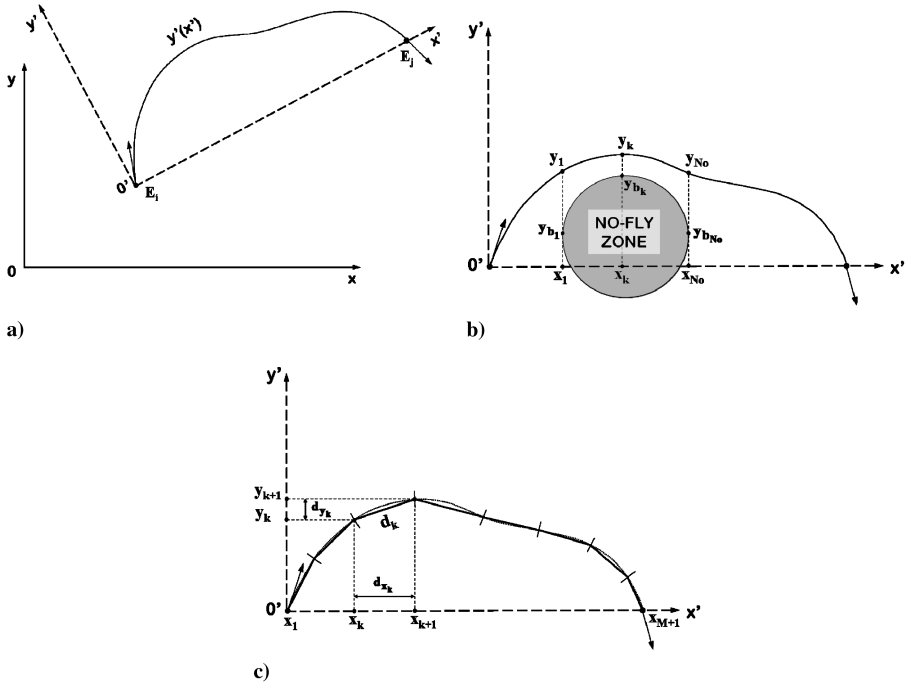

Fig. 4 a) Local reference system ($O'x'y'$) definition. b) Variables definition for obstacle avoidance. c) Variables definition for length calculation.

4) Evaluate and follow the optimal trajectory for the given feedback control sampling period.

5) Measure a new E and restart from step 2 till E_f is reached or changed. If E_f is changed, then restart from step 1.

C. Calculation of Σ-Flight Reachability and Controllability

The calculation of Σ-flight reachability and controllability sets over N_{CPG} is an important element in the determination of the CPG arcs set. These sets can be calculated in three different phases of the algorithm:

1) Choice of the Σ properties: some of them can be readily verified on CPG nodes (e.g., properties Σ_2 and Σ_3), others have impact on the solution of problem (11);

2) Solution of problem (11): if this is not solvable for the pair of nodes (E_i, E_j), then the (i, j) arc is not included in the CPG.

3) On the basis of a check of Σ admissibility in Δ. Nonadmissible arcs are a-posteriori removed from the CPG.

IV. Piecewise Polynomial Flight Paths in Constrained Environments

One of the most natural choices for flight trajectory segments parameterization in 2-D or 3-D spaces is the assumption of l-th order polynomial functions defined on a canonical basis. We have

$$\theta^f_{E_0,E_f}(\delta) \triangleq \Bigg\{ s(t) = f_\delta(t) = \sum_{i=0}^{l} \delta_{i+1} t^{l-i} \colon t \in [t_0, t_f], \delta_i = (\delta_{x1,i}, \ldots, \delta_{xn,i})^T \delta = (\delta_1^T \ldots \delta_p^T)^T \in \Omega \subseteq \mathbb{R}^p, p = n(l+1), E(t_0) = E_0, E(t_f) = E_f \Bigg\} \tag{12}$$

If Δ set is 2-D ($n = 2$) and, for the sake of simplicity, we assume $t = x$ ($p = (l + 1)$), the flight trajectory segments can be expressed as a $y'(x')$ function in a suitable local (O', x', y') reference system (see Fig. 4a):

$$\theta^f_{E_0,E_f}(\delta) \triangleq \{ y(x) = \delta_1 x^{p-1} + \ldots + \delta_{p-1} x + \delta_p \colon x \in [x_0, x_f], \delta \in \Omega \subseteq \mathbb{R}^p, y(x_0) = y_0, y(x_f) = y_f, \dot{y}(x_0) = \dot{y}_0, \dot{y}(x_f) = \dot{y}_f,) \} \tag{13}$$

Equalities $y(x_0) = y_0,\ y(x_f) = y_f,\ \dot{y}(x_0) = \dot{y}_0,\ \dot{y}(x_f) = \dot{y}_f$ can be readily converted into the following linear constraints in view of a formulation for problem (11):

$$A_E \delta = \begin{bmatrix} x_0^{p-1} & x_0^{p-2} & \ldots & x_0 & 1 \\ x_f^{p-1} & x_f^{p-2} & \ldots & x_f & 1 \\ (p-1)x_0^{p-2} & (p-2)x_0^{p-3} & \ldots & 1 & 0 \\ (p-1)x_f^{p-2} & (p-2)x_f^{p-3} & \ldots & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \ldots \\ \delta_{p-1} \\ \delta_p \end{bmatrix} = \begin{bmatrix} y_0 \\ y_f \\ \dot{y}_0 \\ \dot{y}_f \end{bmatrix} = b_E \tag{14}$$

If Σ_1 property is also requested in a discrete number N_2 of points along the trajectory: $|(\mathrm{d}^2 y/\mathrm{d}x^2)(x_i)| \le \bar{\ddot{y}}\ x_i \in [x_0, x_f], i = 1, \ldots, N_2$ additional linear inequality constraints arise:

$$A'_I \delta = \pm \begin{bmatrix} (p-1)(p-2)x_1^{p-3} & (p-2)(p-3)x_1^{p-4} & \ldots & 1 & 0 & 0 \\ (p-1)(p-2)x_2^{p-3} & (p-2)(p-3)x_2^{p-4} & \ldots & 1 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ (p-1)(p-2)x_{N_2}^{p-3} & (p-2)(p-3)x_{N_2}^{p-4} & \ldots & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} \delta_1 \\ \delta_2 \\ \ldots \\ \delta_{p-1} \\ \delta_p \end{bmatrix} \le \pm \begin{bmatrix} \bar{\ddot{y}} \\ \bar{\ddot{y}} \\ \ldots \\ \ldots \\ \bar{\ddot{y}} \end{bmatrix} = b'_I \tag{15}$$

Finally, the obstacle avoidance requirements can be approximated with a finite number of linear inequalities imposing that, in the (O', x', y') reference system, y' evaluated on a discrete set of N_o points has to be above (or below) the obstacle boundary (See Fig. 4b). We have

$$A''_I \delta = \pm \begin{bmatrix} x_1^p & x_1^{p-1} & \ldots & x_1^2 & x_1 & 1 \\ x_2^p & x_2^{p-1} & \ldots & x_2^2 & x_2 & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{N_o}^p & x_{N_o}^{p-1} & \ldots & x_{N_o}^2 & x_{N_o} & 1 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \ldots \\ \delta_{p-1} \\ \delta_p \end{bmatrix} \le \pm \begin{bmatrix} y_{b1} \\ y_{b2} \\ \ldots \\ \ldots \\ y_{bN_o} \end{bmatrix} = b''_I \tag{16}$$

If we assume that cost is proportional to length, the cost function can be converted into a quadratic function of δ parameters by approximating the flight path with a sequence of discrete linear segments, as shown in Fig. 4c. In this case the square of length L^2 can be rewritten as

$$L^2 \cong \sum_{k=1}^{M} d_k^2 = \sum_{k=1}^{M} d_{xk}^2 + d_{yk}^2 = d_x^T d_x + d_y^T d_y = c + \delta^T H^T H \delta \tag{17}$$

where

$$d_k = \begin{bmatrix} d_{xk} \\ d_{yk} \end{bmatrix} = \begin{bmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{bmatrix} \tag{18}$$

c is a constant depending on the choice of $(x_1, \ldots, x_{M+1})$, and d_y can be expressed as a linear function of δ:

$$d_y = \begin{bmatrix} d_{y1} \\ \ldots \\ d_{yM} \end{bmatrix} = \begin{bmatrix} x_2^{p-1} - x_1^{p-1} & x_2^{p-2} - x_1^{p-2} & \ldots & x_2 - x_1 & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{M+1}^{p-1} - x_M^{p-1} & x_{M+1}^{p-2} - x_M^{p-2} & \ldots & x_{M+1} - x_M & 1 \end{bmatrix} \times \begin{bmatrix} \delta_1 \\ \delta_2 \\ \ldots \\ \delta_{p-1} \\ \delta_p \end{bmatrix} = H\delta \tag{19}$$

Denoting with $Q = H^T H \ge 0$ a positive semidefinite symmetric matrix in $\mathbb{R}^{p \times p}$, with $A_I = [A_I'^T \ \ A_I''^T]^T \in \mathbb{R}^{n_I \times p}$, $b_I = [b_I'^T \ \ b_I''^T]^T \in \mathbb{R}^{n_I}$, $A_E \in \mathbb{R}^{n_E \times p}$, $b_E \in \mathbb{R}^{n_E}$ matrices defining linear

inequality and equality constraints, problem (11) is converted into the following quadratic cost optimization problem that is efficiently solved using quadratic programming (QP) algorithms [31]:

$$W_{i,j} = \min_{\delta \in \mathbb{R}^p} \delta^T Q \delta \quad \text{s.t.} \tag{20a}$$

$$A_I \delta \leq b_I \tag{20b}$$

$$A_E \delta = b_E \tag{20c}$$

If Δ set is 3-D ($n = 3$) assuming, for the sake of simplicity, that $t = x$ we have:

$$\theta^f_{E_0,E_f}(\delta) \triangleq \left\{ \begin{bmatrix} y(x) \\ z(x) \end{bmatrix} = \begin{bmatrix} \delta^y_1 x^l + \delta^y_2 x^{l-1} + \ldots + \delta^y_l x + \delta^y_{l+1} \\ \delta^z_1 x^l + \delta^z_2 x^{l-1} + \ldots + \delta^z_l x + \delta^z_{l+1} \end{bmatrix} : \right.$$
$$x \in [x_0, x_f], \delta \in \Omega \subseteq \mathbb{R}^p, y(x_0) = y_0, y(x_f)$$
$$= y_f, z(x_0) = z_0, z(x_f) = z_f, \dot{y}(x_0) = \dot{y}_0, \dot{y}(x_f)$$
$$\left. = \dot{y}_f, \dot{z}(x_0) = \dot{z}_0, \dot{z}(x_f) = \dot{z}_f \right\} \tag{21}$$

Adopting the same machinery as for the 2-D case, it is possible to convert problem (11) into a QP problem (20).

V. Simulation Examples

To show the practical capabilities of the proposed technique, three different scenarios are considered. A first 2-D example with two circular no-fly zones in a rectangular Δ region is used to illustrate the main application principles; a second 2-D example with a more complex scenario represents an urban environment in which to fly an unmanned aerial vehicle (UAV) at constant height; a third scenario, which is an expansion of the first example, shows how the proposed technique can be successfully applied in 3-D problems.

A. Two-Dimensional Flight-Path Generation with Two Circular No-Fly Zones

Consider the 2-D scenario shown in Fig. 5 with two circular no-fly zones in a rectangular domain. To simplify the choice of the CPG nodes we assume that pairs $(s_i, \dot{s}_i)$ are determined by the combinations of 115 points marked with circles in Fig. 5, with a discrete set of 36 equally spaced directions represented with a star of arrows centered in point (3, −1). Some of the nodes are uniformly distributed in the clear area, others on the no-fly zones contours. This choice produces a set of $N = 115 \times 36 = 4140$ nodes. The number of arcs to have a complete graph would be $N \times (N - 1)$ if we exclude autoconnections. However, some of the $(s_i, \dot{s}_i)$ pairs turn out to be non-Δ-compatible with the given domain. A further reduction of the number of arcs is obtained applying the following Σ-properties: 1) $d_2(s_i, s_j) < 5$ km for all i, j: the set of nodes reachable with one arc of polynomial from s_i is composed of nodes s_j falling within a 5 km radius circle centered in s_i. 2) $|\tan^{-1}(\mathrm{d}y/\mathrm{d}x|_i) - \tan^{-1}(\mathrm{d}y/\mathrm{d}x|_j)| \leq 70$ deg. Trajectory direction at node i should not differ from that at node j more that 70 deg in absolute value.

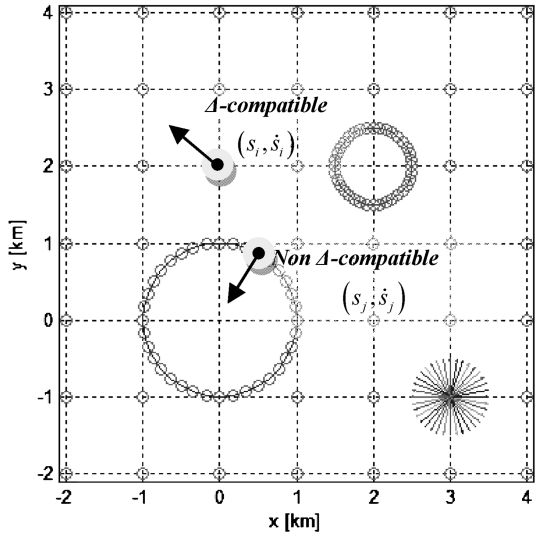

Fig. 5 Two-dimension Δ-space and obstacles. Circles represent nodes positions, arrows in point (3, −1) represent nodes directions. A CPG node is given by a Δ-compatible combination of one position with one direction.

With the application of the earlier mentioned rules only about 300 thousand arcs of the possible 17×10^6 are included in Υ_{CPG} for all the three cases considered:

1) Case 1: use of cubic polynomials to connect nodes [$l = 3$, $p = 4$ in Eq. (12)]: this choice does not leave degrees of freedom for cost minimization Eq. (20) and for additional geometrical requirements. The δ parameters are readily obtained by the solution of Eq. (20c). Nonadmissible arcs are identified a-posteriori checking the flight trajectories admissibility by means of a numerical inspection (see Fig. 6a).

2) Case 2: sixth order polynomials to connect nodes ($l = 6$, $p = 7$). Connection length is minimized, and a 300 m minimum radius of curvature is required (see Fig. 6b).

3) Case 3: same as case 2 without constraints on the radius of curvature (see Fig. 6c).

Figure 7 shows the results obtained applying the proposed algorithm to find trajectories connecting three different starting nodes ($x_1 = -2$, $y_1 = -1$, $\psi_1 = 10$ deg), ($x_1 = -2$, $y_1 = -1$, $\psi_1 = 110$ deg), ($x_1 = -2$, $y_1 = -1$, $\psi_1 = 160$ deg) to the same terminal node ($x_2 = 3$, $y_2 = 3$, $\psi_2 = 10$ deg).

The generation of the CPG on a Core 2 Duo processor PC takes about half an hour for the case of cubic polynomials whereas the optimal path can be computed in less than 100 ms with nonoptimized codes in the MATLAB environment. The QP problems are solved using an active set strategy also known as a projection method implemented in the MATLAB Optimization Toolbox [31]. Table 1 shows the optimal path lengths obtained for the three cases 1, 2, and 3.

The use of polynomial functions having orders higher than sixth does not produce significant improvements in terms of trajectory length for this case study. It is worth to notice that, if the polynomial order is too high, numerical problems may arise in the solution of the QP problems which can be partially solved using an orthonormal basis.

An analysis of the computational cost vs the polynomial order revealed that the effort for the construction of the CPG increases of a factor of about five when passing from cubic to sixth order polynomials. Such a large factor is mainly due to the fact that, with cubic polynomials, optimization problem (11) degenerates into the solution of a linear system. On the other hand, the use of tenth order polynomials almost doubles the computational time compared with fourth order polynomials. However, these are not general rules because much of the computational effort is related to the choice of other parameters and to the numerical implementation of the algorithm. On the other hand, minimum cost path search on the CPG is independent of the polynomial order adopted.

B. Unmanned Aerial Vehicle Flight-Path Generation in Urban Environment

A more complex scenario in terms of both obstacles density and geometry is now considered. In particular, the urban environment shown in Fig. 8 with buildings around which to fly a small rotor wing UAV at a constant height is considered. The combination of 311 points marked with dots with a discrete set of 36 equally spaced directions produces a set of $N = 11196$ nodes. The

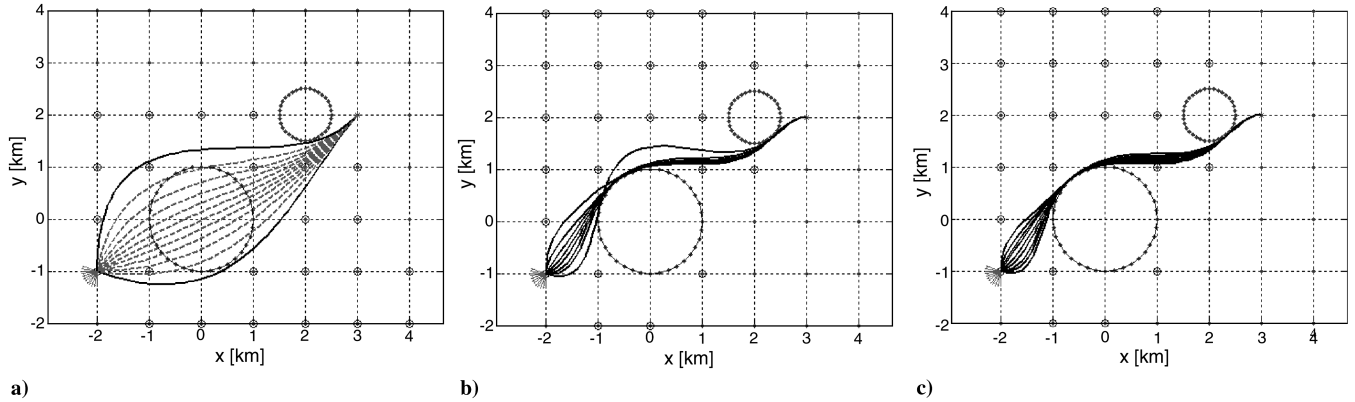

Fig. 6 Generation of CPG arcs connecting $(s_i, \dot{s}_i)$ to $(s_j, \dot{s}_{j_1}), \ldots, (s_j, \dot{s}_{j_k})$ using: a) cubic polynomial functions (dashed curves are nonadmissible in Δ), b) sixth order polynomial with minimum curvature radius of 300 m, c) sixth order polynomial.

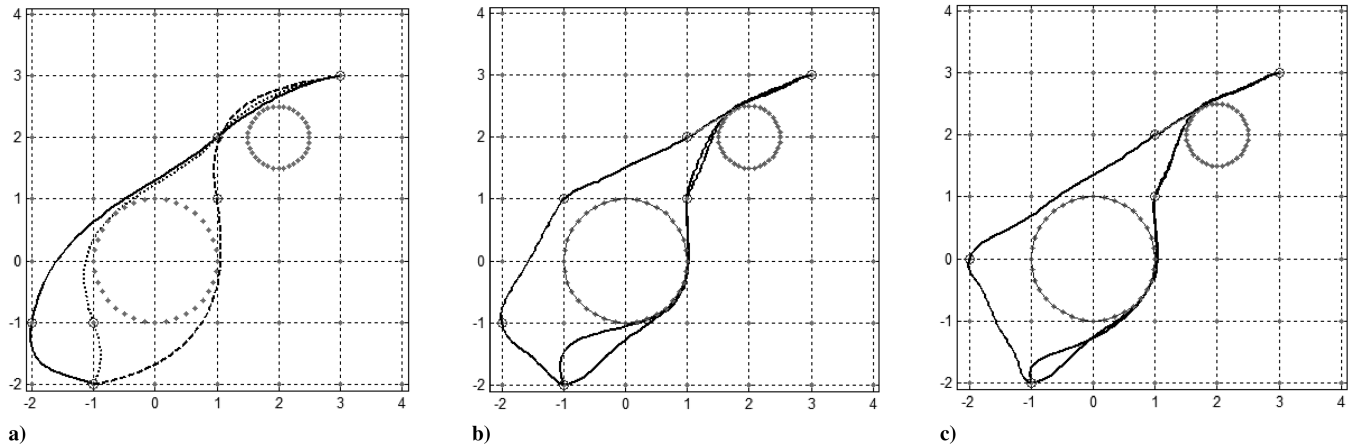

Fig. 7 Optimal paths connecting three different starting nodes having the same position and different directions to the same terminal node: a) cubic polynomials, b) sixth order polynomial with curvature radius limitation $R < 300$ m, c) sixth order polynomial without constraints on the curvature radius.

following Σ properties are applied to obtain Σ-admissible arcs in Δ: 1) $d_2(s_i, s_j) < 200$ m, 2) $|\tan^{-1}(\mathrm{d}y/\mathrm{d}x|_i) - \tan^{-1}(\mathrm{d}y/\mathrm{d}x|_j)| \leq 70$ deg, 3) cubic polynomial functions to connect nodes.

With the application of these rules only 1% of all the possible arcs connection nodes survives in the CPG.

Figure 8 shows results obtained applying the proposed algorithm to find five trajectories connecting different points of the CPG. The calculation of the CPG on a Core 2 Duo processor PC took about 1 h whereas the optimal path on the graph can be computed in less than 100 ms with a nonoptimized code in the MATLAB environment.

The proposed technique was also successfully implemented in feedback assuming that the optimal trajectory is updated every 50 m (sampling period $= 5$ s, constant speed $= 10$ m/s) according to the algorithm described in Sec. III.

An external disturbance was also considered in the simulation causing a constant (x, y) position additive error of -0.1 meter per meter both on x and y variables. This kind of disturbance can simulate the effect of an external constant wind coming from the northeast causing a displacement of the UAV which is not fully compensated by the trajectory tracking control system (see Fig. 9).

As for the computational time of the feedback trajectory calculation, the update of the CPG and the optimal trajectory to the target node were computed in less than 1 s at every step. The reason for such a short time, if compared with the initial construction of the CPG graph, is that only one node (measured E) and corresponding

Table 1 Trajectory lengths obtained with different polynomials and geometric constraints

	Starting direction		
Case	10 deg	110 deg	160 deg
1	7.46	7.31	8.28
2	7.00	7.30	8.19
3	7.00	7.14	8.16

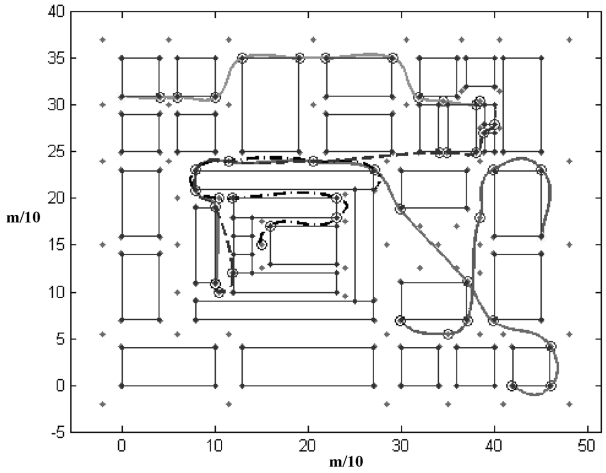

Fig. 8 Two-dimensional Δ-space for urban environment simulations. Five optimal trajectories connecting randomly selected CPG nodes are shown.

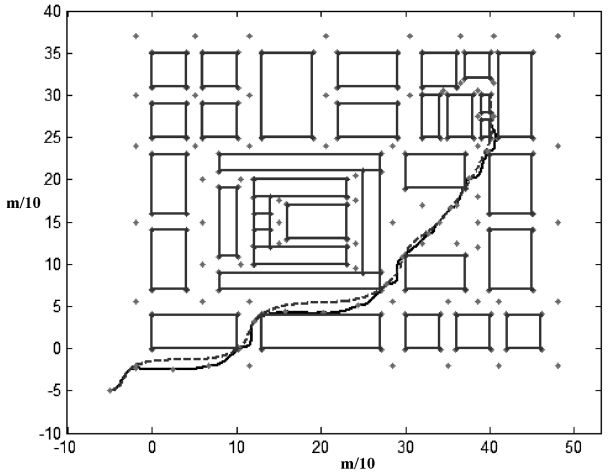

Fig. 9 Application of the proposed technique in feedback. Dashed line is the nominal trajectory obtained at the flight beginning. Continuous line is a trajectory updated in feedback every 50 meters assuming that an external disturbance causes a position error of −0.1 meter per meter on both x and y.

admissible arcs falling in the Σ-flight reachability set have to be computed to enlarge the graph.

The nodes in the space domain were placed on the intersections between streets, on the building corners, and uniformly spaced in large clear areas. This allows making easier changes of direction and possible flight trajectories close to buildings.

C. Three-Dimensional Flight-Path Generation Around Solid Obstacles

As the 3-D case, a simple scenario obtained from an expansion of simulation example described in Fig. 5 is considered. Circular obstacles were transformed into cylinders. Trajectory segments were required to be cubic polynomials in (x, y) variables, whereas climb and descent flight path was obtained using piecewise linear function in the z variable. A maximum (minimum) climb (descent) angle of + (−) 4 deg was also assumed.

Figure 10 shows a trajectory obtained connecting three way points corresponding to nodes ($x_1 = -1, y_1 = -2, z_1 = 0, \psi_1 = 150$ deg), ($x_2 = 3,\ y_2 = 3,\ z_2 = 0.3,\ \psi_2 = -50$ deg), ($x_3 = -2,\ y_3 = 3$, $z_3 = 0.5,\ \psi_3 = 150$ deg). The time for the CPG construction was about 5 h, whereas the trajectory calculation from node to node takes about 300 ms with a MATLAB nonoptimized code.

As for the choice of nodes, the same distribution used in the simulation example was replicated on 10 layers at different altitudes.

VI. Conclusions

The proposed CPG algorithm is a flexible tool to compute optimal trajectories in the presence of obstacles and no-fly zones because it can take into account any kind of geometrical shape of nonadmissible regions and additional geometrical constraints like the curvature radius and the climb angle. The calculation of the CPG requires the solution of a high number of quadratic programming problems. Once the CPG has been obtained, the minimum length flight trajectory connecting two arbitrary nodes of the graph can be computed within a short time. Because the inclusion of a new measured position direction in the core paths graphs can also be performed with a reduced effort it is possible to think of a real-time feedback implementation for the optimal trajectory calculation.

An interesting point to be further investigated is the criteria to choose the discrete set of CPG nodes, Σ-properties, and the cost function. A proper choice of these quantities can drastically reduce the computational burden especially for 3-D problems. The lack of general driving criteria calls for further studies.

It is worth noticing that the proposed approach could greatly benefit from the use of multiprocessor platforms because the CPG arcs construction algorithm can be readily formulated as a parallel process. Also symmetry properties can be used to reduce the computational times.

Further developments to take into account states of the aircraft dynamics, presence of uncertainties, and external disturbance for the possible real-time feedback implementation are also left for future work.

References

[1] Hargraves, C. R., and Paris, S. W., "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 4, 1987, pp. 338–342. doi:10.2514/3.20223

[2] Bulrish, R., and Kraft, D., "*Computational Optimal Control*," International Series of Numerical Mathematics Vol. 115, Birkhäuser, Boston, 1994.

[3] Betts, J. T., Biehn, N., Campbell, S. L., and Huffman, W. P., "Compensating for Order Variation in Mesh Refinement for Direct Transcription Methods," *Journal of Computational and Applied Mathematics*, Vol. 125, No. 1–2, Dec. 2000, pp. 147–158. doi:10.1016/S0377-0427(00)00465-9

[4] Binder, T., Blank, L., Dahmen, W., and Marquardt, W., "Grid Refinement in Multiscale Dynamic Optimization," Rheinisch Westfalischen Technischen Hochschule, Rept. LPT-2000-11, Aachen, Germany, 2000.

[5] Pesch, H. J., "Real-Time Computation of Feedback Controls for Constrained Optimal Control Problems, Part 2: A Correction Method Based on Multiple Shooting," *Optimal Control Applications and Methods*, Vol. 10, No. 2, 1989, pp. 147–171. doi:10.1002/oca.4660100206

[6] Stryk, O. V., and Bulirsch, R., "Direct and Indirect Methods for Trajectory Optimization," *Annals of Operations Research*, Vol. 37,

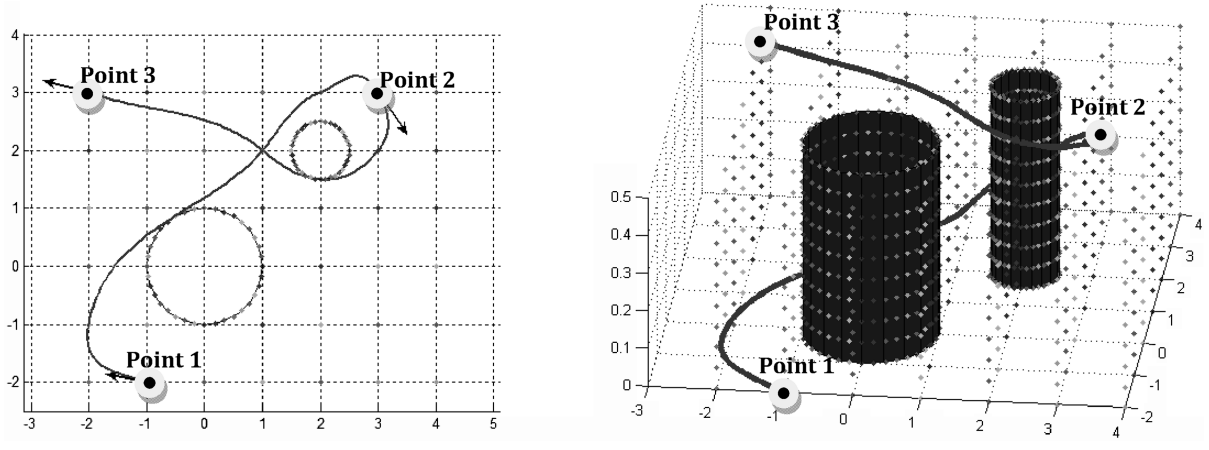

Fig. 10 Example of 3-D trajectory generation.

No. 1, Dec. 1992, pp. 357–373.
doi:10.1007/BF02071065

[7] Geiger, B., R., Horn, J. F., Sinsley, G. L., Ross, J. A., Long, L. N., and Niessner, A. F., "Flight Testing a Real-Time Direct Collocation Path Planner," *Journal of Guidance, Control and Dynamics*, Vol. 31, No. 6, 2008, pp. 1575–1586.
doi:10.2514/1.36533

[8] Laurent-Varin, J., Bonnans, J. F., Berend, N., Haddou, M., and Talbot, C., "Interior-Point Approach to Trajectory Optimization," *Journal of Guidance, Control and Dynamics*, Vol. 30, No. 5, 2007, pp. 1228–1238.
doi:10.2514/1.18196

[9] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.
doi:10.2514/2.4231

[10] Khatib, O., "Real Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, Vol. 5, No. 1, 1986, pp. 90–98.
doi:10.1177/027836498600500106

[11] Hwang, Y. K., and Ahuja, N., "A Potential Field Approach to Path Planning," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, 1992, pp. 23–32.
doi:10.1109/70.127236.

[12] Cen, Y., Wang, L., and Zhang, H., "Real-Time Obstacle Avoidance Strategy for Mobile Robot Based on Improved Coordinating Potential Field with Genetic Algorithm," *Proceedings of 16th IEEE International Conference on Control Applications*, MoC03.4, IEEE Publications, Piscataway, NJ, 1–3 Oct. 2007.

[13] Gavrilova, M., Cortes, J., and Jarvis, R., "Special Issue on Computational Geometry in Navigation and Path Planning," *IEEE Robotics and Automation Magazine*, Vol. 15, No. 2, June 2008, pp. 6–7.
doi:10.1109/MRA.2008.921539

[14] Duleba, I., and Sasiadek, J. Z., "Nonholonomic Motion Planning Based on Newton Algorithm with Energy Optimization," *IEEE Transactions on Control Systems Technology*, Vol. 11, No. 3, May 2003, pp. 355–363.
doi:10.1109/TCST.2003.810394

[15] Erzberger, H., and Lee, H. Q., "Optimum Horizontal Guidance Techniques for Aircraft," *Journal of Aircraft*, Vol. 8, No. 2, 1971, pp. 95–101.
doi:10.2514/3.44235

[16] Asseo, S. J., "In-Flight Replanning of Penetration Routes to Avoid Threats Zones of Circular Shapes," *Proceedings of Aerospace and Electronics Conference (NAECON 1998)*, IEEE Pulications, Piscataway, NJ, 1998, pp. 383–391.

[17] Yang, H. I., and Zhao, Y. J., "Trajectory Planning for Autonomous Aerospace Vehicles amid Known Obstacles and Conflicts," *Journal of Guidance, Control and Dynamics*, Vol. 27, No. 6, 2004, pp. 997–1008.
doi:10.2514/1.12514

[18] Anderson, E., "Extremal Control and Unmanned Air Vehicle Trajectory Generation," Masters Thesis, Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT, 2002.

[19] Anderson, E., Beard, R., and McLain, T., "Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles," *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 3, 2005, pp. 471–477.
doi:10.1109/TCST.2004.839555

[20] Frazzoli, E., Dahleh, M., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129.
doi:10.2514/2.4856

[21] Dever, C., Mettler, B., Feron, E., Popovic, J., and McConley, M., "Nonlinear Trajectory Generation for Autonomous Vehicles via Parameterized Maneuver Classes," *Journal of Guidance, Control and Dynamics*, Vol. 29, No. 2, 2006, pp. 289–302.
doi:10.2514/1.13400

[22] Schouwenaars, T., Mettler, B., Feron, E., and How, J., "Hybrid Model for Trajectory Planning of Agile Autonomous Vehicles," *Journal of Aerospace Computing, Information, and Communication*, Vol. 1, 2004, pp. 629–651.
doi:10.2514/1.12931

[23] Schouwenaars, T., Feron, E., and How, J., "Receding Horizon Path Planning with Implicit Safety Guarantees," *Proceedings of the American Control Conference*, Vol. 6, IEEE Publications, Piscataway, NJ, 2004, pp. 5576–5581.

[24] Singh, L., and Fuller, J., "Trajectory Generation for a UAV in Urban Terrain, Using Nonlinear MPC," *Proceedings of the American Control Conference*, Vol. 3, IEEE Publications, Piscataway, NJ, June 2001, pp. 2301-2308.

[25] Borrelli, F., Subramanian, D., Raghunathan, A. U., and Biegler, L. T., "MILP and NLP Techniques for Centralized Trajectory Planning of Multiple Unmanned Air Vehicles," *Proceedings of the 2006 American Control Conference*, IEEE Publications, Piscataway, NJ, June 2006.

[26] Yokoyama, N., and Suzuki, S., "Modified Genetic Algorithm for Constrained Trajectory Optimization," *Journal of Guidance, Control and Dynamics*, Vol. 28, No. 1, 2005, pp. 139–144.
doi:10.2514/1.3042

[27] Vaidyanathan, R., Hocaoglu, C., Prince, T. S., and Quinn, R. D., "Evolutionary Path Planning for Autonomous Air Vehicle Using Multiresolution Path Representation," *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, IEEE Publications, Piscataway, NJ, 2001, pp. 69–76.

[28] Hu, X. B., Wu, S. F., and Jiang, J., "On-Line Free-Flight Path Optimization Based on Improved Genetic Algorithms," *Engineering Applications of Artificial Intelligence*, Vol. 17, No. 8, 2004, pp. 897–907.
doi:10.1016/j.engappai.2004.08.015

[29] Duan, H. B., Ma, G. J., and Luo, D. L., "Optimal Formation Reconfiguration Control of Multiple UCAVs Using Improved Particle Swarm Optimization," *Journal of Bionic Engineering*, Vol. 5, No. 4, 2008, pp. 340–347.
doi:10.1016/S1672-6529(08)60179-1

[30] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001, ISBN 0-262-03293-7, pp. 595–601.

[31] Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, London, UK, 1981.