J Intell Robot Syst (2010) 57:143–170
DOI 10.1007/s10846-009-9367-1

Integration of Path/Maneuver Planning in Complex Environments for Agile Maneuvering UCAVs

Emre Koyuncu · N. Kemal Ure · Gokhan Inalhan

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 23 September 2009

Abstract In this work, we consider the problem of generating agile maneuver profiles for Unmanned Combat Aerial Vehicles in 3D Complex environments. This problem is complicated by the fact that, generation of the dynamically and geometrically feasible flight trajectories for agile maneuver profiles requires search of nonlinear state space of the aircraft dynamics. This work suggests a two layer feasible trajectory/maneuver generation system. Integrated Path planning (considers geometrical, velocity and acceleration constraints) and maneuver generation (considers saturation envelope and attitude continuity constraints) system enables each layer to solve its own reduced order dimensional feasibility problem, thus simplifies the problem and improves the real time implement ability. In Trajectory Planning layer, to solve the time depended path planning problem of an unmanned combat aerial vehicles, we suggest a two step planner. In the first step, the planner explores the environment through a randomized reachability tree search using an approximate line segment model. The resulting connecting path is converted into flight way points through a line-of-sight segmentation. In the second step, every consecutive way points are connected with B-Spline curves and these curves are repaired probabilistically to obtain a geometrically and dynamically feasible path. This generated feasible path is turned in to time depended trajectory with using time

E. Koyuncu (✉) · N. K. Ure
Controls and Avionics Laboratory, Istanbul Technical University, Istanbul, Turkey
e-mail: emre.koyuncu@itu.edu.tr

N. K. Ure
e-mail: ure@itu.edu.tr

G. Inalhan
Faculty of Aeronautics and Astronautics, Istanbul Technical University, Istanbul, Turkey
e-mail: inalhan@itu.edu.tr

scale factor considering the velocity and acceleration limits of the aircraft. Maneuver planning layer is constructed upon multi modal control framework, where the flight trajectories are decomposed to sequences of maneuver modes and associated parameters. Maneuver generation algorithm, makes use of mode transition rules and agility metric graphs to derive feasible maneuver parameters for each mode and overall sequence. Resulting integrated system; tested on simulations for 3D complex environments, gives satisfactory results and promises successful real time implementation.

Keywords Path planning · Maneuver planning · Dynamic feasibility · Agile maneuvering · Unmanned combat aerial vehicles

1 Introduction

Practical usage of Unmanned Air Vehicles has underlined two distinct concepts at which these vehicles are instrumental. First are the routine operations such as border or pipeline monitoring for which manned systems are expensive and inefficient. Second are scenarios such as an armed conflict reconnaissance or nuclear spill monitoring, in which there is a high risk for human life loss as the proximity to the scenario increases. In this work, we consider a specific case of the second type of scenarios which involves flying through a complex and dense city-like environment rather this be for reconnaissance or monitoring.

Although many kinodynamic motion planning methods that declares generating *dynamically feasible path* have been developed, they rarely can be used in practice especially for the aerial vehicles because of computational complexities. General kinodynamic motion planners require at least exponential time in that dimension of the state space of dynamical systems which is usually at least twice the dimension of the underlying configuration space [11]. In practice, kinodynamic planners are implementable only for systems that have small state-space dimensions. For example, the work presented in [11] suggests a path-planning relaxation which defines a class of maneuvers from a finite state machine, and uses a trajectory based controller to regulate the unmanned vehicle dynamics into these feasible trajectories. However, the trajectories to be controlled are limited to the trajectories generated by the finite state machine and the computational challenges of generating real-time implementable flight trajectories in 3D complex environments still remains as a challenge. Demonstration of path planner solution for flight in the 3D crowded *MelCity* model and landing to base is seen in Fig. 1.

The most important advantage of combination of path and maneuver planning that proposed in this work is related with generation of feasible agile flight trajectories. When the path planning problem is complicated with dynamic constraints on vehicle along with geometrical constraints, problem becomes challenging due to complexity of dimension. Therefore one cannot simply generate a trajectory that is dynamically feasible (feasible in the sense that it would be trackable by a control system in the flight envelope and actuator limits), relying on the path planning. This is where the maneuver planning comes in; by taking the advantage of working with a pre-defined dynamically feasible flight trajectory given by trajectory planner, it can identify and extract appropriate angular velocity and angular attack information

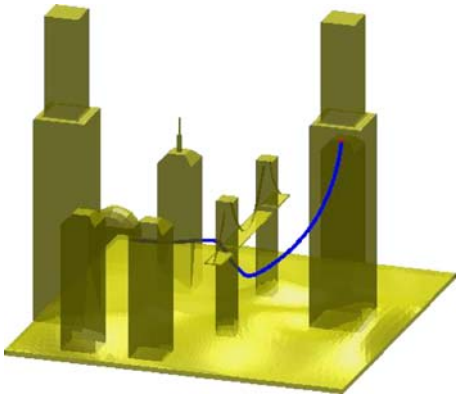

Fig. 1 UCAV flight demonstration in the 3D complex city-like environment and landing to its base

needed to create a maneuver profile without concerning with the trajectory generation and obstacle avoidance. However if the flight trajectory is generated without checking the velocity and acceleration bounds, maneuver planning layer will have hard time searching for angular velocities and angle of attack history that are in the feasible set. Therefore by sharing the dynamic feasibility checks between path planner and maneuver planner, these two layers covers their disadvantages and provides both dynamically and geometrically feasible flight trajectories and maneuver profiles for complex environments. This two step feasibility check is the main contribution of this paper to the field.

In the *Trajectory Planer* layer, we suggest a real-time implementable two-step path planner strategy. As the first step, 3D environment and the passages are rapidly explored using an RRT based planner. From this geometrically feasible but not dynamically feasible path, line-of-sight critical milestones are extracted. Although these milestones allow point-to-point flyable flight path segmentation, it does not necessarily correspond to a fast agile and continuous motion plan. To address this, as a second step, B-Spline method is used for generating C^2 continuous flight path that pass through these milestones. In face of geometrically and dynamically unfeasibility, generated path is probabilistically reshaped to eliminate the collisions and dynamically unfeasibility thanks to local support property of the B-Spline curves and at the end the *time scale* is adjusted to allow dynamic achievability.

Main contribution of the *Maneuver Planner* layer is a new perspective on maneuver/motion planning algorithms, which does not require any pre-build maneuver libraries and relies on the parameterized modal decomposition of arbitrary flight maneuvers. Integration with a path planning algorithm results in capability of generating feasible flight trajectory and maneuver profile which can be tracked by a switched control system where every maneuver mode is locally controllable. Integrated architecture of the Path/Maneuer Planning is demonstrated in the Fig. 2.

Rest of paper is organized as follows. In Sections 2 and 3, we gave detailed descriptions of the Path Planner and Maneuver Planner respectively. Literature surveys and related framework has been given at the beginning of these sections. In Section 4, example solution of the integrated Path/Maneuver Planning is demonstrated for the selected environment and complete computational time-table for the other example environments also has been given. The conclusions are discussed in Section 5.

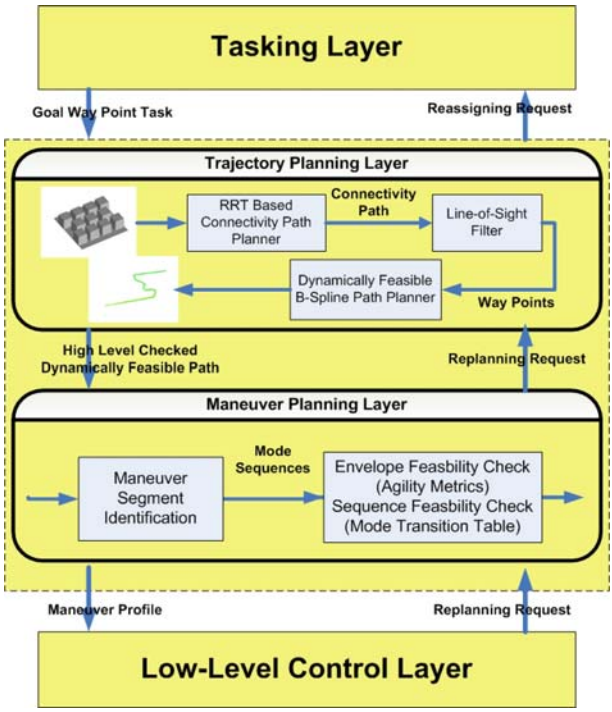

Fig. 2 Integrated planning architecture of UCAVs

2 Dynamically Feasible Path Planning Algorithm

For developing a real-time implementable planner, motion planning researches have been focused on sampling based approaches that rapidly search either the configuration or the state space of the vehicle. In the last few decades, sampling-based motion planning algorithms have shown success in solving challenging motion planning problems in complex geometries while using a much simpler underlying dynamic model in comparison to an air vehicle. Roadmap-based planners, like well-known Probabilistic Road Mapping (PRM) method as mentioned in [18], are typically used as multi-query planners (i.e. simultaneous search of the environment from different points) that connect these multiple queries using a local planning algorithm. PRM planners converge quickly toward a solution path, if one exists, as the number of milestones increases. This convergence is much slower when the paths must go through narrow passages. For complex environments, some extended algorithms are suggested for PRM like planners in [3] and [14]. Tree-based planners build a new roadmap for each query and the newly produced samples are connected to the samples that are already exists in the tree as in [13, 17], and [23]. Rapidly-Exploring Random Tree (RRT) is the most popular representative of tree-based planners that is an exploration algorithm for quickly searching high-dimensional spaces that have both global and differential constraints. Sampling-based planners, especially tree-based planners (RRT and single-query PRM variants), have been adapted to solve dynamically feasible paths that accommodate kinodynamic constraints.

Kinodynamic planning refers to problems in which the motion must satisfy non-holonomic and/or dynamic constraints. The main philosophy behind kinodynamic planning is searching a higher dimensional state space that captures the dynamics of the system [16, 23].

Gradual motion planning methods -our approach can be represented in this class- are recently proposed to solve complex path planning problem in cluttered environments. These methods first solve a relaxed form of the problem and then the approximate solution is refined to solve the original problem with a repairing method. In [15], a roadmap is initially generated by allowing some penetration into the collision workspace. Later, milestones are carried to collision-free space. In Iterative Relaxation of Constraints (IRC) method [1], first a relaxed version of problem is solved and then this coarse solution is used as a guide to solve original problem iteratively. The strategy of using an approximate solution to obtain a collision-free path is also used in Lazy PRM [2] and C-PRM [30]. In our earlier work [22], using a similar strategy, a Mode Based PRM method is refined with modal flight-path segments to obtain flyable trajectories.

From a deterministic perspective, B-Spline curves have been used in many dynamic path planning and control problem implementations. In [20], dynamic trajectory is generated with the minimum travel time for two-wheeled-driven type mobile robot. In [25] visibility-based path is modified to continuous feasible path via B-Spline curves. Using the well known local support property of B-Spline curves, real-time path modification methods are proposed for multiple mobile robots in [28] and robot manipulators in [7]. Constant acceleration time-scalable path generation method for the unmanned helicopters flying in the urban environments is used in our earlier work in [21] that we will use similar method but this time for the unmanned combat aerial vehicles.

In comparison, our method utilizes both the probabilistic and the deterministic aspects to obtain a real-time implementable planner strategy. In the first step, the algorithm rapidly explores the complex environment and the passages using an RRT planner because of its well quick spreading ability. In this part, our strategy focuses only finding an obstacle-free path that can be tracked from the initial point to the goal point with line segments in the configuration space. Vehicle's dynamic constraints are completely disregarded to decrease the computational time. This coarse obstacle-free path will be called as *connectivity path*. After finding the connectivity path, this path is filtered with the line-of-sight implementation to eliminate the points that cause long detours. Remained points that we call as *way points* naturally appear in entering and exiting regions of the narrow passages that are formed between the obstacles. An advantage of this refinement is that we can use these way points as guide-milestones that point hard regions and directions of the next coming hard regions in the environment.

In the second step of our strategy that dynamically feasible path is searched, every way-point connected with forth-order B-Spline curve and collision and dynamic feasibility cases are checked on curve. These forth-order B-spline curve presents C^2 continuous flight path. If the generated curve is not feasible, probabilistic repairing is achieved by randomized waypoint expansion on the connecting line path and the unit flight time is expanded to limit the accelerations within controllable regime. Since B-Spline curves have local support property, these repairing processes can be made on local path segments of interest. All path planning process is illustrated in Fig. 3.

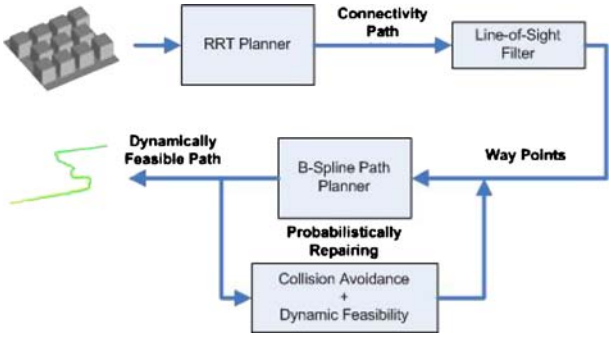

Fig. 3 Dynamically feasible path planning process

2.1 First Step: Connectivity Path

In real-time applications, planners should be able give a reliable answer in minimal permitted time slot. In motion planning problem, especially in complex environments, it is hard to say when planners should stop searching or change the searching strategy (i.e. switching to a more complex planner etc.). Moreover, finding an obstacle free geometrical path does not necessarily mean that a dynamically feasible path can be implemented by the vehicle exists. Although geometrical paths can be implemented via point to point navigation by the helicopter like vehicles with a inefficient manner but this flight strategy is not applicable for agile combat vehicle operations in under-threat environments. For the vehicles that have complex dynamics like combat aerial vehicles, directly searching in high dimensional state-space—as kinodynamic planners do—consumes long computational time to find a feasible path. Specifically, in our earlier work [22], we observed that before the major feasible path planning phase, defining the geometrical obstacle free path and trackable way points significantly accelerates the searching ability and decreases the total computational time of planner.

For finding connectivity path, RRT algorithm is used because of its rapid spreading ability. RRT is a considered as being an efficient algorithm to search even high dimensional spaces. However, one of the important drawbacks of using RRT as a stand-alone planner is biasing of the distribution of milestones towards the obstacle regions if the configuration space has large obstacles. Bi-directional RRT method shows performance more than single tree approach but it has also discontinuity problem on the connection points of the paths. Therefore, we choose to use single Goal-Biased RRT [23] approach that converges to goal configuration rapidly. We tested performance of the algorithm in different complex environments to conserve both rapid converging to solution and spreading abilities of the RRT, we chose the 50% percent goal biasing value. In this phase, we are only motivated by RRT's good property to obtain connectivity path. Our strategy does not focus on dynamically feasibility in this part of the path planner. Therefore, RRT algorithm is only used for searching configuration-space of the vehicle with primitive maneuvers that includes level and climbing flight and changing instantaneous heading direction. Construction of connectivity path algorithm is given as Goal Biased RRT Algorithm.

In Algorithm 1, to find the connectivity path, Goal Biased RRT method is used that one single tree is extended from the initial point. Each loop attempts to extend the τ tree first toward the random selected point m_{rand}, and second toward the goal

Algorithm 1: Goal Biased RRT Algorithm

```
   input  : initial configuration q_init and goal configuration q_goal
   output: connectivity path
1  τ ← q_init and i ← 1
2  repeat
3      Select random point m_rand in C and its neighbor point m_near in τ
4      Generate m_new is gone with trajectory e_new from m_near toward m_rand
5      if e_new is in C_free then
6          τ ← m_new and i + 1
7          if m_new is in end region then
8              break with success
9      Select neighbor point m_near of q_goal in τ
10     Generate m_new is gone with trajectory e_new from m_near toward q_goal
11     if e_new is in C_free then
12         τ ← m_new and i + 1
13         if m_new is in end region then
14             break with success
15     if i = N max iteration number then
16         break with fail
17 until end region is reached with success
18 Select connectivity path can be gone back from end region to initial point in τ
```

point by adding new points. To expand the tree, nearest point already within the τ tree to the sampled random point (in Line 3) and the nearest point to the goal point is selected (in Line 9) respectively in every one loop. *Generate* function generates new points m_{new} on the direction of the selected nearest points m_{near} at random selected distances as shown in Line 4 and 10. If direction angles exceed predefined limits, max direction angles are selected. These boundaries should be chosen according to vehicle's kinematic boundaries. If new generated point and trajectory is within obstacle-free configuration (checked in Line 5 and 11) then m_{new} is added τ tree as shown in Line 6 and 12. If τ tree reaches *end region* anytime, algorithm returns *connectivity path. End region* can be obtained within a tolerable capture region as explained in [19]. A solution of the algorithm in a complex city-like environment is illustrated in Fig. 4.

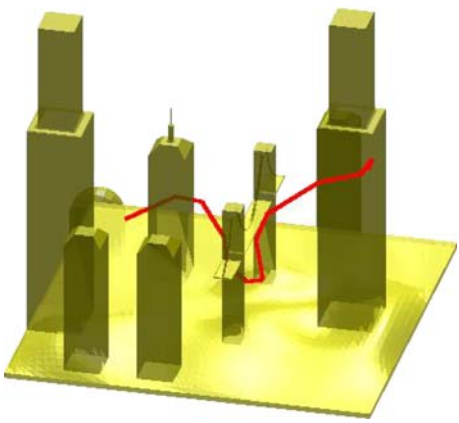

Fig. 4 Demonstration of the RRT based finding connectivity path algorithm

Algorithm 2: Line-of-Sight Filtering

```
input  : connectivity path
output: way point set WP
m_visib ← m_1 and m_i ← m_2
repeat
    Generate line ℓ_visib from m_visib to m_i
    if ℓ_visib is collide with C_obst then
        WP ← m_{i-1}
    else
        i + 1
until last point of connectivity path is reached
```

Because of the RRT's extending strategy and our simplifications, undesirable detours are frequently seen in obtained *connectivity path*. Since we only consider finding the obstacle-free region; we can simply remove the points that cause these detours. In this phase of our strategy, *connectivity path* is refined by Line-of-Sight Filter algorithm that erases points that result in useless fluctuations with using a line-of-sight arguments. As can be seen in Fig. 5, remaining points generally appear in nearby entering and exiting field of the narrow passages and inherently hard regions. Hence, these guard points also indicate where hard regions are beginning, what the direction of the next-coming hard region. These points also give a sense of agile maneuvering that are needed to fly over these points.

In this part of algorithm, a simple iteration checks if the selected point m_{visib} can connect with the previous points in *connectivity path* with a line segment without colliding with any obstacle. If the line segment collides with an obstacle, in other words, if the current point cannot be connected to the selected point, last connectible point is added to the *way point* sequence and the subsequent search continues from this point. This algorithm runs until the last point of *connectivity path* is reached with a line segment. A solution is illustrated in Fig. 5.

2.2 Second Step: Dynamically Feasible B-Spline Algorithm

In the first step, generated connectivity path with straight line segments result in a simple and implementable piecewise flight plan. However, this flight plan is not a fast

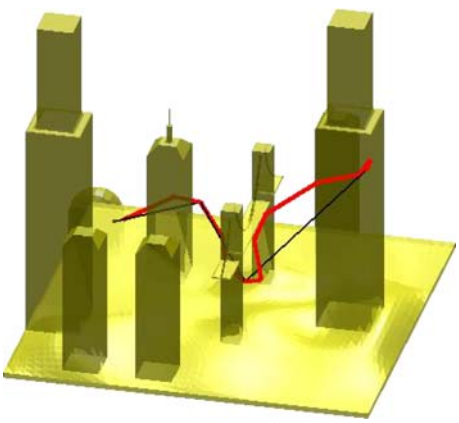

Fig. 5 Demonstration of the refining connectivity path with the line-of-sight filter algorithm

agile and continuous motion plan - a desirable feature in many complex unmanned combat aerial vehicles applications. After obtaining the way points (we will call remaining points as *way point* set) on the environment, many deterministic and sampling based path planner methods can be used to find the dynamically feasible path between the way points. Moreover, generated path must be continuous on the way points dynamically. During the path generation phase, since feasibility is desired, trajectory generation method should allow reshaping to supply collision avoidance and dynamic feasibility. Therefore, local support is also a desirable property on the path generation method. Local support means that the paths only influence a region of the local interest. Thus, obstacle avoidance and dynamic-feasibility repairing can be achieved without changing the whole shape of the generated path. B-Spline approach can supply these main requirements. An overview of B-Spline can be found in [29] .

Basically, output $C(u)$ can be defined in terms an k order B-Spline curve;

$$C(u) = \sum_{i=0}^{n} P_i N_{i,k}(u) \quad 0 \le u \le u_{max} \tag{1}$$

The coefficients P_i in Eq. 1 are called control points that will represent way points and pseudo way points in our approach.

The B-Spline basis functions $N_{i,k}$ are given by the Cox De Boor recursion;

$$N_{i,0}(u) = \begin{cases} 1, & u_i \le u \le u_{i+1} \\ 0, & otherwise \end{cases} \tag{2}$$

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u) \tag{3}$$

A B-Spline curve can be constructed from Bezier curves joined together with a prescribed level of continuity between them. A nondecreasing sequence of real numbers $U = \left[u_0 \ldots u_{max} \right]$ is called the knot vector. Frequently, the knot points are referred to as the break points on curve [26]. B-Spline basis function $N_{i,k}$ is zero outside the interval $\left[u_i, u_{i+k}\right]$ and non-negative for all values of k, i and u.

Derivatives of B-Spline curve exist on the knot vector span. Since, the kth-order B-spline is actually a degree $(k-1)$ polynomial, produced curve can be differentiated $k-1$ times.

$$C(u)^{(j)} = \sum_{i=0}^{n} P_i N_{i,p}^{(j)}(u)\, 0 \le u \le u_{max} \tag{4}$$

A valuable characteristic of the B-Spline curves is that the curve is tangential to the control polygon (formed by the control points) at the starting and ending point if some modifications are supplied. This characteristic can be used in order to define the starting, ending and transition directions of the curve by inserting an extra *pseudo control points* in directions which are defined according to way points' orientations assigned in the first step as explained in [26].

In this strategy, we choose generate forth-order path B-Spline (cubic polynomials) to obtain continuous inertial velocity and acceleration.

For generating B-Spline trajectory pass through way points, two *pseudo control points* are inserted after and before the every waypoint(except initial and last way point) in direction of the their tangent vector that these tangent directions are assigned during the path planning step. Note that; for the first way point, only further pseudo way point and for the last way point, only back pseudo way point should be added to way point set. The distance value between the way-point and the added pseudo-way points will define the transition velocity and acceleration on the way points of the path. Hence, C^2 continuity, in other words, continuous velocity and acceleration transition is naturally achieved on the way points.

For generate cubic B-Spline curves, we use specific nonuniform knot vector form $\mathbf{U} = [\ 0\ 0\ 0\ 0\ U_{mid}\ 1\ 1\ 1\ 1\ ]$ to obtain the coincidence between the first and last control points and the first and the last ends of the generated B-Spline curve respectively. Detailed information about this effect can be found in [29] as *open uniform knot vector* effect. $\mathbf{U_{mid}}$ is represents middle *knot vector* that is initially uniformly distributed in $(0, 1)$ interval -number of points depends on the number of control points- and algorithm can add new knot points to the vector without preventing its uniform form. We choose using arbitrary $[0, 1]$ interval for parameter u such that it represents unit-time scale [38]. This property is later used to allow dynamic feasibility via time scaling (i.e. expanding the time horizon of the maneuver). Overall B-Spline path planning algorithm can be demonstrated in Algorithm 3.

This algorithm tries to find dynamically feasible B-Spline curve passes through on the way points with their heading angles and runs until the last way point is connected with a feasible path. Initially, m number way points - generated in the first step- are added in *control point* set **P** as seen in Line 2. Then, for every way point, except first and last way point, *back* pseudo way point $gp_{i,b}$ and *further* pseudo way point $gp_{i,f}$ is located on random selected distance d from way points on their heading tangent directions and these *pseudo way point* set **gp** is also added in *control point* set **P** that is demonstrated in Line 3 to 6. Different from other way points, for the first way point, only *further pseudo way point* $gp_{1,f}$ is located and for the last way point, only *back pseudo way point* $gp_{m,b}$ is located. Thus, algorithm initially begins with $3m - 2$ control points where m indicates that number of way points but note that the algorithm can add new control points during to implementation to repair the B-Spline. As initial form, open uniform knot vector form that is chosen in unit interval [0,1] is used in our implementation as shown in Line 8. As depicted in Line 9, in a loop, B-Spline basis function is generated via u parameter and then collision and dynamic feasibility is checked on every discrete point of the curve as shown in Line 10. Since velocity and acceleration on the path is a function of time, for each point of the trajectory we have to check if the instantaneous velocity and acceleration is within the limits of the flight envelope. This Dynamic Feasibility check is done by checking the first and the second derivatives of the B-Spline curve which gives the velocities and accelerations of the aircraft respectively. If these velocity and acceleration values are within the limitations of the aircraft (*flight envelope*) using chosen *unit time scale*, generated path segment is accepted as dynamically feasible. One of the most critical step in the path planning layer is to determine the velocity and accelerations on the trajectory; if the aircrafts velocity constraints are not taken into account, maneuver planning layer would not be able to find feasible maneuver reference from this generated trajectory. This concept is used for giving a sense on *Dynamic Feasibility* on the side of the Path Planning layer as a first step

Algorithm 3: Dynamically Feasible Trajectory Generating with B-Spline

input : way point set $\mathbf{g} = [g_1 \dots g_m]$
output: *dynamically feasible path*
1 *TimeScale* ← unit-time scale
2 $\mathbf{P} \leftarrow \mathbf{g} = [g_1 \dots g_m]$ as control point set
3 **foreach** *element* g_i *of the* **g do**
4 **Insert** back pseudo way point $gp_{i,b}$ to the random selected distance d from g_i on its negative heading direction
5 **Insert** further pseudo way point $gp_{i,f}$ to the random selected distance d from g_i on its positive heading direction
6 $\mathbf{P} \leftarrow \mathbf{gp} = [gp_{1,f}\ gp_{2,b}\ gp_{2,f} \dots gp_{m-1,b}\ gp_{m-1,f}\ gp_{m,b}]$ as control point set
7 $\mathbf{U} \leftarrow [0\ 0\ 0\ 0\ \mathbf{U_{mid}}\ 1\ 1\ 1\ 1]$
8 **for** $u \leftarrow 0$ **to** 1 **do**
9 **Evaluate** B-Spline basis function
10 **Check** collision and dynamic feasibility
11 **if** *collision occurs or point of the spline is not dynamically feasible* **then**
12 **Change** locations of the pseudo way-points $gp_{i,b}$,$gp_{i,f}$ of the interest curve segment according to u
13 **Set** u value as indicates that local interest curve segment $-k/2$ segment
14 $m_1 + +$
15 **if** $m_1 > M_1$ **then**
16 **Change** locations of the way points g_i and its pseudo way-points $gp_{i,b}$,$gp_{i,f}$ of the interest curve segment according to u in its small region
17 **Set** u value as indicates that local interest curve segment $-k/2$ segment
18 $m_2 + +$ and $m_1 = 0$
19 **if** $m_2 > M_2$ **then**
20 $\mathbf{P} \leftarrow P_{new}$ as new control point around unfeasibility and update knot vector
21 **Set** u value as indicates that local interest curve segment $-k/2$ segment
22 $m_3 + +$ and $m_1, m_2 = 0$
23 **if** $m_3 > M_3$ **then**
24 **Change** *TimeScale* to insert min/max values of velocity and acceleration in dynamically feasible region
25 **Set** u value as 0
26 $m_4 + +$ and $m_1, m_2, m_3 = 0$
27 **if** $m_4 > M_4$ **then**
28 **break** with *fail*
29 **Set** *TimeScale* as generated path can be implemented as possible as in optimal time
30 **return** B-Spline*Trajectory*

Dynamic Feasibility check. The more precise Dynamic Feasibility check will be done on Maneuver Planning layer as explained in following section. Maneuver planer layer may also request from Path Planner to re-plan of the trajectory if dynamic feasibility can not be repaired during Maneuver planning using the *Replanning request* connection seen in demonstration of the all interactions between the layers (Fig. 2).

If feasibility cannot be obtained during the path planning, repairing methods are implemented hierarchically. Firstly, location pseudo control points are slid on the same tangent directions as shown in Line 12 and the algorithm decreases the u value from current interest curve segment $-k/2$ curve segment where k is represents order that is four in this implementation. Note that, B-Splines' local support property

allows local control over the generated spline. Specifically, this control is over the curve segments with $\pm k/2$ polygon spans around the displaced or newly added point. Therefore, when any changes is made on spline, instead of evaluate all spline over and over again, u value is decreased by value interval that spans local interest. Note that, all the repairing steps are tried with predefined threshold iteration times illustrated as M_is in the algorithm. After predefined number of trials, if the spline cannot be repaired, the way point and its pseudo way points within local interest are carried to a new collision-free locations and these locations are chosen as small random-selected distance away from the prior locations as seen in Line 16.

If the B-Spline still cannot be repaired, new control point P_{new} is added in control point vector **P** around the region in which collision or infeasibility has occurred (Line 20). Since we know the infeasible knot value and its interval in the knot vector, new knot point is added to the midpoint of the infeasible knot interval. Hence, only a limited interval of the knot vector **U** is updated. Reader should remember, only $\pm k/2$ polygon spans around the displaced or newly added point will be effected with this change.

If all these processes can not repair the path, the *time scale* value is scaled in Line 24 to reallocate the min-max velocity and acceleration interval of the trajectory (time depended path) within the dynamically feasible interval that can be achieved by the aircraft (falls into limits of flight envelope). For example in Fig. 6, search begins in a point outside the flight envelope and in two steps it is moved into limits of flight envelope by time scaling. Finding the feasible velocity-acceleration by this method is similar to what authors had done for finding the feasible modal inputs in the [22]. Note that other than flight envelope check there isn't any dynamic model involved in path planning layer. All the other feasibility problems (actuator saturation, attitude discontinuity etc.) are left to maneuver planning layer.

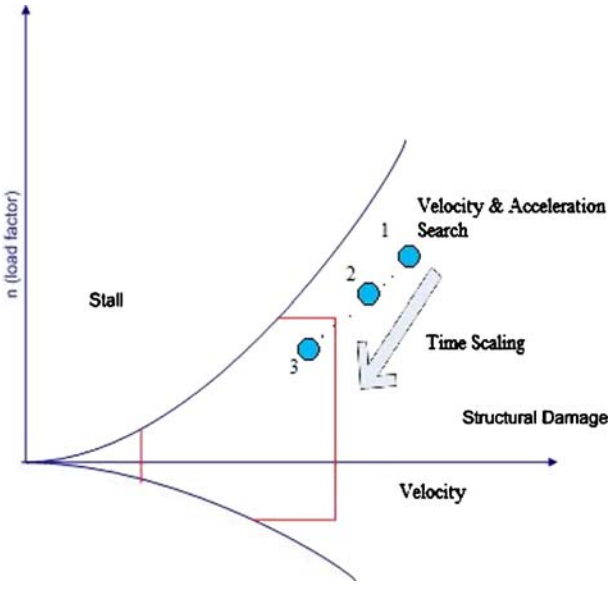

Fig. 6 Feasible velocity—acceleration search on flight envelope

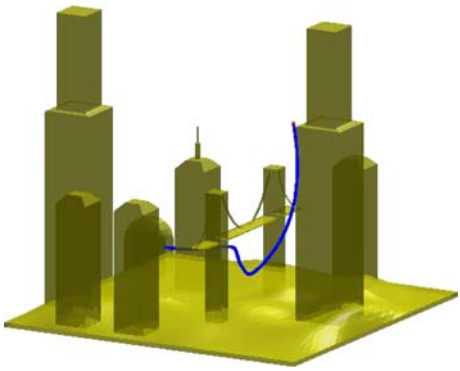

Fig. 7 Dynamically feasible path solution of the B-Spline planner algorithm in the complex 3D environment for UCAV

The end result is a time expanded or shortened flight path. Dynamic feasibility of the all generated spline should be checked from the beginning considering to newly changed time scale. After the generating B-Spline, *TimeScale* is also set again to fly over the all path in optimal time interval. Dynamically Feasible Path solution in the *MelCity* model for the UCAV is demonstrated in Fig. 7.

3 Maneuver Planning Algorithm

Multi Modal control framework basically consists of decomposition of the arbitrary maneuvers into set of maneuver modes and associated maneuver parameters. The main aim of the work was the help to reduce complexity of the both planning and control part. Complexity of maneuver planning part has been reduced by reducing the dimension of the problem (modal sequence has strictly lower dimension than state space description) and control part was relaxed by designing specific controllers for each mode and switch between them in order track maneuver mode sequence instead of designing a single controller for maneuver tracking over full flight envelope. In this paper we shall only focus on maneuver planning part, discussion of the switched control layer can be found on [34] and [35]. Motion planning problem for aerospace vehicles are complicated by the fact that, planners based on optimal performance begins to fail in means of computation, when one takes into account of constraints related with dynamical equations of aircraft. Due to fact that, aircrafts state space is at least 12 dimensional, input-state search becomes too complicated; therefore such planners are only successful for vehicles with small state space dimensions [11]. To reduce the complexity of this problem, motion description languages and quantized control concept have been adopted into motion planning [4]. Motion description languages, makes use of classified combination of simplified control laws to track generalized outputs. Most of these languages are strongly connected with the concept of hybrid systems, which in general, classifies the motion by using discrete states which switches in between according to input and state information and each discrete state having its own continuous dynamics. A subclass of such languages which is based on classification of behavior (or reaction) of the dynamic systems, has been successfully adapted for non-holonomic robotic

systems [36]. More recently, closed loop hybrid control systems were developed based on linear temporal logic for the same purpose by [10]. For aerospace vehicles, a hybrid model for aircraft traffic management was developed in [5]. Study showed that, hybrid system representation gives opportunity to calculate reachable sets of the system and design hybrid control laws to drive the system to safe states [5]. Frazzoli [11] suggested a maneuver automaton, which uses a number of feasible system trajectories to represent the building blocks of the motion plan of the aircraft, and a trajectory based (based on maneuver regulation principle) control system which asymptotically regulates the actual trajectory to the trajectory generated by maneuver automaton. However, motion plans and controllable trajectories are restricted to the library of the maneuver automaton. Such libraries can be built by using interpolation between feasible trajectories [6]. Feron [33] extended this system for online planning of feasible trajectories in partially unknown environments by using receding horizon iterations. Description of aircraft dynamics from hybrid system point of view has been studied previously in [11, 12, 27]. These works have been successful in using the advantages of hybrid system methodology in control of both single and multiple aircrafts. However, these approaches did not include the full flight envelope dynamics of the aircraft. Specifically, both mode selection and controller design is strictly based on selected maneuvers; therefore controllability is limited [11, 12, 27] to these predefined trajectories. In our work, we make use of parameterized sub maneuvers which builds up complex maneuver sequences. We show that it is possible to cover almost any arbitrary maneuver and the entire flight envelope by this approach.

In this section we shall detail the maneuver planning layer. As it is indicated in Fig. 2 this layer is below the path planning layer, in the sense that it receives path-flight trajectory information from PP layer and shapes the trajectory by adding aircrafts attitude angular rates time history while checking the dynamic feasibility of these maneuvers. Maneuver planning takes advantage of multimodal control framework described in [22] and [35].

Multi Modal control framework basically consists of decomposition of the arbitrary maneuvers into set of maneuver modes and associated maneuver parameters. The main aim of the work was the help to reduce complexity of the both planning and control part. Complexity of maneuver planning part has been reduced by reducing the dimension of the problem (modal sequence has strictly lower dimension than state space description) and control part was relaxed by designing specific controllers for each mode and switch between them in order track maneuver mode sequence instead of designing a single controller for maneuver tracking over full flight envelope. In this paper we shall only focus on maneuver planning part, discussion of the switched control layer can be found on [34] and [35].

In this section, we will first briefly review the multi modal control framework. Then, step by step maneuver generation algorithm is explained.

3.1 Brief Review of Multi Modal Control Framework

Basically, main idea is to divide an arbitrary flight maneuver into smaller maneuver segments (called maneuver modes) and associated maneuver parameters (called modal inputs). If the maneuver modes are found properly, one can describe any

maneuver by giving the maneuver mode sequence. This idea makes use of the fact that, 12 states of the conventional aircraft are not independent during all maneuvers and one does not need to give all the state trajectory of the aircraft to define a maneuver.

Complete list of modes and their modal inputs along with state constraints on each mode was given in [35]. We review this table (Table 1) below since the same modes will be used during design of the algorithms. Note that for 6 DOF flight state space variables are chosen as:

$$X = \left[V_T \ \alpha \ \beta \ \varphi \ \theta \ \psi \ P \ Q \ R \ n_p \ e_p \ h \right]^T \tag{5}$$

Where, V_T is the total speed, α and β are aerodynamic angles, angle of attack and sideslip angle respectively. $\Phi = \left[\varphi \ \theta \ \psi \right]^T$ is 3-2-1 Euler angle set(which are replaced with Quaternions during simulations). $\omega = \left[P \ Q \ R \right]^T$ is the angular velocity vector in the body axes, and $\rho = \left[n_p \ e_p \ h \right]^T$ is the set of Cartesian coordinates. Flight path Euler angles (or wind axis angles) are denoted with Φ_w. Note that, safety mode on Table 1 is an artificial mode which serves as an emergency break for the control framework, in the case that aircraft goes out of domains of a particular mode or becomes unstable, it recovers the aircraft by setting it back to level flight.

So via Table 1 state trajectory $X(t)$ is replaced by the triplet (q_i, σ_i, τ_i), $i = 1$, 2..., N , where q_i is the i^{th} maneuver mode, σ_i is the set of modal input values associated with i^{th} mode, and τ_i is the time duration of the i^{th} mode. is the number of maneuver modes in total. This triplet is abbreviated as simply "modal sequence". Since this approach is of lower dimension than state space description, it reduces the complexity of motion planning problems.

Another advantage of maneuver decomposition methodology, other than reduction of the order of the problem, is; it gives opportunity to design specific controllers for each mode of the system. This task is very natural to the system, because each set of modal inputs also serve as a reference output profile for a tracking controller. It is also obvious that if a successful (and possibly nonlinear controller due to coupled nonlinear dynamics of agile maneuvers) each mode is designed, one can gain control over full flight maneuver sequence by switching the controllers. For the assignment of such a switched controller family see [35], and for design of an actual system based on Higher Order Sliding Modes see [34].

Table 1 Flight modes and modal inputs

	Mode	State constraints	Modal inputs
q_0	Level flight	$\dot{h} = 0, (\dot{\varphi}, \dot{\theta}, \dot{\psi}) = 0$	V_T, α
q_1	Climb/descent	$(\dot{\varphi}, \dot{\theta}, \dot{\psi}) = 0$	$V_T, \left(\dot{h}, \theta_w\right)$
q_2	Roll	$(\dot{\theta}, \dot{\psi}) = 0$	$V_T, \int P_w dt$
q_3	Longitudinal placeLoop	$(\dot{\varphi}, \dot{\psi}) = 0$	$(V_T, r_{loop}), \dot{\theta}$
q_4	Lateral placeLoop	$\dot{h} = 0, (\dot{\varphi}, \dot{\theta}) = 0$	$(V_T, r_{loop}), \dot{\psi}$
q_5	3D Mode	{}	V_T, P, Q, R/ $V_T, \varphi_w, \theta_w, \psi_w$
q_6	Safety	{}	{0, 1}

However to ensure that controllers are capable of tracking the maneuver, one must guarantee that maneuvers are feasible in the sense that they are executable by a piloted system, thus satisfying the saturation envelopes. An additional criterion for switching stability is the smooth connection of each mode to another in terms of kinematic parameters, if they are not; discontinuous jumps in output profiles while switching the control system can result in degradation of tracking performance or even instability.

In the next subsection, structure of the maneuver generation algorithm based on the multi modal control framework will be given.

3.2 Maneuver Generation Algorithm

Main aim of this section is to develop a maneuver planning algorithm, which extracts the mode sequence from given flight trajectory and derive modal inputs (maneuver parameters) for each mode based on the feasibility constraints. General structure of the algorithm is shown on Fig. 8.

In the Fig. 8, each block represents a part of the algorithm directed for a specific task. After receiving flight path with velocity history from path planning layer, segment identification part decomposes the flight path into a sequence of maneuver modes. Next feasible modal input for each mode is determined by the help of agility metric graphs. Mode Transition table checks if every two adjacent mode is compatible with each other, if not, transition modes are placed between each mode for sequential feasibility, then modal inputs for these transition modes are found similar to previous step. Finally, a time interval for each mode is determined and maneuver profile is generated.

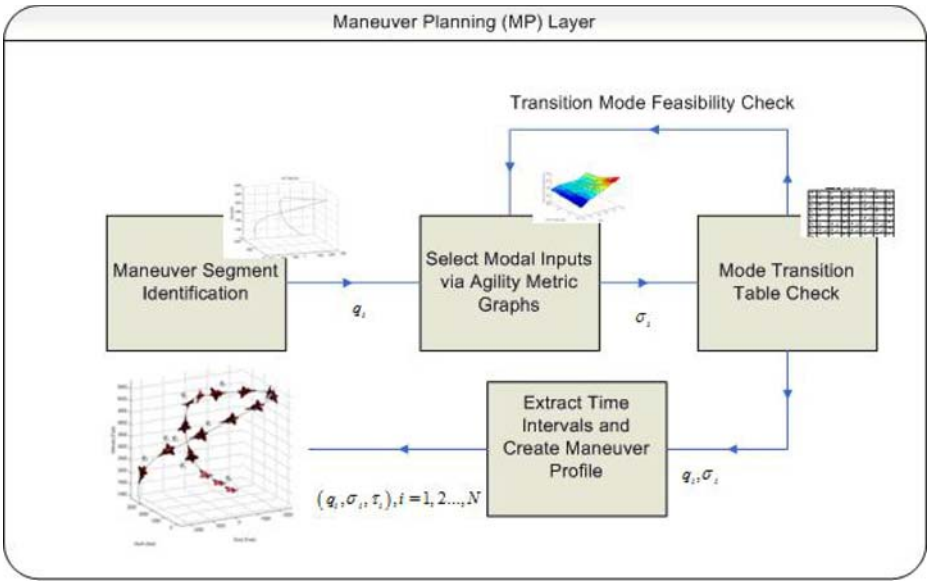

Fig. 8 Overview of the maneuver planning algorithm

3.2.1 Maneuver Segment Identification

Initially, path planning algorithm provides the flight trajectory history, $n_p(t)$, $e_p(t)$, $h(t)$. From this data it is easy to recover by the velocity variables in wind axes via Eq. 6 [31]

$$\begin{bmatrix} \dot{n}_p \\ \dot{e}_p \\ \dot{h} \end{bmatrix} = V_T \begin{bmatrix} \cos\theta_w \cos\psi_w \\ \cos\theta_w \sin\psi_w \\ \sin\theta_W \end{bmatrix} \Rightarrow \begin{array}{l} \psi_w(t) = \tan^{-1}\left(\dfrac{\dot{e}_p}{\dot{n}_p}\right) \\ \theta_w(t) = \tan^{-1}\left(\dfrac{\dot{h}}{\dot{e}_p}\sin\psi_w\right) \\ V_T(t) = \sqrt{\dot{n}_p^2 + \dot{e}_p^2 + \dot{h}^2} \end{array} \tag{6}$$

Next flight trajectory is divided into waypoints, and by comparing the velocity variables between each waypoint, maneuver modes listed on Table 1 can be determined via Table 2.

On the Table 2. "C" means constant and "T" means time varying. Table is self-explanatory; for example in level flight, flight path angle (or wind axis pitch angle) must be zero so that aircrafts altitude doesn't change, while heading can take any value as long as it doesn't vary with time (i.e. zero derivative). This straightforward logic is applicable to every mode on the table.

Note that, at this point it is not possible to recover roll mode from given trajectory, since B-Splines (or any curve in space) do not carry this information. In the methdology, roll mode is counted as a transition mode between maneuver segments and inserted by maneuver planning layer. How the roll mode is inserted into maneuver profile is explained in the subsection: 'Satisfying the Sequential Constraints'.

Since mode labeling action only depends on wind Axes Euler Angles, it is possible to recover the mode sequence and velocity on each mode from given flight trajectory. However complete modal sequence is incomplete because modal input sequence cannot be recovered without attitude history. In the next step each mode will be analyzed via agility metric graphs, and modal inputs will be recovered from flight equations or agility metric graphs.

3.2.2 Selection of Modal Inputs

Since velocity is already given by path planner, only remaining modal inputs to be obtained are angular velocities which rely heavily on information of angle of attack. It is obvious that angle of attack cannot be determined alone from given data, so we have to obtain it from some other methods. Selection of angle of attack is a very critical part of creating agile and feasible path, because larger values of angle of

Table 2 Wind axes euler angles identification table

	θ_w	$\dot{\theta}_w$	ψ_w	$\dot{\psi}_w$
Level flight	0	0	C	0
Climb / descent	C	0	C	0
Roll	C	0	C	0
Lon. loop	T	T	C	0
Lat. loop	C	0	T	T
3D Mode	T	T	T	T

attack may cause stall whereas smaller values can result in aerodynamic inefficiency or event saturation of true inputs of the aircraft (control surfaces and throttle). Therefore we have to consult the flight and saturation envelope of the aircraft for a healthy selection of agility metrics.

One way to combine flight envelope, actuator saturation envelope and aggressiveness properties is to use agility metrics. Agility metrics were initially developed for comparing agility characteristics of fighter aircrafts, because classical metrics (such as thrust to weight ratio) were incapable of showing the true agility potential of these aircrafts. Agility metrics are usually given in terms of aircraft states or a time for a specific task (such as time to go through 90 degrees of roll angle), then these metrics are plotted against velocity or angle of attack for various aircrafts.

The strategy for this section consists of specifying an agility metric for each mode (such that associated metric is closely related to the dominant states of each mode) then evaluating the metric from nonlinear flight model simulations for various angles of attack and Mach number (thus building a library of agility metrics for feasible velocity-angle of attack intervals). Then for each mode due to fact that velocity (thus the Mach number) is specified from path planning, is it possible to select an angle of attack from feasible interval (to speed up the process it is selected randomly, it may also be possible to optimize it for some cases, but optimization is not primary objective, because it will decelerate the process and complicate its real time implementability).

In this study a 6 DOF high fidelity nonlinear F-16 model is used for simulations [9]. Selected agility metrics were take from various NASA reports [8, 24, 37]. Selected agility metric for multi modal control frameworks are;

Level and Climbing Flight For level and climbing/diving flight total speed and acceleration capability is the most important parameter. Maximum and minimum achievable speed depends heavily on available power and thrust. Selected agility metric is power onset/loss parameter which can be written as:

$$\dot{P}_s = \frac{d}{dt}\left(\frac{V_T\,(T-D)}{W}\right) \tag{7}$$

Where T is thrust, D is drag and W is weight. This agility metric quantifies maximum thrust and drag of the aircraft, which determines the total speed and acceleration capability.

Roll Mode For rolling motion either average roll rate or time to go through certain roll displacement can be used. Second one is more convenient as it gives transient performance more clearly. For specific angle 90 degrees can be used, because most of the rolling maneuvers consists of rolling aircraft to side (knife edge), or inverting it (180 degrees). Therefore selected agility metric is

$$TR_{90} \tag{8}$$

which means “Time To Go Through 90 Degrees Roll Angle”. 3D plot of this metric is shown in Fig. 9.

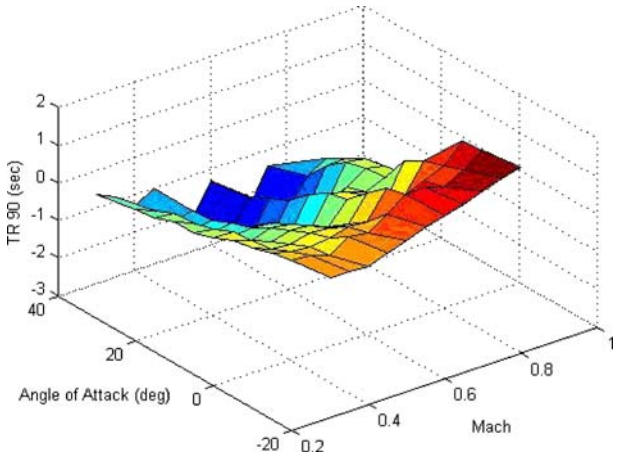

Fig. 9 Time to capture 90 deg roll angle metric for sea level

Longitudinal Loop For pitch up/down motion, very simple and convenient metric is average pitch rate which can be written as

$$Q_{avg} = \frac{\int_{t\_1}^{t_2} Q dt}{t_2 - t_1} \tag{9}$$

Lateral Loop For turning performance load factor and turning radius is chosen as a predominant factor, by using the simple kinematic equation [31]:

$$r_{loop} = \frac{V_T^2}{g(n^2 - 1)} \tag{10}$$

3D Mode In 3D mode both rolling and pitching motion becomes dominant; therefore we seek a metric which can combine these two properties. An appropriate metric is loaded roll which is given by the formula

$$PN = p_w N_{z,w} \tag{11}$$

This is simply the product of roll rate in wind axes and normal acceleration in wind axes. This metric belongs to torsional agility and combines the rolling motion of the aircraft with bending of the flight path.

Once the angle of attack history for each mode is obtained (note that final and initial angle of attack for each mode is selected appropriately for continuity), it is possible to recover all of the modal inputs for each mode shown in Table 1. From the rotation matrices and assumption of $\beta = 0$, $\psi_w = \psi$ it is possible to recover:

$$R(-\beta, \alpha, 0) . R(\psi_w, \theta_w, \varphi_w) = R(\psi, \theta, \varphi) \tag{12}$$

By setting sideslip angle zero, and comparing two sides of these equations which do not contain the variable, we write the equations in closed form;

$$\begin{aligned} \theta &= f_a(\psi_w, \theta_w, \psi, \alpha) \\ \varphi &= f_b(\psi_w, \theta_w, \psi, \alpha) \end{aligned} \tag{13}$$

After obtaining the body axes Euler angles almost all of the modal inputs for each mode can be obtained. Total velocity is a modal input for every mode is available

from path planner. For level flight and Climb/Descent mode, everything needed is already available on Table 2 (Climbing/Diving rate). In roll mode, only desired roll angle displacement (value of the integral associated with roll mode on Table 1) is needed, which can be obtained from Euler roll angle time history. Loop modes require the body Euler angle rates as modal input, which can be obtained from $\theta\,(t)$ and $\psi\,(t)$ for Longitudinal Loop and Lateral Loop respectively. Things are a bit more complicated in 3D mode, because it is required to extract the angular body rates from a given 3D trajectory and attitude data. Since all the body Euler angles are available, kinematical equation for Euler angles can be solved inversely to acquire the angular rates, but this is not convenient since equations have singular points and requires manipulating trigonometric equations. More elegant approach would be converting the Euler angles to Quaternions (shown by $b_i, i = 0, 1, 2, 3$) , and solve the algebraic, singularity free Quaternion kinematical equation to obtain body angular rates. Equation 14 gives the well known formula for converting Euler angles to Quaternions and Eq. 15 shows the kinematical Quaternion equation which has to be solved inversely in order to obtain the angular rates.

$$B = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \pm\left(C\frac{\phi}{2}C\frac{\theta}{2}C\frac{\psi}{2} + S\frac{\phi}{2}S\frac{\theta}{2}S\frac{\psi}{2}\right) \\ \pm\left(S\frac{\phi}{2}C\frac{\theta}{2}C\frac{\psi}{2} - C\frac{\phi}{2}S\frac{\theta}{2}S\frac{\psi}{2}\right) \\ \pm\left(C\frac{\phi}{2}S\frac{\theta}{2}C\frac{\psi}{2} + S\frac{\phi}{2}C\frac{\theta}{2}S\frac{\psi}{2}\right) \\ \pm\left(C\frac{\phi}{2}C\frac{\theta}{2}S\frac{\psi}{2} - S\frac{\phi}{2}S\frac{\theta}{2}C\frac{\psi}{2}\right) \end{bmatrix} \quad S : Sin,\ C : Cos \tag{14}$$

$$\dot{B} = \begin{bmatrix} \dot{b}_0 \\ \dot{b}_1 \\ \dot{b}_2 \\ \dot{b}_3 \end{bmatrix} = \begin{bmatrix} 0 & -P & -Q & R \\ P & 0 & R & -Q \\ Q & -R & 0 & P \\ R & Q & -P & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{15}$$

3.2.3 Satisfying the Sequential Constraints

Multi modal control system is based on the successful execution of one mode after another, but this doesn't mean that mode sequence is arbitrary. We can simply cast the mode compatibility condition as; final states of the first maneuver mode must intersect with the second modes initial conditions [32]. This is valid for kinematical variables, or to be more specific the attitude angles (since geometric path is given continuous from path planning layer). The only problem is associated with pitch angle and roll angle, (for example level flight ends with zero roll angle where as the coordinated turn starts with non-zero roll angle). If we neglect this issue, reference maneuýver profiles will have discontinuous attitude histories which could degrade the performance of low level controller and may even cause instability.

Therefore a transition mode (roll mode to change and longitudinal loop mode to change) must be inserted between incompatible modes. To speed up the process a mode transition table (Table 3) was prepared; showing which mode is compatible with each other, and which needs a transition mode.

Table 3 Mode transition table

δ_{ij}	q_0	q_1	q_3	q_4	q_5	q_6
q_0	1	θ^*	1	φ^*	θ^*, φ^*	1
q_1	θ^*	θ^*	θ^*	θ^*, φ^*	θ^*, φ^*	1
q_3	1	θ^*	1	θ^*, φ^*	θ^*, φ^*	1
q_4	φ^*	θ^*, φ^*	θ^*, φ^*	φ^*	θ^*, φ^*	1
q_5	θ^*, φ^*	θ^*, φ	θ^*, φ^*	θ^*, φ^*	θ^*, φ^*	1
q_6	1	0	0	0	0	0

in which 1 means that modes are always compatible and sequential feasibility check is not needed at all.

At this step, mode transitions which satisfy the table are neglected and the modes which require attitude tweaking (in either roll or pitch angle) is checked to be compatible. If they are not, additional translational modes (roll mode and longitudinal loop) are inserted between these modes to connect the attitudes of each mode. Since roll mode cannot be identified from trajectory data it only acts as a transition mode and it is not included in the table. Note that table is symmetrical (except for safety mode which is accessible from every mode but only allows transition to level flight).

This attitude changes are made quickly as possible to not to change shape of the flight trajectory (note that feasibility of modal inputs of transition modes are also checked via agility metric plots which corresponds to the loop in Fig. 8), if these transitions make dramatic changes on the flight trajectory, re-planning of the trajectory may be required to make sure that path avoids the obstacles, but this is a very rare case since most of the time environment is big enough to avoid obstacles during execution of transition modes.

3.2.4 Recovering the Feasible Modal Sequence

After checking the mode transition table, and finding the appropriate attitude changes for sequential feasibility, these transition modes are added to original mode sequence and final modal sequence is recovered. This modal sequence is feasible in the sense that it satisfies the envelope and sequential constraints.

Algorithm 4: Maneuver Generation

input : *Flight Trajectory* $(n_p(t), e_p(t), h(t))$
output: *Feasible Modal Sequence* $(q_i, \sigma_i, \tau_i), i = 1, 2..., N$
Solve the velocity variables from equation 6 **Discretize** the flight trajectory, with constant time step Δt to M waypoints **repeat**
  Label q_i according to $\left(\theta_{w_{j+1}} - \theta_{w_j}\right)$ from table 2.
until j=M
repeat
  switch *Mode Label* **do**
    CheckAgility Metric Grap, Recover α **Recover** Modal inputs σ_i through table 2 and equations 14 and 15 **Adjust**σ_i such that it is compatible withσ_{i-1}
until i=L
repeat
  Check Mode Transition Between q_{i-1} and q_i via table 3 **Insert** Transition modes between modes if neccessary **Recover** Modal inputs of transition modes via previous step
until i=L
GatherFeasible modal sequence by combining identified modes with transition modes

Table 4 Generated mode sequences

Mode	Label	Time intervals (sec)
q_0	Level flight	[0, 0.83]
q_3	Longitudinal loop (transition)	[0.83, 1]
q_1	Dive	[1, 3.57]
q_2	Roll mode (transition)	[3.57, 4.57]
q_4	Lateral loop	[4.57, 26.7]
q_2	Roll mode (transition)	[26.7, 27.7]
q_0	Level flight	[27.7, 30.3]

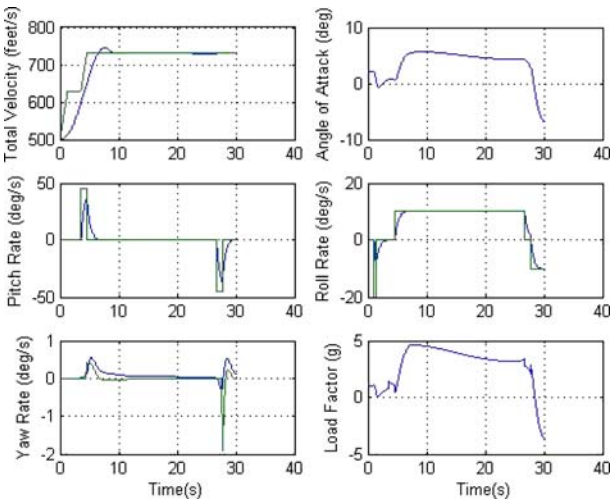

Fig. 10 State history and tracking performance

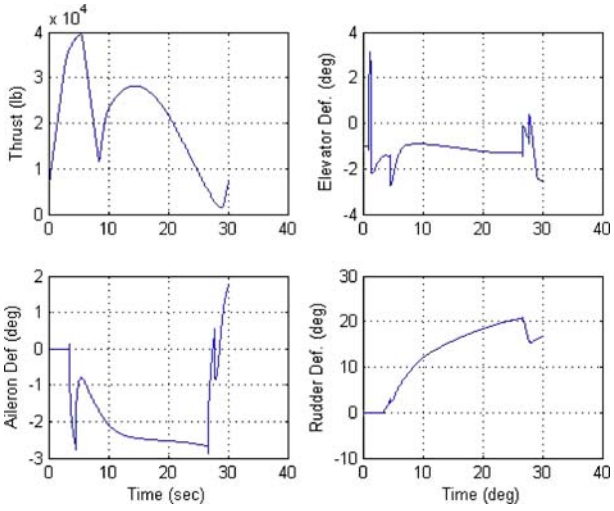

Fig. 11 Time history of true inputs of the aircraft

In the next section integration of path and maneuver planning algorithms for a complex environment is shown for an example mission.

4 Simulation Results

For simulation purposes we consider an example 200×200×200 unit cube complex city-like environment. In first part; path planning layer, constructs a 3D flight trajectory which avoids the obstacles while satisfying the velocity and acceleration constraints (by checking the flight envelope). This flight trajectory with velocity is given to maneuver planning layer. Example solution of the Path Planner in the 3D MelCity model environment is seen in Fig. 12.

Maneuver planning algorithm decomposes the path into flight modes in multi modal control framework and derives the feasible modal sequence based on the search of agility metric graphs and mode transition rules. Generated model sequences on the solution of the path planner is seen in Fig. 12 and timetable of the maneuver sequences is seen in Table 4. Three transition modes are placed between the other modes to obtain the attitude continuity.

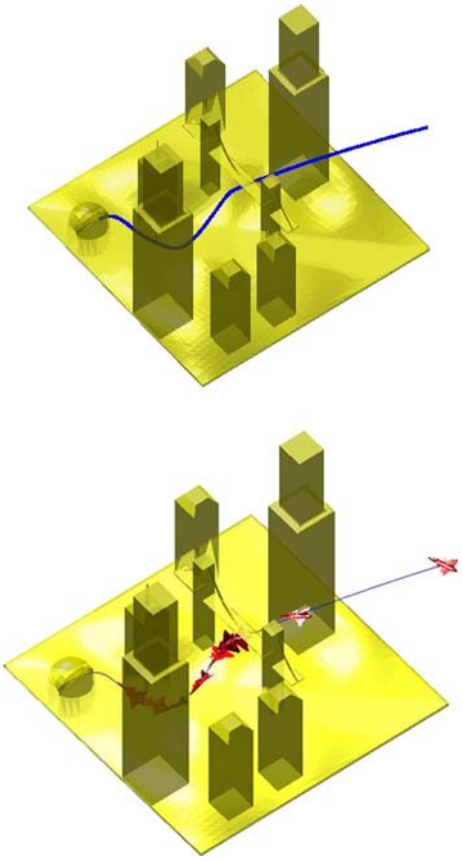

Fig. 12 Integration and solutions of the path planner and maneuver planner

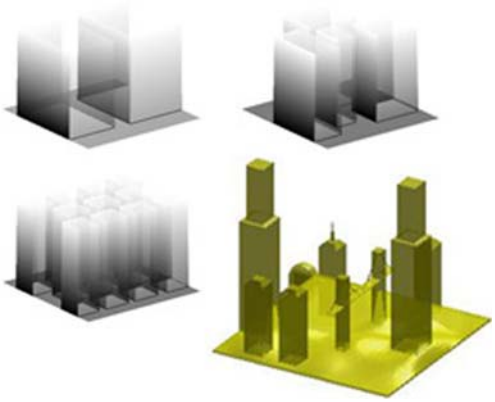

Fig. 13 3D Test environments for the performance test

For low level control purposes the switched Higher Order Sliding Mode Control system were used, [34], time histories of important states (with reference tracking performance) and control inputs are shown on Figs. 10 and 11. Judging the magnitude of state and control inputs we conclude that overall architecture has been successful in finding a feasible maneuver sequence (Figs. 12 and 13).

To get a better understanding of what we have gained by this integrated architecture,two extra simulations were done. In the first simulation, path planning step has no flight envelope feasibility check, which results in breakdown in maneuver planning section, because the algorithm fails to compute feasible modal inputs from agility metric graphs and velocity given by path planner is out of range, so no results could be obtained from this simulation. This shows that flight envelope check is a critical part of the integrated architecture.

In second simulation, no maneuver planning is used; trajectory from path planner is directly given to a trajectory tracking control system. Results of this simulation is shown on Fig. 14. This simulation results show that, attitude is unstable on flight path and there are deviations from the trajectory, which has two reasons. First, nonlinear

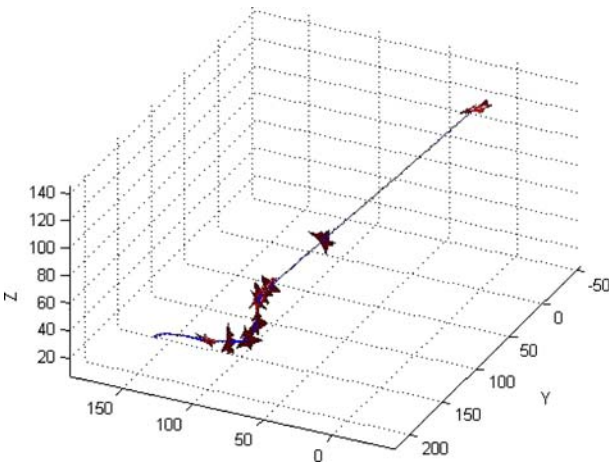

Fig. 14 Direct path planning trajectory tracking without maneuver planning

Table 5 Dynamically feasible path and maneuver sequences construction times (seconds)

		Trajectory planner			Maneuver planner	
		Connectivity path planner	B-Spline path planner	Total	Solution time	Number of mode sequence
Single-passage	avr	0.440	0.206	0.646	0.233	6
Problem	std	0.287	0.011	0.293	0.001	
City-like	avr	0.977	0.326	1.303	0.354	9
Environment	std	0.935	0.254	0.930	0.005	
Mostly-blocked	avr	3.930	2.182	5.837	0.421	13
Environment	std	2.504	3.347	3.912	0.006	
MelCity model	avr	3.306	0.538	3.844	0.322	6
Volume; 2^3x	std	1.528	0.650	1.212s	0.001	

controllers based on trajectory references are likely to become unstable on agile trajectories while multi modal control framework uses stable controllers for each mode [34]. Second reason is the absence of maneuver planning part; since there isn't any feasibility check on the agility metrics and attitude continuity, it is very reasonable to get unstable attitude and deviations from flight path due to saturated inputs, which have been supported by the simulation results.

Overall, these simulation results show that, when they are isolated path and maneuver planning layers have critical defects. They must be integrated together to get a feasible and controllable flight path.

We tested the performance of our method on some environments in varying ratio of obstacle-space. The computational times of the all phases of the algorithm are illustrated in Table 5 for 3D single-narrow-passage problem, city-like environment, mostly-blocked environment and MelCity model environment that has volume 2^3 times greater than the others as all seen in Fig. 13. All the experiments were conducted on a 3.00 GHz Intel Pentium(R) 4 processor with 2 Gb memory and the average results are obtained over 50 runs.

On side of the path planner, increasing complexity of the environment, as shown in Table 5, mainly increases computational time of the *connectivity path* that is implemented with a simplified version of RRT. Since repairing part of the algorithm is visited much more in planning complex environments, computational time of the B-Spline based planner phase is also rises. However, this rising rate does not grow exponentially and computational times mostly based on Finding Connectivity Path phase. On side of the maneuver planning, since length of the generated path by the path planner increases when the complexity of the environment is increased, computational time of the maneuver planner phase slightly rises according to environment complexity. The complete solution times suggest that our method will be applicable for real-time implementations as the solution time is favorably comparable to implementation times.

5 Conclusion

Trajectory design of an air vehicle in dense and complex environments, while pushing the limits of the vehicle to full performance is a challenging problem in two facets. The first facet is the control system design over the full flight envelope and the

second is the trajectory planning utilizing the full performance of the aircraft. In this work, we try to address the mostly second facet via the generating dynamically feasible trajectory planning and refining of the flight trajectory using the flight modes from which almost any aggressive maneuver can be decomposed. Hence, a real-time implementable two layer planner strategy is implemented for obtaining 3D flight-path generation for an Unmanned Combat Aerial Vehicles in 3D Complex environments. Integrated path planning and maneuver generation system enabled each layer to solve its own reduced order dimensional feasibility problem, thus simplified the problem and improved the real time implement ability.

In Trajectory Planning layer, to solve the motion planning problem of an unmanned combat aerial vehicles, we suggested a two step planner. Initially, simplified version of the RRT planner is used for rapidly exploring the environment with an approximate line segments. The resulting connecting path is converted into flight way points through a line-of-sight segmentation. In the second step, remaining way points are connected with cubic B-Spline curves and these curves are repaired probabilistically to obtain a geometrically (prevents collisions) and dynamically feasible (considers velocity and acceleration constraints) path. In the maneuver planning layer, the flight trajectory are decomposed to sequences of maneuver modes and associated parameters (considers saturation envelope and attitude continuity constraints). Maneuver generation algorithm derives feasible maneuver parameters for each mode and overall sequence by using of mode transition rules and agility metric graphs. Resulting integrated system is tested on simulations for 3D complex environments and it gave satisfactory results to used for real time implementation for UCAVs operating in challenging urban environments.

One of the venues considered for future work is maneuvering in extreme narrow passages in which the aircraft has to roll or tilt to pass through the very narrow passages. In the problems we have examined distance between obstacles are far wider compared to wing span of the aircraft, so we did not include this case. Another venue for future work includes expanding the maneuver modes by adding un-coordinated turns (non-zero sideslip angle). In a framework sense, these extensions will require tighter integration of the path and the maneuver planning layers in the case of uncertain environments. Moreover, extension of the algorithms presented to UAV fleets is another natural application of this work.

References

1. Bayazit, O.B., Xie, D., Amato, N.M.: Iterative relaxation of constraints: a framework for improving automated motion planning. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp. 3433–3440 (2005)
2. Bohlin, R., Kavraki, L.E.: A randomized algorithm for robot path planning based on lazy evaluation. Handbook on Randomized Computing, pp. 221–249. Kluwer, Dordrecht (2001)
3. Boor, V., Overmars, M.H., van der Stappen, A.F.: The Gaussian sampling strategy for probabilistic roadmap planners. IEEE Int. Conf. Robot. Autom. 6 (1999)
4. Brockett, R.W.: Languages for motion description and map making. Proc. Symp. Appl. Math. **14**, 181–293 (1990)
5. Sastry, S., Tomlin, C., Pappas, G.J.: Conflict resolution for air traffic management: a study in multi-agent hybrid systems. IEEE Trans. Automat. Contr. **43** (1998)
6. Dever, C., Mettlera, B., Feron, E., Popovic, J., McConley, M.: Nonlinear trajectory generation for autonomous vehicles via parameterized maneuver classes. J. Guid. Control Dyn. **29**, 289–302 (2006)

7. Dyllong, E., Visioli, A.: Planning and real-time modifications of a trajectory using spline techniques. Robotica, **21**(5), 475–482 (2003)
8. Murphy, P.C., et al.: Fighter agility metrics. Candidate Control Design Metrics for an Agile Fighter (1991)
9. Nguyen, L.T., et al.: Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability. NASA Technical Paper 1538 (1979)
10. Fainekos, G., Gazit, H.K., Pappas, G.J.: Hybrid controllers for path planning: a temporal logic approach. In: IEEE Conference on Decision and Control (2005)
11. Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. AIAA J. Guid. Control **25**(1), 116–129 (2002)
12. Ghosh, R., Tomlin, C.: Nonlinear inverse dynamic control for mode-based flight. In: Proceedings of AIAA Guidance, Navigation and Control Conference and Exhibit (2000)
13. Hsu, D.: Randomized single-query motion planning in expansive spaces, p. 134. PhD Thesis (2000)
14. Hsu, D., Jiang, T., Reif, J., Sun, Z.: The bridge test for sampling narrow passages with probabilistic roadmap planners. In: IEEE International Conference on Robotics & Automation (2003)
15. Hsu, D., Kavraki, L.E., Latombe, J.-C., Motwani, R., Sorkin, S.: On finding narrow passages with probabilistic roadmap planners. In: International Workshop on Algorithmic Foundations of Robotics, pp. 141–153 (1998)
16. Hsu, D., Kindel, R., Latombe, J.-C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. Int. J. Rob. Res. **21**(2), 233–255 (2002)
17. Hsu, D., Latombe, J.-C., Motwani, R.: Path planning in expansive configuration spaces. Int. J. Comput. Geom. Appl. **4**, 495–512 (1999)
18. Kavraki, L., Svestka, P., Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
19. Kindel, R., Hsu, D., Robert, J.C., Latombe, S.: Randomized kinodynamic motion planning with moving obstacles. Int. J. Rob. Res. **21**(3), 233–255 (2000)
20. Komoriya, K., Tanie, K.: Trajectory design and control of a wheel-type mobile robot using b-spline curve. In: IEEE/RSJ International Workshop on Intelligent Robots and Systems '89. The Autonomous Mobile Robots and its Applications. IROS '89. Proceedings, pp. 398–405 (1989)
21. Koyuncu, E., Inalhan, G.: A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems 2008. IROS 2008, pp. 815–821 (2008)
22. Koyuncu, E., Ure, N.K., Inalhan, G.: A probabilistic algorithm for mode based motion planning of agile unmanned air vehicles in complex environments. Int. Federation of Automatic Control (IFAC'08) World Congress (2008)
23. LaValle, S., Kuffner, J.: Randomized kinodynamic planning. In: 1999 IEEE International Conference on Robotics and Automation. Proceedings, vol. 1, pp. 473–479 (1999)
24. Liefer, R.K.: Fighter agility metrics. NASA Technical Paper Report No: AD-A22447 (1990)
25. Munoz, V., Ollero, A., Prado, M., Simon, A.: Mobile robot trajectory planning with dynamic and kinematic constraints. In: 1994 IEEE International Conference on Robotics and Automation. Proceedings, vol. 4, pp. 2802–2807 (1994)
26. Nikolos, I.K., Valavanis, K.P., Tsourveloudis, N.C., Kostaras, A.N.: Evolutionary algorithm based offline/online path planner for uav navigation. IEEE Trans. Syst. Man Cybern., Part B **33**(6), 898–912 (2003)
27. Oishi, M., Tomlin, C.: Nonlinear control of a vstol aircraft. In: The Proceedings of the 38th IEEE Conference on Decision and Control (1999)
28. Paulos, E.: On-line collision avoidance for multiple robots using b-splines. University of California Berkeley Computer Science Division (EECS) Technical Report, (Report No. UCB//CSD-98-977) (1998)
29. Piegl, L.A., Tiller, W.: The NURBS Book. Springer, New York (1997)
30. Song, G., Amato, N.: Randomized motion planning for car-like robots with c-prm. In: 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Proceedings, vol. 1, pp. 37–42 (2001)
31. Stevens, B.L., Lewis, F.L.: Aircraft Simulation and Control. Wiley, New York (2002)
32. Pappas, G.J., Koo, T.J., Sastry, S.: Modal control of systems with constraints. In: Proceedings of the 40th IEEE Conference Decision and Control, pp. 2075–2080 (2001)
33. Feron, E., Schouwenaars, T., How, J.: Receding horizon path planning with implicit safety guarantees. In: American Control Conference (2004)

34. Ure, N.K., Inalhan, G.: Design of higher order sliding mode control laws for multi modal agile maneuvering ucavs. In: 2nd Int. Symposium on Systems and Controls in Aerospace (2008)
35. Ure, N.K., Inalhan, G.: Design of a multi modal control framework for agile maneuvering ucavs. In: IEEE Aerospace Conference (2009)
36. Krishnaprasad, P.S., Manikonda, V., Hendler, J.: Languages, behaviors, hybrid architectures and motion control. Mathematical Control Theory (1998)
37. Valasek, J., Downing, D.R.: An investigation of fighter aircraft agility. NASA Technical Paper 588 (1993)
38. Vazquez, G.B., Sossa, A.H., Diaz de Leon, S.J.L.: Auto guided vehicle control using expanded time b-splines. In: IEEE International Conference on Systems, Man, and Cybernetics, Humans, Information and Technology, vol. 3, pp. 2786–2791 (1994)