Dynamically Feasible Probabilistic Motion Planning in Complex Environments for UAVs

Emre Koyuncu
Istanbul Technical University, Controls and Avionics Laboratory
Turkey

Gokhan Inalhan
Istanbul Technical University, Aeronautical and Astronautical Faculty
Turkey

Abstract
In this work, we consider the problem of generating practically implementable path plan for flying unmanned 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 step feasible trajectory generating approach. 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.
After this first step we explain two different methods to create Dynamically Feasible Path, first one that we called Modal-Maneuver Based PRM Planner is suitable for agile unmanned aerial vehicles that their maneuvers can be define with distinct modes. This allows significant decreases in control input space and thus search dimensions, resulting in a natural way to design controllers and implement trajectory planning using the closed-form flight modes. In this approach the resulting connectivity path and the corresponding milestones are refined with a single query Probabilistic Road Map (PRM) implementation that creates dynamically feasible flight paths with distinct flight mode selections and their modal control inputs. In our second approach that we called Probabilistic B-Spline Planner, 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 scale factor considering the velocity and acceleration limits of the aircrafts.

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.
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. If the flight trajectory is generated without checking the velocity and acceleration bounds, lower level layers 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 lower level layers, these layers covers their disadvantages and provides both dynamically and geometrically feasible flight trajectories for complex environments. 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 (Frazzoli et al., 2002). In practice, kinodynamic planners are implementable only for systems that have small state-space dimensions. For example, the work presented in (Frazzoli et al., 2002) 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 .
In our approach, 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, we will explain two different methods to create Dynamically Feasible Path. First one that we called Modal-Maneuver Based PRM Planner is developed for agile unmanned aerial vehicles that their maneuvers can be define with distinct modes. This allows significant decreases in control input space and thus search dimensions, resulting in a natural way to design controllers and implement trajectory planning using the closed-form flight modes. In this approach the resulting connectivity path and the corresponding milestones are refined with a single query Probabilistic Road Map (PRM) implementation that creates dynamically feasible flight paths with distinct flight mode selections and their modal control inputs. In our second approach, Probabilistic B-Spline Path Planner, every consecutive way points are connected with C^2 continuous B-Spline curve. 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 considering the velocity and acceleration limits of the aircrafts.
Rest of this chapter is organized as follows. In Section 2, framework of the dynamically feasible path planning and literature survey is given. In section 3, first step, finding geometrically

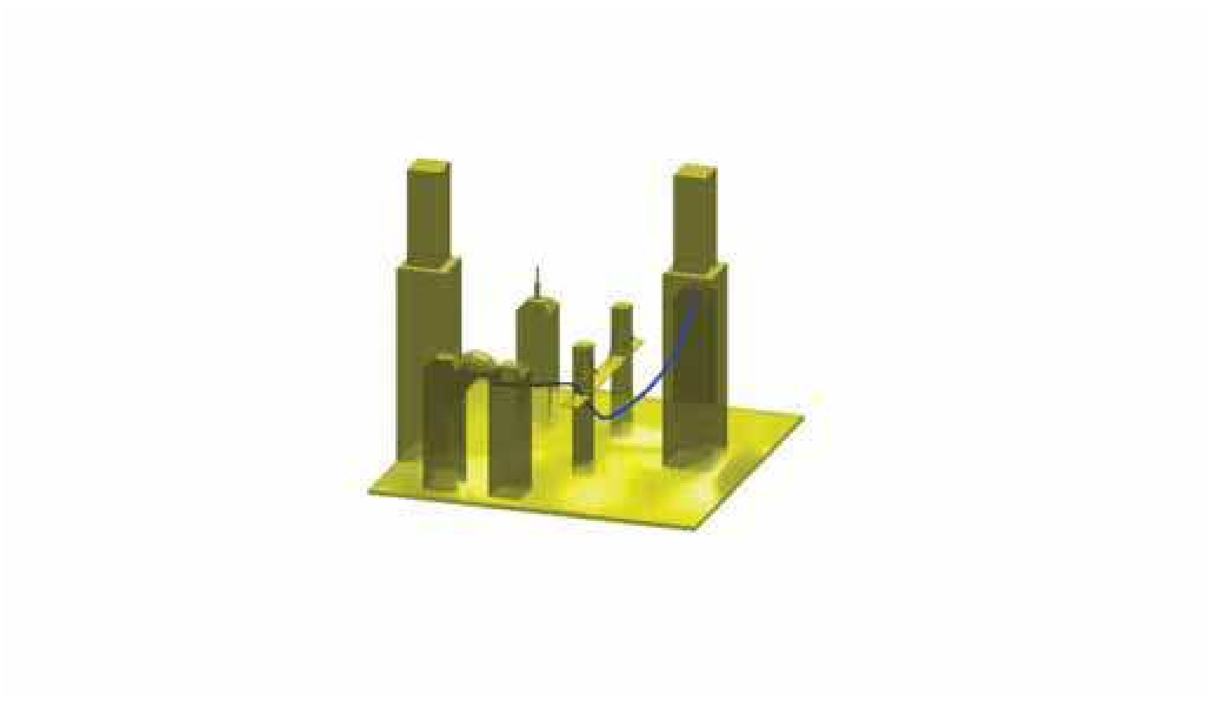


Fig. 1. UCAV flight demonstration in the 3D complex city-like environment and landing to its base

connectivity path method is explained and in Section 4, details of the following step that two generating dynamically feasible trajectory methods is explained. In Section 5, system architecture of the ITU CAL mobile robotic testbed is presented and its experiments are demonstrated in Section 6. Simulation results and its computational time tables are also given in Section 6. The conclusions are discussed in Section 7.

2. Framework of the Dynamically Feasible Path Planning Algorithms

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 (Kavraki et al., 1996), 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 (Hsu et al., 2003) and (Boor et al., 1999). 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 (Hsu et al., 1999), (Hsu, 2000), and (LaValle & Kuffner, 1999). 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 nonholonomic and/or dynamic constraints . The main philosophy behind kinodynamic planning is searching a higher dimensional state space that captures the dynamics of the system (LaValle & Kuffner, 1999), (Hsu et al., 2002).

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 (Hsu et al., 1998), 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 (Bayazit et al., 2005), 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 (Bohlin & Kavraki, 2001) and C-PRM (Song & Amato, 2001).

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. Dynamic constraints of the vehicle 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 guider-milestones that point out hard regions and directions of the next coming hard regions in the environment.

In explained first method, Modal-Maneuvering PRM planner, milestones are created for each flight segment using randomized flight mode selection. This exploits all the full flight-envelope and capability of the vehicle, and once the planner samples enough number of milestones near one of the way-points, it continues with these milestones to reach other way points. One distinct feature of this planner in comparison with other probabilistic planning methods is the reduced input space selection. Specifically, in each query, instead of choosing all input variables randomly, our planner chooses maneuver mode and its parameters that are constrained by vehicle dynamics. In addition, the size of the search is limited to only partial flight segments. Both of these factors contribute significantly in reduced computation time.

In explained second strategy that we will call Probabilistic B-Spline Path Planner, every way point is connected with a B-Spline curve, then collisions 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 methods are achieved by randomized waypoint expansion on the connecting line path and the unit flight time is expanded within controllable regime considering to feasible velocity and acceleration interval. Since B-Spline curves have local support property, these repairing processes can be made on local path segments of interest.

B-Spline curves have been used in many dynamic path planning and control problem implementations. In (Komoriya & Tanie, 1989) , dynamic trajectory is generated with the minimum travel time for two-wheeled-driven type mobile robot. In (Munoz et al., 1994) 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 (Paulos, 1998) and robot manipulators in (Dyllong & Visioli, 2003). Constant acceleration time-scalable path generation method for the unmanned helicopters flying in the urban environments is used in our earlier work in (Koyuncu & Inalhan, 2008) that we will use similar method but this time for the unmanned combat aerial vehicles.

3. Finding Geometrically Connectivity Path: First Step

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, 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 (LaValle & Kuffner, 1999) approach that converges to goal configuration rapidly. We tested performance of the algorithm in different complex environments and 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 good property of the RRT algorithm 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 I, 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 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

Algorithm 1: Goal Biased RRT Algorithm

```
    input  : initial configuration $q_{init}$ and goal configuration $q_{goal}$
    output: connectivity path
1   $\tau \leftarrow q_{init}$ and $i \leftarrow 1$
2   repeat
3       Select random point $m_{rand}$ in $C$ and its neighbor point $m_{near}$ in $\tau$
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           $\tau \leftarrow m_{new}$ 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 $\tau$
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          $\tau \leftarrow m_{new}$ 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 $\tau$
```

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 (Kindel et al., 2000). A solution of the algorithm in a complex city-like environment is illustrated in Fig. 2.
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.3, 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.3.

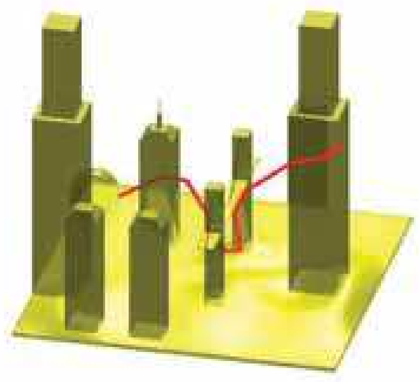


Fig. 2. Demonstration of the RRT based Finding Connectivity Path Algorithm

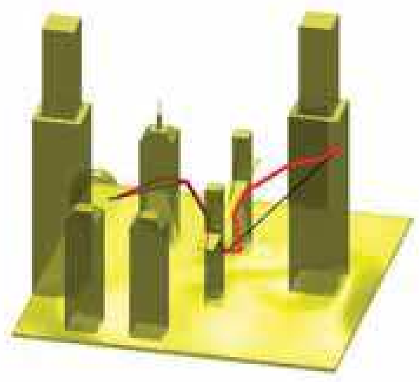


Fig. 3. Demonstration of the refining Connectivity Path with the Line-of-Sight Filter Algorithm

4. Generating Dynamically Feasible Trajectory: Second Step

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 agile and continu-

Algorithm 2: Line-of-Sight Filtering

```
  input  : connectivity path
  output: way point set WP
1 $m_{visib} \leftarrow m_1$ and $m_i \leftarrow m_2$
2 repeat
3     Generate line $\ell_{visib}$ from $m_{visib}$ to $m_i$
4     if $\ell_{visib}$ is collide with $C_{obst}$ then
5         $WP \leftarrow m_{i-1}$
6     else
7         $i + 1$
8 until last point of connectivity path is reached
```

ous motion plan - a desirable feature in many complex unmanned 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.
On this point, we suggest two methods to create *Dynamically Feasible Path*, first one that we called *Mode Based PRM Planner* is suitable for agile unmanned vehicles that their maneuvers can be define with distinct modes.This mode-based structure is also well suited designing a systematic hybrid flight control system. Our second path planner method is called *Probabilistic B-Spline Planner*. This algorithm generates continuous flight paths using B-Spline fitting strategy. In our earlier work, we used this strategy to generate constant acceleration time-scalable path for the unmanned helicopters (have hovering and instantaneously changing heading angle etc. abilities) flying in the urban environments in (Koyuncu & Inalhan, 2008). We will explain the similar method that generates C^2 continuous flight path but this time it will also be suitable for the agile aircrafts (unmanned combat aerial vehicles).

4.1 Modal-Maneuver Based Probabilistic Road Mapping Method

Our Mode-Based PRM approach (Koyuncu et al., 2008) is based on the simple idea of exploiting the full flight envelope of the air vehicle through distinct flight modes from which almost any maneuver can be created. This mode-based structure is especially well suited both creating flight paths and also designing a systematic flight control system. However, this structure does not necessarily solve the critical problem of finding a) the possible fly-through passages and b) the necessary mode selections to utilize these passages. These two points and the corresponding solution method are the main focus of this work.
One distinct feature of this planner in comparison with other probabilistic planning methods is the reduced input space selection. Specifically, in each query, instead of choosing all input variables randomly, our planner chooses maneuver mode and its parameters that are constrained by vehicle dynamics. In addition, the size of the search is limited to only partial flight segments. Both of these factors contribute significantly in reduced computation time. It is noted by (Frazzoli et al., 2002) that, in 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. Because of this consideration, in practice kinodynamic planners are implementable only for systems that have small state-space dimensions. Thus, for the air vehicles that we focus on, it is hard and time-consuming

to obtain a feasible path using a standard kinodynamic planner. We address this problem by directing the search not to the expensive state-space, but to only a subset of the input space as required by the flight modes (and their resulting controlled state-space selections). Thus, for almost every flight mode, the input space is either two or three dimensional, with the most complex mode 3D spin, being four dimensional.

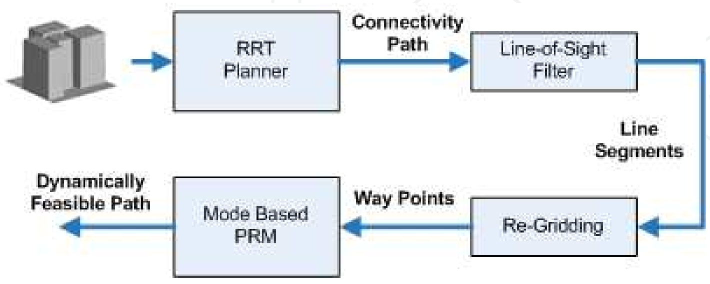


Fig. 4. Dynamically Feasible Path Generating Process with Mode Based PRM Planner

Motion planning of agile vehicles from a control perspective has been mainly studied under the topic of hierarchical hybrid systems. Basically, the flight path of an aircraft can be divided into modes, and these modes serve as reference blocks to a control system, and the control system regulates these modes with given specifications by possibly using nonlinear control laws (Ghosh & Tomlin, 2000). Note that this modal approach can also be used for trajectory optimization for multiple vehicles (Inalhan et al., 2002). (Frazzoli et al., 2002) suggested a path-planning system which defines a class of maneuvers from a finite state machine, and uses a trajectory based controller to regulate the agile vehicle dynamics into these feasible trajectories. This approach has got the advantage of generating both feasible and optimal trajectories in an environment with an obstacle while ensuring robust tracking of these trajectories. However, the trajectories to be controlled are limited to the trajectories generated by the finite state machine. Similar approaches have also been developed in (Schouwenaars et al., 2004). Although these works provide asymptotic tracking, the modes as governed by the state-machines do not exploit the full flight envelope and the generated maneuvers are not really tailored toward an environment, which demand tactical advantage through exploitation of the vehicle's full capability - a feature that exists in sample based motion planning algorithms and we exploit this feature while creating dynamically feasible paths using the probabilistic roadmap approach over flight modes. We illustrate the control of a complex agile maneuver over such modes in Fig. 5.
Multi Modal control framework 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 work we shall only focus on side of the path planning, discussion of the switched control layer can be found on (Ure & Inalhan, 2009) and (Ure & Inalhan, 2008)
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

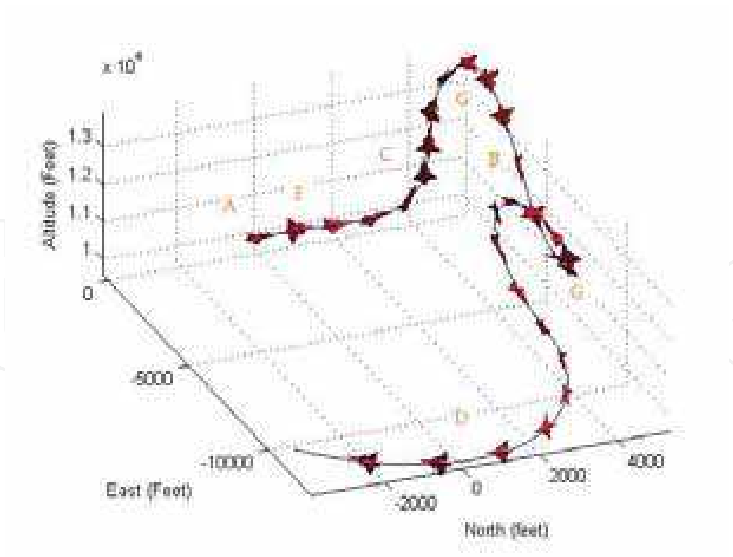


Fig. 5. Flight Modes of an Aggressive Maneuver of F-16 over Flight Using the Full-Envelope Dynamic Model

not independent during all maneuvers and one does not need to give all the state trajectory of the aircraft to define a maneuver. Demonstration of those modes table (Table 1) is seen in below and these modes and their generations are explained in (Ure & Inalhan, 2008) and one of applications is demonstrated in (Koyuncu et al., 2009). These same modes table approach will be used during design of the algorithms.

	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	$(\dot{\varphi}, \dot{\theta}, \dot{\psi}) = 0$	$V_T, (\dot{h}, \theta_w)$
q_2	Roll	$(\dot{\theta}, \dot{\psi}) = 0$	$V_T, \int P_w dt$
q_3	Longitudinal placeLoop	$(\dot{\varphi}, \dot{\psi}) = 0$	$\left(V_T, r_{loop}\right), \dot{\theta}$
q_4	Lateral placeLoop	$\dot{h} = 0, (\dot{\varphi}, \dot{\theta}) = 0$	$\left(V_T, r_{loop}\right), \dot{\psi}$
q_5	3D Mode	$\{\}$	$V_T, P, Q, R/$ $V_T, \varphi_w, \theta_w, \psi_w$
q_6	Safety	$\{\}$	$\{0,1\}$

Table 1. Flight Modes and Modal Inputs

Previous step provided us to obtain a flight path with *way-points* that generally appear as long straight flight segments that occasionally enter and exit passages. As a result of the underlying randomized exploration toward the goal region, the tendency of the path is toward larger passages and toward direct paths that lead to the goal region. Although this provides a reasonably good approximation, it does not necessarily correspond to a flyable trajectory as it is based on a simple point mass model with velocity and heading.
This part of our planning strategy is an extension of single-query PRM algorithm. It iteratively builds a tree-shaped roadmap to connect the way-points one-by-one. In every inner loop, it

Algorithm 3: Mode-Based PRM Planner

```
    input  : way point vector
    output: dynamically feasible path
 1  repeat
 2      Tree ← reached points
 3      goal points ← other way points
 4      repeat
 5          Select a milestone $m_{rand}$ from Tree probabilistically
 6          Select a maneuver mode uniformly at random
 7          Select modal inputs according to selected milestone
 8          Create a trajectory segment $e_{new}$ with selected modal inputs and maneuver mode
 9          Extend $m_{rand}$ with $e_{new}$ to $m_{new}$
10          if $e_{new}$ is in $C_{free}$ then
11              if $m_{new}$ is in approaching field of any goal points then
12                  Compute weight value $w$
13                  reached points ← $(m_{new}, w)$
14                  if size(reached points) is enough then
15                      exit with success
16              else
17                  Tree ← $m_{new}$ and $i + 1$
18          if $i = N$ max iteration number then
19              exit with failure and $f + 1$
20      until success or failure
21      if $f = M$ max failure number then
22          Erase prior path segment and go back prior interest region to recalculate
23      else
24          $f \leftarrow 0$
25  until last way point is reached
```

first selects at random a milestone as in Line 5, maneuver-mode and its modal inputs from Table 1 as seen in Line 6 and 7 and then generates a trajectory with selected maneuver mode and modal inputs from selected milestone as shown in Line 8. If this trajectory does not collide with any obstacle (checked in Line 10), its end point is added to tree as a new milestone as shown in Line 17 and its modal inputs are stored. If newly generated milestones fall in nearby region of any goal points, the planner assigns a weight value to these milestones according to their approaching angles as depicted in Line 12.

Milestone Selection; The planner selects an existing milestone in the Tree at random according to direct proportion to their values. A milestone which has higher weight value has a greater chance of being selected by planner, in the other words; milestones which can be propagated with more smooth trajectories have greater chance to continue. These weight values are assigned by *approaching field*. This milestone selection technique pushes our planners to side of kinodynamic planners that use single-query PRM method. Differently, RRT based kinodynamic planners select existing milestones which are nearest (metrically) to randomly-selected states (within all space). PRM like kinodynamic planners can select a milestone in tree according to their values that are charged by planners before. Hence, planner can make decision about which milestones should be chosen more densely. Therefore, our planner uses alike method to select milestones.

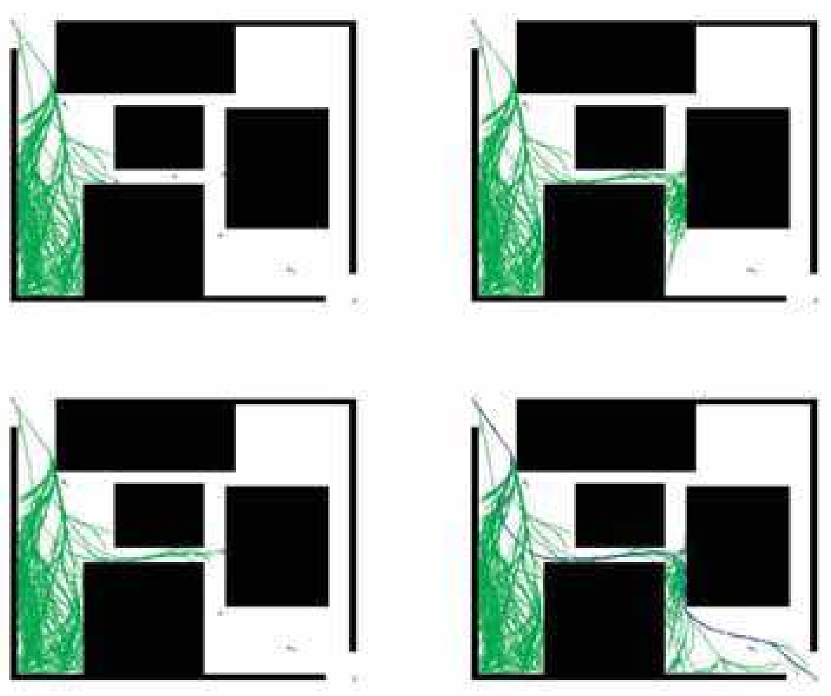


Fig. 6. 2D Demonstration of Mode-Based PRM Searching

Modal-Input selection; Our planner only chooses modal inputs of the distinct maneuver modes instead of choosing control inputs from high dimensional control-input space. These distinct modes can exploit the full flight envelope and almost every flight paths can be created with their combinations. In every loop of algorithm, after the milestone selection, planner first chooses flight maneuver-mode and then chooses its modal inputs according to weight value of the selected milestone. For example, considering to selecting level-flight mode, the milestones which have close angles with line-of-sight to next-coming goal in a small interval (therefore, it is assigned with higher weight values by *approaching field*) can be propagated with longer straight flight paths. Hence, if this milestone is selected by planner, higher velocity rates (in constant time, longer distance rates) are mostly preferred as modal input and then more agility is obtained.

Computing Weight Values; If new generated collision-free milestone falls in nearby of any way points in distinct distance metric, according to its angles and distance, it is assigned with the specific weight value. For deciding these values, every way points are enclosed with *approaching fields* includes distinct regions. If the angles of the milestone (felt in the region) are within specific rate interval of the region, it is enumerated with the respective weight value. Thus, it is aimed that; to charge the milestones which have angles closer to angle rates that can carry it easily (i.e. with smoother curve) to next-coming way point with higher values. The planner more densely selects the milestones are charged with higher value. Hence, it is

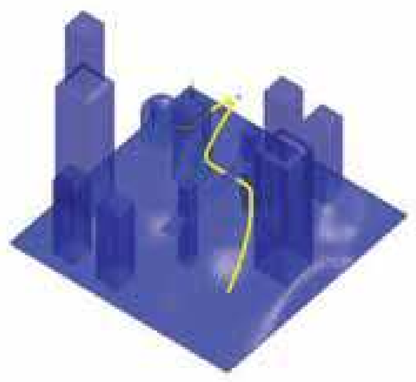


Fig. 7. Complete Solution of Mode Based PRM Planner

intended to create more smooth flight segments. This tunneling effect provides a straightforward solution to the classical narrow passage effect seen in standard PRM methods.
In the main loop, the inner PRM path segment loops try to connect the way-points one-by-one until the ultimate goal region is attained. During this iteration, if the inner loop returns an increased number of failures, prior path segment is erased as depicted in Line 22 and PRM searching is run from prior interest region. Snap-shots from the evolving PRM iterations and the completed solution are given in Fig. 6.

4.2 Probabilistic B-Spline Planning Method

In our second way, we still desire that generated path must be continuous on the way points. During the path generation phase, 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 (Piegl & Tiller, 1997) .
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;

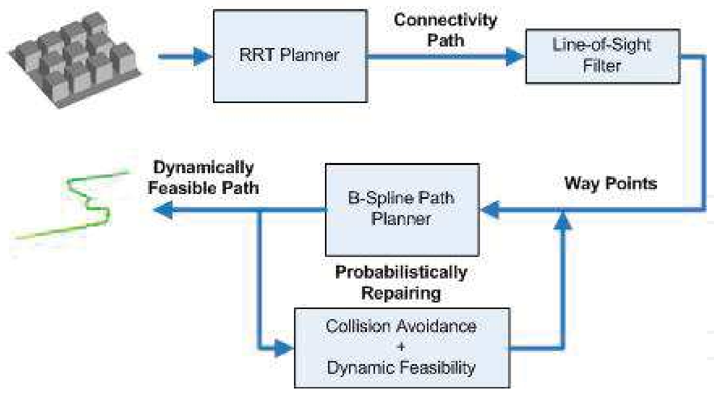

RRT Planner
Connectivity Path
Line-of-Sight Filter
Way Points
B-Spline Path Planner
Dynamically Feasible Path
Probabilistically Repairing
Collision Avoidance + Dynamic Feasibility

Fig. 8. Dynamically Feasible Trajectory generating using Probabilistic B-Spline Planner

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

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

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 $U = \left[\begin{array}{ccc} u_0 & \ldots & u_{max} \end{array}\right]$ is called the knot vector. Frequently, the knot points are referred to as the break points on curve (Piegl & Tiller, 1997). B-Spline basis function $N_{i,k}$ is zero outside the interval $[u_i, u_{i+k}]$ 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 (Nikolos et al., 2003).
In this strategy, we choose generate forth-order path B-Spline (cubic polynomials) to obtain continuous inertial velocity and acceleration. All dynamically feasible trajectory planning process illustrated in Fig. 8. In our earlier work (Koyuncu & Inalhan, 2008), we used constant acceleration time-scalable path generation method for the unmanned helicopters(have hovering and instantaneously change-heading abilities) flying in the urban environments that we used to generate third-order segmented (between the way point) B-Spline curves. It was good approximation to obtain rapid path planner algorithm that generates continuous inertial velocity and constant acceleration. In this approach, we will generate single cubic B-Spline curve. Complete process of that method seen in Fig. 8.
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 $\mathbf{U} = [\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & U_{mid} & 1 & 1 & 1 & 1 \end{array}]$ 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 (Piegl & Tiller, 1997) 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 (Vazquez et al., 1994). 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 $\mathbf{P}$ 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 $\mathbf{gp}$ is also added in *control point* set $\mathbf{P}$ 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, lower layers(maneuver planners, low-level control layers etc.) would not be able to find feasible references from this generated trajectory. This concept is used for giving a sense on *Dynamic Feasibility* as Path Planners do.

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, local support property of the B-Splines allows local control over the generated spline. Specifically, this control is over the curve segments with $\pm k/2$ polygon spans around the

Algorithm 4: 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  P ← 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  P ← gp = $[gp_{1,f}\ gp_{2,b}\ gp_{2,f} \cdots gp_{m-1,b}\ gp_{m-1,f}\ gp_{m,b}]$ as control point set
7  U ← $[0\ 0\ 0\ 0\ \mathbf{U_{mid}}\ 1\ 1\ 1\ 1]$
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                 P ← $P_{new}$ 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-SplineTrajectory
```

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

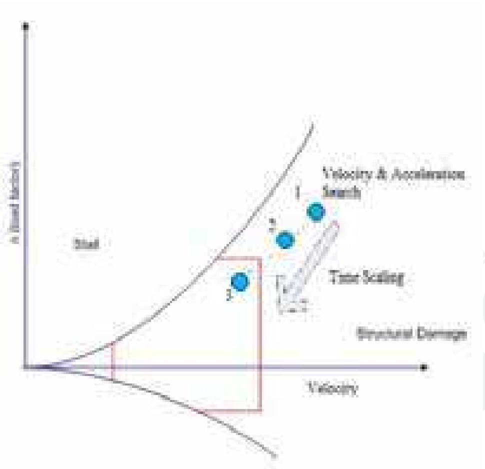


Fig. 9. Feasible Velocity - Acceleration Search on Flight Envelope

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 $\mathbf{P}$ 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 $\mathbf{U}$ 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. 9, 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 (Koyuncu et al., 2008). Note that other than flight envelope check there is not any dynamic model involved in path planning layer. All the other feasibility problems (actuator saturation, attitude discontinuity etc.) are left to low-level layers.

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 of the B-Spline Planner Algorithm in the *MelCity* model for a UCAV is demonstrated in Fig. 10

In (Koyuncu et al., 2009), one of application of this approach is demonstrated. In this work (Koyuncu et al., 2009), with adding one more interval maneuver planning layer, generated dynamically feasible trajectory is refined with modal-maneuvers that are briefly explained in previous section. This additional step gave a change to use advantages of the multi modal maneuvering approach.

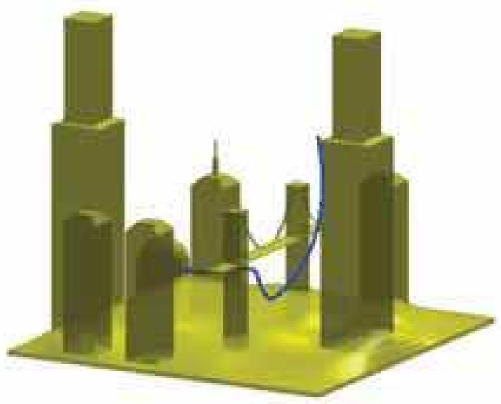


Fig. 10. Dynamically Feasible Path solution of the B-Spline Planner Algorithm in the complex 3D environment for UCAV

5. Mobil Robot Testbed for Algorithm Verification Tests

For 2D workspace verification tests of the algorithms, we set up a mobile robotic testbed in Controls and Avionics Lab (CAL) at the Istanbul Technical University. On this testbed, we implemented simplified 2D versions of the our algorithms to observe the practical implementability. 2D verification is good implementation method for such that 3D algorithms since its have rapid prototyping property. Similar approach is seen in (Clark et al., 2003), that such 3D Multi-robot space system tests have been implemented in 2D MARS test-platform.

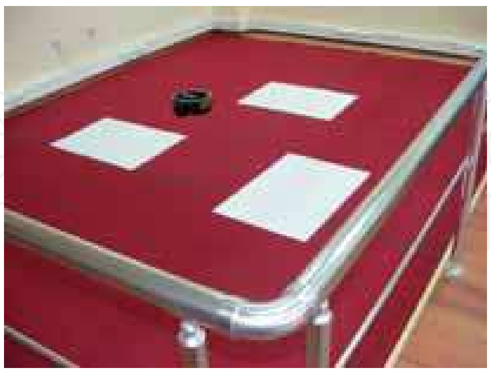


Fig. 11. ITUCAL Robotic Testbed

5.1 Mobile Robot Platform

The robotic testbed platform consists of a 4m x 3m movement platform with autonomous mobile robots. The testbed robots build in CAL are circular shaped and have two independently driven wheels which allows the robots to rotate on its center points. The robots are equipped with a 400 MHz Linux computer (Gumstix) and 8-bit microcontroller (Robostix) to perform own tracking controllers and communication.

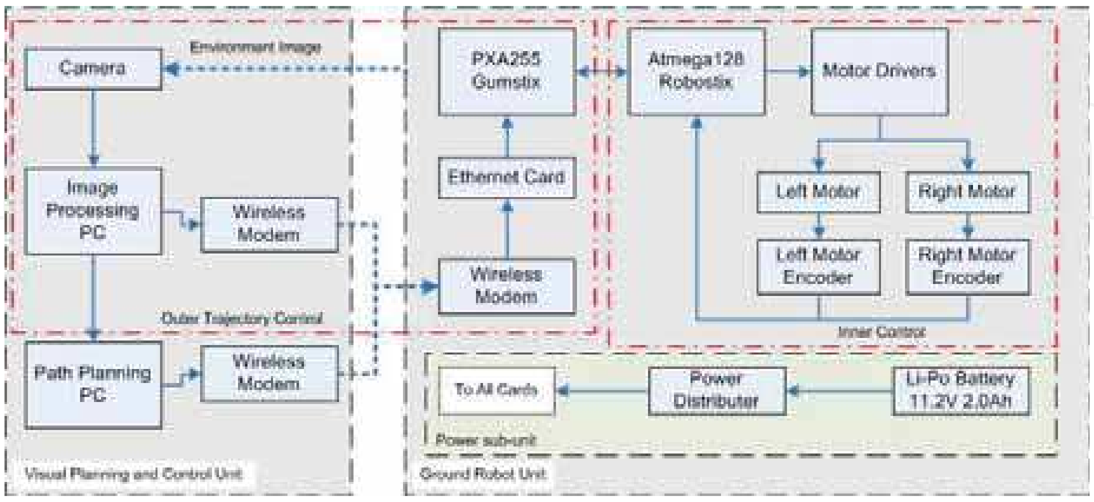


Fig. 12. System architecture of ITUCAL Robotic Testbed

5.2 Visual Positioning

The configurations and color-ids(for multi-robot applications) of the robots are detected by an overhead vision positioning system that is mounted on top of the platform. Every robot have two spots on its top side and one of them takes place center of the robot while other one takes place on heading of the robot with its own id color. Configurations of the robot consists of position and heading direction is tracked thanks to these center and heading spots. General appearance of the testbed can be seen in Fig. 11.
Visual positioning application runs on a centralized PC and uses two CCTV camera to cover the platform. Taken images are processed to detect both configurations of the robots and the obstacles. Processed images are turned in to point-cloud matrix to perform in the path planner computer. All visual positioning application is coded on MATLAB and visual positioning unit can give us configurations of the robot and obstacles in 15 Hz while path planning unit gives solution in > 3 Hz on this application specifications. Example processed images can be seen in Fig. 13

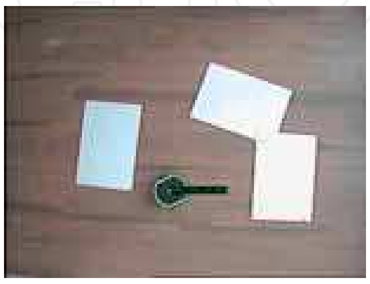
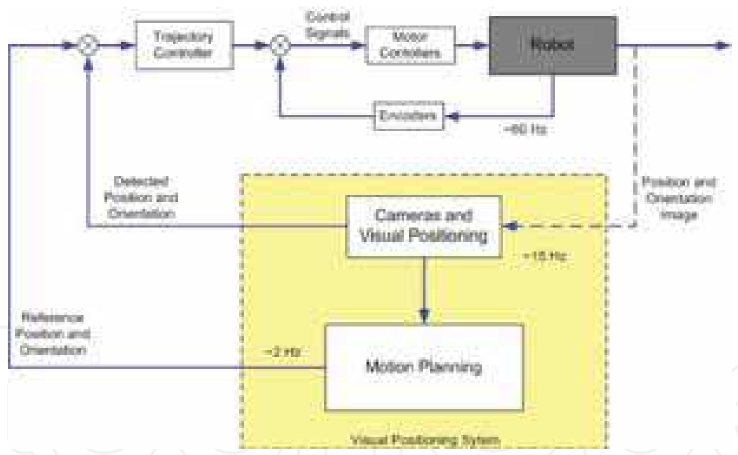


Fig. 13. Example processed environment image, detection obstacles and robots

5.3 Network Communication
Two centralized computers (visual positioning and path planner) and robots' computers runs on the system at the same time. These all computers are connected through a LAN network. Communication among the centralized PCs is performed with the physical ethernet cable while Centralized PCs and robots are connected with wireless network. Data communication between the units are demonstrated in Fig. 12.
Testbed Network is based on a publish/subscribe architecture. To broadcast messages, sender publishes a message to all subscribers and receivers accepts only messages belongs to them according to head-tags of the messages.

5.4 Performing Low-Level Control
Every robots have ability to run own low-level control algorithms. Outer loop control algorithm, a nonlinear trajectory controller, runs on robots' own embedded Linux computers (Gumstix). To perform this algorithm, reference path is received from Path Planner PC while current configurations is received from Visual Positioning PC via wireless network. According to position and orientation errors, trajectory controller evaluates the angular velocities of both two motors that leads the robot to track reference path. Evaluated angular velocities are sent to microcontroller (Robostix) as reference control variables through UART port. Robostix also counts the pulses of the optic encoders of the motors to evaluate current angular velocities. Received reference angular velocities and current angular velocities are compared and PWM signals are generated via PID controllers (as inner control loop) and then these PWM signals are sent to motor drivers. These both application is coded in C performs on Gumstix and Robostix. All these control architecture demonstrated in Fig. 14.


Trajectory Controller
Control Signals
Motor Controllers
Robot
Encoders
~60 Hz
Detected Position and Orientation
Cameras and Visual Positioning
Position and Orientation Image
~15 Hz
Reference Position and Orientation
~2 Hz
Motion Planning
Visual Positioning Sytem
Fig. 14. Control Architecture of the ITUCAL Robotic Testbed

6. Experiments and Simulation Results

6.1 Physical Hardware Demonstrations
To demonstrate the applicability of the algorithms on physical systems, robot experiments have been implemented. In first group experiments on the ITU CAL Robotic Testbed, simplified version of the Modal-Maneuver Based PRM is used. On this application, two independently controlled primitive maneuver modes -straight forward mode and turning mode- are

used on robots. Example tunnel problem solving tests can be seen in Fig. 15. On this experiment, the path planner evaluated its solution in 357 ms. Visual positioning system published current configurations in 7 Hz and motor controllers run at 60 Hz. Robot completed its all motion in 57 s.

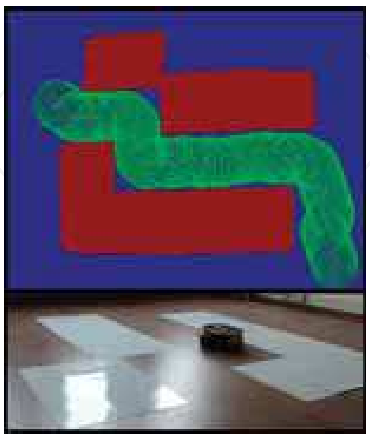


Fig. 15. Modal-Maneuver Based PRM experiments on Robotic Testbed

In the second group experiments in the ITU CAL Robotic-Testbed, Probabilistic B-Spline Based Trajectory Planner is implemented. A capture seen in Figure 16 is taken from one of the experiments of the Probabilistic B-Spline Planner in robotic testbed. Evaluated path is tracked by a nonlinear control algorithm runs on robots computer. This experiment took 29 s and the path planner evaluated its solution in 278 ms while visual positioning system published current configurations in 15 Hz and motor controllers run at 60 Hz.

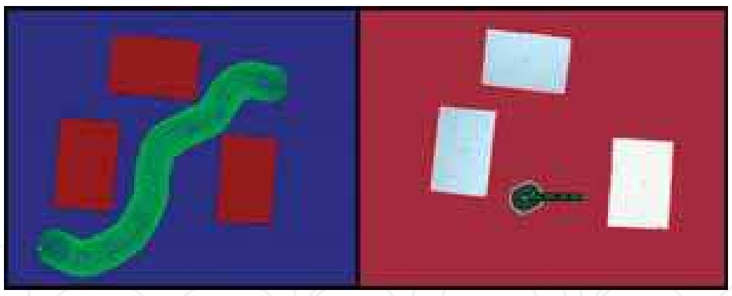


Fig. 16. Probabilistic B-Spline Trajectory Planner experiments on Robotic Testbed

6.2 Simulations of 3D Environments

To illustrate the applicability the algorithms on 3D complex environments in varying ratio of obstacle-space, performance of the algorithms is tested 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. All the experiments were conducted on a 3.00 GHz Intel Pentium(R) 4 processor and the average results are obtained over 50 runs.
In the first group simulations, Modal-Maneuver Based PRM algorithm for aerial vehicles is tested and computational times of the all phases of the algorithm are illustrated in Table 2
As can be seen in results, total times mostly based on Mode-Based Planner phase. As anticipated, increasing blocked space also increases the solution time as seen in Mostly-Blocked

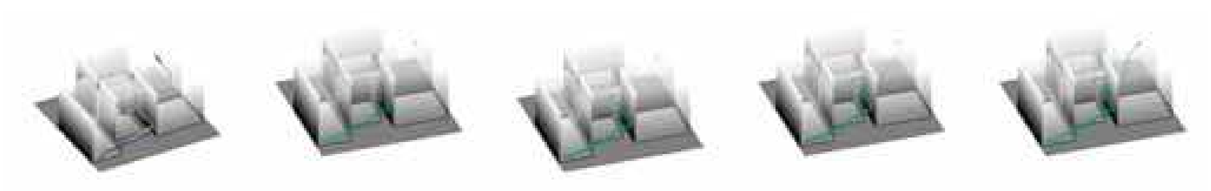


Fig. 17. Simulation example; Modal-Maneuver Based PRM Path Planer Construction Steps for City-Like Environment

		Connectivity Path Planner	Filtering Phase	Mode-Based Planner	Total Time
Single-Narrow Passage	time	0.350 s	0.032 s	32.706 s	33.089 s
	std	0.231 s	0.008 s	17.699 s	17.732 s
City-Like Environment	time	0.712 s	0.036 s	42.397 s	43.145 s
	std	0.942 s	0.008 s	24.478 s	24.639 s
Mostly-Blocked Environment	time	1.376 s	0.042 s	222.229 s	223.648 s
	std	1.132 s	0.011 s	273.451 s	273.134 s

Table 2. Modal-Based PRM Path Planer Construction Times (Seconds)

environment test. However, this increasing rate does not grow exponentially according to percentage of obstacle space. Note that in this approach, modal inputs of the independently controlled modes directly obtained that can be used by low level control layers. Therefore, this approach may be seen slower than other path planner methods, but this method significantly decrease task-load of the low-level layers and should be compared with kinodynamic approaches.
In the second group simulations, we tested the performance of Probabilistic B-Spline Trajectory Planning method on 3D environments. The computational times of steps of the algorithm are illustrated in Table 3 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.

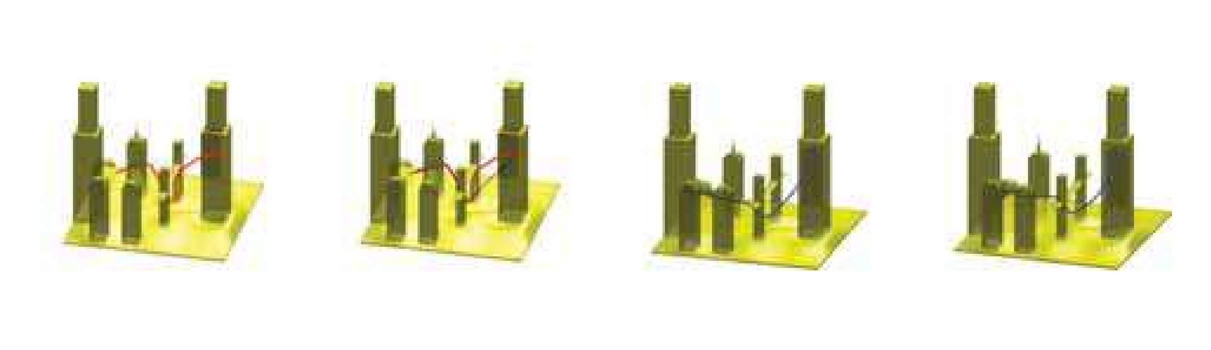


Fig. 18. Simulation example; Probabilistic B-Spline Path Planer Construction Steps for MelCity Model Environment

On this approach, increasing complexity of the environment, as shown in Table 2, 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

		Connectivity Path Planner & Filtering	B-Spline-Based Planner	Total Time
Single-Passage Problem	avr	0.440 s	0.206 s	0.646 s
	std	0.287 s	0.011 s	0.293 s
City-Like Environment	avr	0.977 s	0.326 s	1.303 s
	std	0.935 s	0.254 s	0.930 s
Mostly-Blocked Environment	avr	3.930 s	2.182 s	5.837 s
	std	2.504 s	3.347 s	3.912 s
MelCity Model Volume; 2^3x	avr	3.306 s	0.538 s	3.844 s
	std	1.528 s	0.650 s	1.212 s

Table 3. Mode-Based Path Planer Construction Times (Seconds)

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. 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.

7. 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. Hence, a real-time implementable two step planner strategy is implemented for obtaining 3D flight-path generation for an Unmanned Aerial Vehicles in 3D Complex environments. Thus simplifications on the problem improved the real time implement ability.
In our approach, 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 second step, we explained two different methods to generate dynamically feasible trajectory. First one that we called Modal-Maneuver Based PRM Planner is developed for agile unmanned aerial vehicles that their maneuvers can be define with distinct modes. This allows significant decreases in control input space and thus search dimensions. In this approach the resulting connectivity path and the corresponding milestones are refined with a single query Probabilistic Road Map (PRM) implementation that creates dynamically feasible flight paths with distinct flight mode selections and their modal control inputs. In our second approach, remaining way points are connected with cubic (C^2 continuous) B-Spline curve and this curve is repaired probabilistically to obtain a geometrically (prevents collisions) and dynamically feasible (considers velocity and acceleration constraints) path. At the end, the *time scaling* approach allow dynamic achievability considering the velocity and acceleration limits of the aircrafts. Resulting strategy is tested on real-time physical hardware system using ITU CAL mobile robot testbed for 2D environments and simulations for 3D complex environments. Computational times showed satisfactory results to used for real time implementation for UAVs operations in challenging urban environments.

One of the limitations of the algortihm is on very narrow passages, which require aircraft to tilt considerably to avoid collision. In the problems we have examined distance between obstacles are far wider compared to wing span of the aircraft so we didn't include this case. One of the possible future works is to handle these extreme cases. Moreover, extension of the algorithms presented to UAV fleets is another natural application of this work.

8. References

Bayazit, O. B., Xie, D. & Amato, N. M. (2005). Iterative relaxation of constraints: a framework for improving automated motion planning, *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on* pp. 3433–3440.

Bohlin, R. & Kavraki, L. E. (2001). A randomized algorithm for robot path planning based on lazy evaluation, *Handbook on Randomized Computing, Kluwer Academic Publishers, p.221-249 (2001)* pp. 221–249.

Boor, V., Overmars, M. H. & van der Stappen, A. F. (1999). The gaussian sampling strategy for probabilistic roadmap planners, *IEEE International Conference on Robotics & Automation* p. 6.

Clark, C. M., Rock, S. & Latombe, J.-C. (2003). Dynamic networks for motion planning in multi-robot space systems, p. 8.

Dyllong, E. & Visioli, A. (2003). Planning and real-time modifications of a trajectory using spline techniques, *Robotica* **21**(5): 475–482.

Frazzoli, E., Dahleh, M. A. & Feron, E. (2002). Real-time motion planning for agile autonomous vehicles, *AIAA Journal of Guidance and Control* **25**(1): 116–129.

Ghosh, R. & Tomlin, C. (2000). Nonlinear inverse dynamic control for mode-based flight, *AIAA Guidance, Navigation and Control Conference* .

Hsu, D. (2000). Randomized single-query motion planning in expansive spaces, *PhD Thesis* p. 134.

Hsu, D., Jiang, T., Reif, J. & Sun, Z. (2003). The bridge test for sampling narrow passages with probabilistic roadmap planners, *IEEE International Conference on Robotics & Automation* .

Hsu, D., Kavraki, L. E., Latombe, J.-C., Motwani, R. & Sorkin, S. (1998). On finding narrow passages with probabilistic roadmap planners, *International Workshop on Algorithmic Foundations of Robotics* pp. 141 – 153.

Hsu, D., Kindel, R., Latombe, J.-C. & Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles, *International Journal of Robotics Research* **21**(2): 233 – 255.

Hsu, D., Latombe, J.-C. & Motwani, R. (1999). Path planning in expansive configuration spaces, *International Journal Computational Geometry and Applications* **4**: 495–512.

Inalhan, G., Stipanovic, D. & Tomlin, C. (2002). Decentralized optimization, with application to multiple aircraft coordination, *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on* **1**: 1147–1155 vol.1.

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

Kindel, R., Hsu, D., claude Robert, J. & Latombe, S. (2000). Randomized kinodynamic motion planning with moving obstacles, *The International Journal of Robotics Research* **21**(3): 233–255.

Komoriya, K. & Tanie, K. (1989). Trajectory design and control of a wheel-type mobile robot using b-spline curve, *Intelligent Robots and Systems '89. The Autonomous Mobile Robots* *and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on* pp. 398 – 405.

Koyuncu, E. & Inalhan, G. (2008). A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments, *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* pp. 815 – 821.

Koyuncu, E., Ure, N. K. & Inalhan, G. (2008). 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* .

Koyuncu, E., Ure, N. K. & Inalhan, G. (2009). Integration of path/maneuver planning in complex environments for agile maneuvering ucavs, *Proc. 2th Int. Symposium on Unmanned Aerial Vehicles (UAV'09)* .

LaValle, S. & Kuffner, J. (1999). Randomized kinodynamic planning, *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* **1**: 473 – 479 vol.1.

Munoz, V., Ollero, A., Prado, M. & Simon, A. (1994). Mobile robot trajectory planning with dynamic and kinematic constraints, *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on* pp. 2802 – 2807 vol.4.

Nikolos, I., Valavanis, K., Tsourveloudis, N. & Kostaras, A. (2003). Evolutionary algorithm based offline/online path planner for uav navigation, *Systems, Man, and Cybernetics, Part B, IEEE Transactions on* **33**(6): 898–912.

Paulos, E. (1998). On-line collision avoidance for multiple robots using b-splines, *University of California Berkeley Computer Science Division (EECS) Technical Report* **98-977**.

Piegl, L. A. & Tiller, W. (1997). *The NURBS BookâĂŐ*, Springer-Verlag New York, Inc.

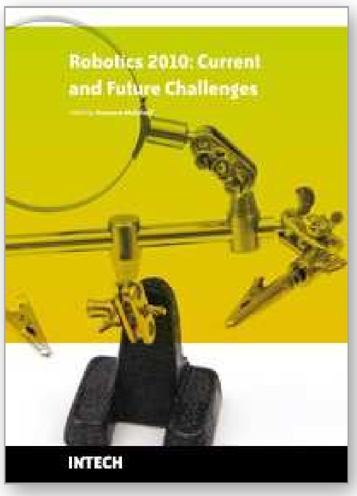
Schouwenaars, T., Feron, E. & How, J. (2004). Hybrid model for receding horizon guidance of agile autonomous rotorcraft, *IFAC Symposium on Automatic Control* .

Song, G. & Amato, N. (2001). Randomized motion planning for car-like robots with c-prm, *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on* **1**: 37 – 42 vol.1.

Ure, N. & Inalhan, G. (2008). Design of higher order sliding mode control laws for multi modal agile maneuvering ucavs, *2nd Int. Symposium on Systems and Controls in Aerospace* .

Ure, N. & Inalhan, G. (2009). Design of a multi modal control framework for agile maneuvering ucavs, *IEEE Aerospace Conference* .

Vazquez, G. B., Sossa, A. H. & de Leon, S. J. L. D. (1994). Auto guided vehicle control using expanded time b-splines, *Systems, Man, and Cybernetics,Humans, Information and Technology, IEEE International Conference on* **3**: 2786–2791 vol. 3.



ISBN 978-953-7619-78-7
Hard cover, 494 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010