Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

# A Probabilistic Algorithm for Mode Based Motion Planning of Agile Unmanned Air Vehicles in Complex Environments

**Emre Koyuncu*, N. Kemal Ure*,
Gokhan Inalhan****

**Controls and Avionics Laboratory, Istanbul Technical University, Istanbul, Turkey, (e-mail: emre.koyuncu@itu.edu.tr, ure@itu.edu.tr)*
***Faculty of Aeronautics and Astronautics, Istanbul Technical University, Istanbul, Turkey, (e-mail: inalhan@itu.edu.tr)*

**Abstract:** In this work, we consider the design of a probabilistic trajectory planner for a highly maneuverable unmanned air vehicle flying in a dense and complex city-like environment. Our design hinges on the decomposition of the problem into a) flight controls of fundamental agile-maneuvering flight modes and b) trajectory planning using these controlled flight modes from which almost any aggressive maneuver (or a combination of) can be created. 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. Focusing on the trajectory planning part, we provide a three-step probabilistic trajectory planner. In the first step, the algorithm rapidly explores the environment through a randomized reachability tree search using an approximate line segment models. The resulting connecting path is converted into flight milestones through a line-of-sight segmentation. This 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. We address the problematic issue of narrow passages through non-uniform distributed capture regions, which prefer state solutions that align the vehicle to enter the milestone region in line with the next milestone to come. Numerical simulations in 3D and 2D demonstrate the ability of the method to provide real-time solutions in dense and complex environments.

## 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. Specifically, we design a trajectory planning algorithm that utilizes the full flight envelope of a highly maneuverable air vehicle while moving through this environment.

Our approach 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 doesn't 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 paper. To address these problems, we consider a three-step probabilistic trajectory planning algorithm.

In the first step, the algorithm explores the complex environment and the passages through a randomized reachability tree using a simplified version of standard RRT implementation. The end result is a connectivity path with line segments that can be tracked from the initial point to the goal point in the configuration space. It only gives a sense of passages that may be tracked in the environment in the next steps of planner.

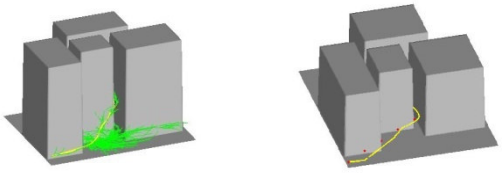

Fig. 1. Complete solution of the mode based probabilistic trajectory planner for an aircraft.

This connectivity path is then filtered in the second step of our method with a line-of-sight segmentation. In this phase, running through each of the points as set by the connectivity path, only the last points that can be seen line-of-sight by its previous milestone point are marked and the others (which are in between, and cause fluctuations on the *connectivity*

*path* as a result of probabilistic approach) are eliminated. The end result is a clean flight path with milestones (i.e. flight way-points) that generally appear as long straight flight segments that occasionally enter and exit passages. As a result of the underlying randomized exploration towards the goal region, the tendency of the path is towards larger passages and towards direct paths that lead to the goal region. This provides a reasonably good approximation for a flyable trajectory.

In the last step, a *dynamically feasible path* between the *way-points* (and their possible neighborhood relaxations) is created with a single-query probabilistic road map planner. We create milestones 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. To address the issue of flight-attainability, specifically while entering and exiting fly-through passages, all the way-points are covered by "approaching field distribution" that will favor randomized paths that align with the next way-point. A complete solution of our approach is illustrated in Fig. 1.

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 the next subsection, we review the key computational and solution quality aspects in light of the related work.

### *1.1 Previous and Related Works*

In the last few decades randomized sampling-based motion planning algorithms have shown success in solving challenging robotic 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 paths must go through narrow passages – a point which will be addressed in more detail in Section 2. Tree-based planners build a new roadmap for each query and the newly produced samples of each query are connected to the samples that are already existing in the tree as in (Hsu, et. al., 1997), (Hsu, 2000), (Sanchez and Latombe, 2003), (Plaku, et. al., 2005) and (LaValle, 1998). Rapidly-Exploring Random Tree (RRT) is 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.

Over the past several years, sampling-based planners, especially tree-based planners (RRT and single-query PRM variant), have been adapted to solve dynamically feasible path accommodate kinodynamic constraints. The main philosophy behind kinodynamic planning is searching a higher dimensional state space that captures the dynamics of the system as mainly mentioned in (LaValle and Kuffner, 2001) and (Hsu, et al., 2002) via the above two distinct sampling strategies. In single query PRM implementations, increasing the number of milestones within the tree causes the solution time to increase exponentially as a result of the tree size. One of the key points within our strategy is to segment the overall space into small spaces by defining *way points*. Instead of searching the whole configuration space with one PRM algorithm (that chooses milestones in a potentially huge PRM tree) , we are searching feasible paths between way points with small PRM trees by trusting RRT planner's exploring ability we used in first step. Also note the important experimental results of RRT based path planning for micro air vehicles in (Griffiths, Beard et. al. 2007).

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.

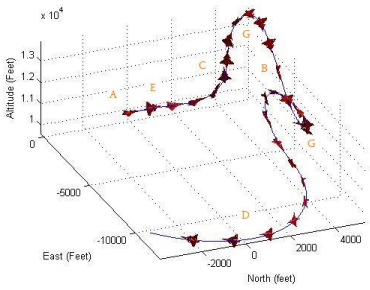


Fig. 2. Nonlinear control of an aggressive maneuvering of F-16 over flight modes using the full-envelope dynamic model including stall.

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 (Tomlin and Rosh, 2000). Note that

this modal approach can also be used for trajectory optimization for multiple vehicles (Inalhan, Stipanovic, Tomlin, 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 towards 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. 2. Detailed description of this will be given in Section 3.

Rest of paper organized as follows. In Section 2; we describe the motion planning algorithm in two subsections: 2.1 Finding the *connectivity path* and 2.2 Finding the Feasibility Path. Section 2.1 specifically covers the RRT implementation (2.1.1) and the path filters (2.1.2), where as Section 2.2 focuses on single-query PRM implementation details and numerical results. Section 3 provides an example on the concept of the modal-based maneuvers as perceived from flight dynamics point of view. The conclusions and future work are discussed in Section 4.

## 2. MOTION PLANNING

Our general motion planning strategy is first to find *connectivity path* rapidly disregarding the vehicle's attitude dynamics. In this phase, we use RRT Algorithm because of its quick spreading ability. After this first phase, *connectivity path* is refined and re-gridded as can be reached *way points*. Finally, feasible paths are searched by PRM Algorithm between these *way-points* with maneuvers selected from Modal-Based maneuver set of vehicle. Furthermore, we define joint fields on way points as *approaching field* is mentioned in Feasible Path section. All of these processes can be seen in Fig. 3.

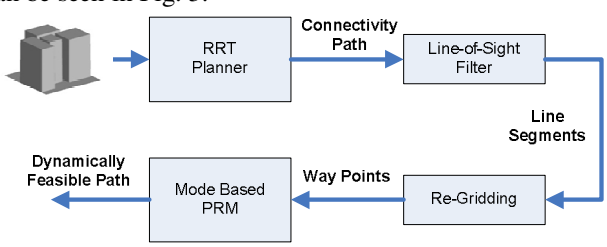


Fig. 3. Motion Planning Strategy

Note that, to facilitate the intelligibility on figures, we choose 2D environment demonstrations, but our main concern is 3D complex environments.

### *2.1 Connectivity Path*

#### *2.1.1 Goal Biased RRT Algorithm*

RRT is an exploration algorithm for quickly searching high-dimensional spaces that have both global and differential constraints (Lavalle, 2002). However, one of the important drawbacks of this method to use as a stand-alone planner is biasing of the distribution of milestones towards the obstacle regions if the configuration space has large obstacles. This drawback may be caused undesirable and severely slow down the rate of convergence as mentioned in (Hsu, et al., 2002). We are only motivated by RRT's good property to obtain *connectivity path* that can be tracked during flight. In this phase, our strategy does not focus on feasibility of the path and we do not do an input space sampling. In this part, we only investigate whether the points can be connected to each other while not hitting the obstacles. Construction of Goal Biased RRT algorithm is given below as Algorithm I.

**Algorithm I: Goal Biased RRT Algorithm**

**Input** : initial and goal positions; $q_{init}$ , $q_{goal}$
**Output**: *connectivity path*
1: **Add** initial point $q_{init}$ to $\tau$ tree and $i \leftarrow 1$
2: **repeat**
3: **Select** random point $m_{rand}$ in C
4: **Select** nearest neighbour $m_{near}$ of $m_{rand}$ in $\tau$ tree
5: **Generate** $m_{new}$ from $m_{near}$ toward $m_{rand}$
6: **Generate** trajectory $e_{new}$ from $m_{near}$ to $m_{new}$
7: **if** $m_{new}$ and $e_{new}$are in collision-free region **then**
8: **Add** $m_{new}$ to $\tau$ tree and $i+1$
9: **if** $m_{new}$ in *end region* **then**
10: **break** with *success*
11: **Select** nearest neighbour $m_{near}$ of $q_{goal}$ in $\tau$ tree
12: **Generate** $m_{new}$ from $m_{near}$ toward $q_{goal}$
13: **Generate** trajectory $e_{new}$ from $m_{near}$ to $m_{new}$
14: **if** $m_{new}$ and $e_{new}$are in collision-free region **then**
15: **Add** $m_{new}$ to $\tau$ tree and $i+1$
16: **if** $m_{new}$ in *end region* **then**
17: **break** with *success*
18: **if** $i$= N max iteration number **then**
19: **break** with *fail*
20: **until** *end region* is reached with *success*
21: **Select** *connectivity path* points can be gone back *end region* to initial point in $\tau$ tree

In this part of algorithm, each loop attempts to extend the $\tau$ tree first towards the random selected point $m_{rand}$, and second towards the goal point by adding new points. To select points, nearest point already within the $\tau$ tree to the sampled random point (in Line 4) and the nearest point to the goal point is selected (in Line 11) respectively. *Generate* functions generates new point $m_{new}$ on the direction of the selected nearest points $m_{near}$ at random selected distances as shown in Line 5 and 12. If direction angles exceed predefined limits, max direction angles are selected. These boundaries

may be chosen according to vehicle's kinematic boundaries. If new generated point and its trajectory is in obstacle-free region (checked in Line 7 and 14) then $m_{new}$ is added $\tau$ tree as shown in Line 8 and 15. If $\tau$ tree reaches *end region* anywhere, algorithm returns with success and gives *connectivity path. End region* can be obtained within a tolerable capture region as explained in (Hsu and et al., 2002). An example simulation in 2D is illustrated in Fig. 4.

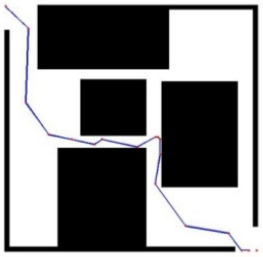

Fig. 4. Connectivity Path

*2.1.2 Path Filters*

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 some filters. Line-of-Sight Filter algorithm 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 narrow passages and inherently hard regions. These fields give a sense of agile maneuverings that are needed to fly over these points.

**Algorithm II** : ***Line-of-Sight Filtering***

**Input**: *connectivity path*
**Output**: *visible points*
1: **Add** initial point $m_1$ to *visible points*,
$m_{visib} \leftarrow m_1$ and $m_i \leftarrow m_2$
2: **repeat**
3: **Generate** line $l_{visib}$ from $m_{visib}$ to $m_i$
4: **if** $l_{visib}$ collides with an obstacle
5: **Add** $m_{i-1}$ to *visible points*
6: $m_{visib} \leftarrow m_{i-1}$
7: **else**
8: $i + 1$
9: **until** last point of *connectivity path* $m_m$ is reached

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 connectable 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.

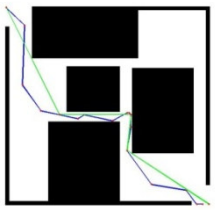

Fig.5 Line-of-Sight filter result.

Sometimes, the distance between two neighbor points in the *visible points* can be long and therefore it could potentially take a long time to find a dynamically feasible path with a PRM implementation. To address this potential pitfall, we implement a simple *Re-Gridding* algorithm that adds new guide points between neighboring points, if the distance is longer than a pre-described threshold; *long*. This term is defined as distance metric value and is chosen according to density of environment.

**Algorithm III** : ***Re-Gridding***

**Input**: *visible points*
**Output**: *way points*
1: **Add** *visible points* to *way points*
2: **for** $i_{th}$ point $m_i$ in *way points;*
from $m_2$ to last point $m_m$
3: **if** distance from $m_{i-1}$ to $m_i$ is *long* **then**
4: **Add** point to *way points* middle of $m_{i-1}$ and $m_i$

*2.2 Dynamically Feasible Path: Mode-Based PRM*

Previous subsection provided the methods for 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 towards the goal region, the tendency of the path is towards larger passages and towards direct paths that lead to the goal region. Although this provides a reasonably good approximation, it doesn't necessarily correspond to a flyable trajectory as it is based on a simple point mass model with velocity and heading.

The last part of our planning strategy is an extension of single-query PRM algorithm that given as Algorithm IV. It iteratively builds a tree-shaped roadmap to connect the way-points one-by-one. In every inner loop, it first selects at random a milestone as in Line 5, maneuver-mode as in Line 6 and modal inputs as in Line 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 11), its end point is added to tree as a new milestone as shown in Line 16 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.

**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. Therefore, if this milestone is selected by planner, higher velocity rates (in constant time, longer distance rates) are mostly preferred as modal input.

**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 intended to create more smooth flight segments. This tunneling effect provides a straight-forward solution to the classical narrow passage effect seen in standard PRM methods.

To overcome this narrow passage problem, some of the previous works focused on improving sampling important areas of the configuration space using workspace information as in (Boor, et. al.,1999) (Hsu, et. al., 2003) and (Kurniawati and Hsu, 2004). They are briefly discussed in (Tsianos, et al.,2007) and (Saha and Latombe, 2005). Some retraction methods use medial axis are proposed as in (Holleman and Kavraki, 2000) and (Lien, et. al., 2003) that are based on the slightly fattening the free space, constructing a roadmap in fattened free space and repairing colliding portions of roadmap. But especially for the complex dynamic and kinematic system, repairing strategy cannot be obtained or solution may take a considerable amount of time.

**Algorithm IV** : **Mode-Based PRM Planner**

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

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 21 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 and example completed solution for 3D environment is given in Fig. 7.

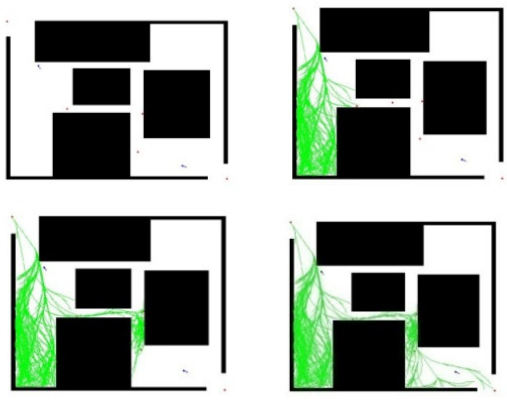

Fig. 6. Mode-Based PRM Searching

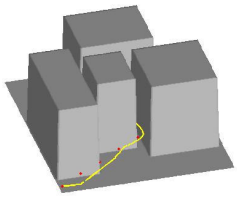

Fig. 7. Dynamically Feasible Path for 3D city-like environment

We tested the performance of our method on some dense environments in varying ratio of obstacle space to all work space. The results are illustrated in Fig. 9 for 2D and 3D environments. All the experiments were conducted on a 1.60 GHz Intel Pentium M processor and the average results are obtained over 10 runs. The experiment setting is arranged such that the vehicle will be able to cruise to the goal region in 40 secs (2D) and 60 secs (3D) if not faced with obstacles.

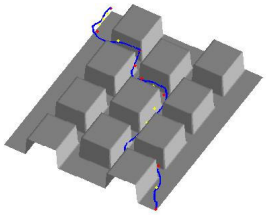

Fig. 8. Example solution in 3D workspace that has 44 % of volume is blocked by obstacles

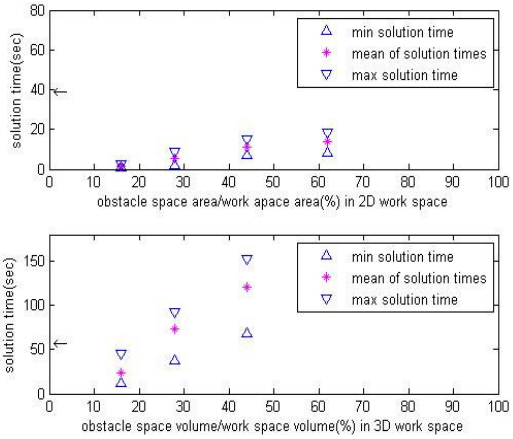


Fig. 9. Minimum, mean and maximum values of solution times according to varying percentages of blocked space.

During the all of this implementation, solution time of finding a Connectivity Path were observed to below 1.6 seconds for 2D workspace and below 3.6 seconds for 3D workspaces in general. As anticipated, increasing blocked space also increases the solution time. However, this increasing rate does not grow exponentially according to percentage of obstacle space and the algorithm can be implemented. In addition, the solution times suggest that our method will be applicable for real-time implementations as the solution time is favorably comparable to implementation times.

## 3. MODAL-BASED MANEUVERING

The main idea of modal based maneuvering is to divide the motion of an aircraft into simpler modal blocks, and to build any maneuver using these blocks. Many of the standard agile maneuvers and standard flight patterns implement these modes listed below:

- Primitive Modes: Level Flight, Climb/Descent, Longitudinal Loop, Lateral Loop
- Transition Modes: Stability Axis Roll, Re-Orientation
- Complex Modes: 3D Spin

Primary modes are the most basic building blocks of arbitrary maneuver sets. In order to sequence these modes efficiently, we define some transition maneuvers between them. For example aircraft must go through a roll mode without sideslip to transfer from level flight to coordinated turn, or it can be desired to invert the aircraft with a stability axis roll. Also in case the aircraft results in an undesired disoriented attitude, re-orientation mode regulates the aircraft into a stable safe state in which it can begin to execute any of the primary modes. This mode also prevents aircraft from stalling due to high angle-of-attack flight. The final mode is called a complex mode, because it actually entails an attitude dictated combination of primary modes listed above, this mode is used to control whenever two of the primary modes must be executed simultaneously.

For example, a complex helical maneuver (to obtain a gradual "focus on" loitering) implements a simultaneous pure pitch and coordinated turn (or simply a constant roll) action at the same time (means that maneuver is governed by complex mode), where as an Immelmann Turn (implemented to do a 180 degrees direction change with a tactical height gain advantage) implements a level-flight, longitudinal loop, roll and level flight modes in sequence. Both complex and basic maneuvers are used in motion planning where each mode has its use and their sequence is selected from a probabilistic algorithm as described in the previous section. In our complimentary work (Ure, and Inalhan, 2008), we demonstrated the ability to achieve full-controlled autonomous aggressive maneuvering of these modes via a switching sliding mode control design for an F-16 over the full flight envelope including stall. This is also illustrated in Fig. 2 for a complex maneuver. Please refer to the authors to obtain the pre-publication results on this.

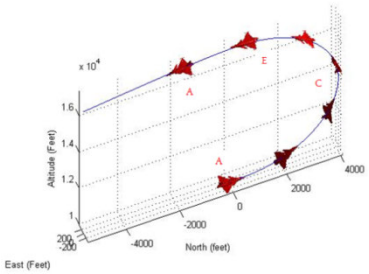


Fig. 10. Controlled Immelmann Turn using flight mode based control structure

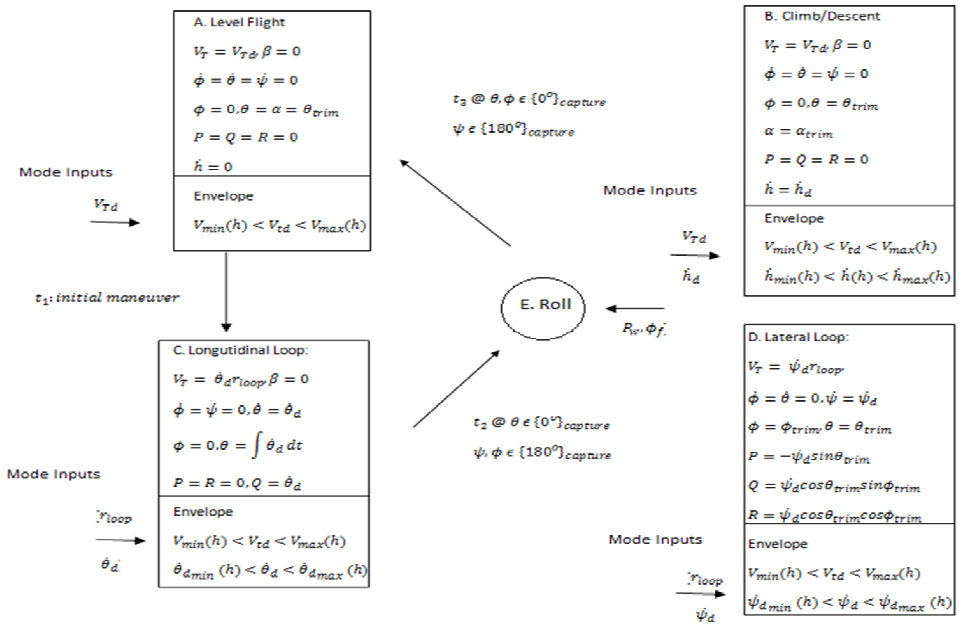


Fig. 11 Immelmann Maneuver Mode Chart.

From a path planning point of view, this actually brings a considerable advantage, as the simultaneous control and trajectory planning problem over the whole state variables and control inputs can be decomposed into a) flight control design for specific modes and b) path planning over these prescribed modes with a small set of design parameters. To illustrate this, consider the standard six degrees of freedom nonlinear dynamics model of an aircraft, in which the state variables are defined as follows;$U, V, W$are velocities in wind axes, $\phi, \theta, \psi$ are roll, pitch and yaw Euler angle set, $P, Q, R$ are body angular rates. With the total velocity corresponding to$V_T$, and aerodynamic angles (angle of attack and sideslip angle) are $\propto$ and $\beta$.

From a controls point of view, once the controls are designed for obtaining the six specific modes these modes can be specified via reference input signals denoted simply by *A. Level Flight*($V_{Td}$), *B. Climb/Descent*($V_{Td}$, $\dot{h}_d$), *C. Longitudinal Loop*($r_{loop}, \dot{\theta}_d$), *D. Lateral Loop*($r_{loop}, \dot{\psi}_d$), *E. Stability Axis Roll* ($P_w, \phi_f$), *F. Re-Orientation* , *G. 3D Spin*($P_d, Q_d, R_d$). It must be indicated that, these modes have their limits in flight envelope of the aircraft; these are usually in terms of modal inputs. For example in level flight, aircraft's maximum and minimum velocity corresponding to current altitude are determined before the execution and if the mode specifications are not within the limits; aircraft can suffer from actuator saturation or stall due to high angle of attack. So we specify envelopes for each mode and ensure the command signal lies within these intervals to provide feasibility.

Under this basic dynamic equations can be compacted using the basic mode specifications over a general Mode Maneuver Chart. For an example, Immelmann Turn is illustrated in Fig. X with capture coordinates corresponding to the desired switching sequence – ACEA and a simulation result for this maneuver for an F-16 aircraft (using the full-envelope model including stall) is shown in Fig. 10. Note that we haven't included the re-orientation and 3D spin modes for simplicity of presentation.

## 4. CONCLUSIONS

Trajectory design of an air vehicle in dense and complex city-like 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 second facet via the decomposition of the flight controls and the trajectory planning using flight modes from which almost any aggressive maneuver can be created. Our three step hybrid mode based probabilistic trajectory planner aims at rapid generation of reachability tree (and hence milestones) to any desired physical location, and feasible trajectory generation over the milestones utilizing the basic modes. In our numerical simulations, we observed that the approach not only has real-time implementation capability, but also shows features which will allow alternative path creations through distributed computing and plan-as-you-along capability.

In our future work, we will explore these issues with a distributed seeding, simultaneous search of the complex environment. Inclusion of dynamically varying and reacting environments is another point that needs to be addressed for any realistic experimental validation. Although through numerous implementations, RRTs are observed to be very good at finding a feasible solution; one of the key open questions is the non-existence of a formal certificate of guarantee to find a feasible solution if one exists.

REFERENCES

Boor, V. M.H. Overmars, F.v. Stappen (1999). The Gaussian Sampling Strategy for Probabilistic Roadmap Planners, *In. Proc. IEEE Int. Conf. on Robotics & Automation*, 1018–1023.

Cheng, P., Z. Shen, S.M. LaValle, (2001). RRT-Based Trajectory Design for Autonomous Automobiles and Spacecraft, *Archives of Control Sciences*, **11(3-4)**, 167-194.

Frazzoli, E., (2001). Robust Hybrid Control for Autonomous Vehicle Motion Planning, *PhD Thesis,* Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA.

Frazzoli, E., M.A. Dahleh and E. Feron (2002). Real-Time Motion Planning for Agile Autonomous Vehicles, *AIAA J. of Guidance, Control, and Dynamics,* **25(1)**, 116–129.

Ghosh, R., C. Tomlin (2000). Nonlinear inverse dynamic control for mode based flight, *Proceedings of the AIAAGuidance, Navigation and Control Conference*

Griffiths, S., J. Sanuders, A. Curtis, T. Maclain, R. Beard (2007). Obstacle and Terrain aviodance for miniature aerial vehicles, *IEEE Robotics and Automation Magazine*,Subimtted for review

Holleman, C., and L.E. Kavraki (2000). A Framework for Using the Workspace Medial Axis in PRM Planners. In *Proceedings of the 2000 International Conference on Robotics and Automation (ICRA 2000)*, 1408–1413, IEEE Press, San Fransisco, CA.

Hsu, D., J.C. Latombe, and R. Motwani (1997). Path planning in expansive configuration spaces, In *Proc. IEEE Int. Conf. on Robotics & Automation*, 2719–2726.

Hsu, D. (2000). Randomized Single-query Motion Planning in Expansive Spaces, *Ph.D. Thesis*, Dept. of Computer Science, Stanford University, Stanford, CA.

Hsu, D., R. Kindel, J.C. Latombe, S. Rock (2002). Randomized Kinodynamic Motion Planning with Moving Obstacles ,*Int. J. of Robotics Research*, **21(3)**, 233-255.

Hsu, D., T. Jiang, J. Reif, and Z. Sun (2003). Bridge Test for Sampling Narrow Passages with Probabilistic Roadmap Planners, *In Proc. IEEE Int. Conf. Robotics & Automation (ICRA)*, 4420-4426.

Hsu, D., J.C. Latombe, and H. Kurniawati (2006). On the Probabilistic Foundations of Probabilistic Roadmap Planning, *Int. J. of Robotics Research*, **25(7)**, 627-643.

Inalhan G., D.M. Stipanovic, C.J. Tomlin (2002), Decentralized Optimization, with Application to Multiple Aircraft Coordination, *in the Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas.

Kavraki, L.E., P. Svestka, J.C. Latombe, and M. Overmars (1996). Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces, *IEEE Transactions on Robotics and Automation*, **12(4)**, 566-580.

Kurniawati, H., and D. Hsu (2004). Workspace importance sampling for probabilistic roadmap planning, *IEEE/RSJ Int. Conf. on Intelligent Robots Systems.*

Ladd, A.M. and L.E. Kavraki (2005). Motion Planning in the presence of drift, underactuation and discrete dystem changes. In *Robotics: Science and Systems I,*233–241, MIT Press, Boston, MA.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning, *Technical Report 11*, Computer Science Dept., Iowa State University.

LaValle, S.M. (2002). From dynamic programming to RRTs: Algorithmic Design of feasible trajectories, *Control Problems in Robotics*, 19-37. Springer-Verlag, Berlin.

LaValle, S.M., J.J. Kuffner (2001). Raandomized kinodynamic Planning, *Int. Journal of Robotic Research, **20(5)**, 378-400.*

Lien, J.M., S.L. Thomas, and N.M. Amato (2003). In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 4439-4444, Taipei, Taiwan.

Lindemann, S.R., S.M. LaValle (2005). Current issues in sampling-based motion planning, *Robotics Research: The Eleventh International Symposium*, 36-54, Springer-Verlag, Berlin.

Plaku, E., M.Y. Vardi, L.E. Kavraki (2007). Discrete search leading continuous exploration for kinodynamic motion planning, *Robotics: Science and Systems*, Atlanta, Georgia.

Rodriguez, S., S. Thomas, R. Pearce and N.M. Amato (2006). RESAMPL: A Region-Sensitive Adaptive Motion Planner, *Proc. Int. Workshop. on Algorithmic Foundation of Robotics (WAFR)*, New York City, NY.

Saha, M., J.C. Latombe, (2005). Finding Narrow Passages with Probabilistic Roadmaps: The Small-Step Retraction Method, *EEE/RSJ Conf. on Intelligent Robots and Systems*, Edmonton, Canada.

Sanchez, G., J.C. Latombe (2003). A single-query bi-directional probabilistic roadmap planner with lazy collision checking, Int. *Journal of Robotics Researches*, 403-407.

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

Tomlin C., J. Lygeros, L. Benvenuti ,S. Sastry (1995) Output tracking for a non-minimum phase dynamics CTOL aircraft model, *Proceedings of 34th IEEE Conference on Decision and Control*

Tsianos, K.I., I.A. Sucan, L.E. Kavraki (2007). Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review*, pp. (accepted for publication)

Ure, N.K., G. Inalhan (2008). Design of a Full-Envelope Nonlinear Flight Controls for an Aggresive Maneuvering F-16 Aircraft", *Controls and Avionics Laboratory Report.*