İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

PROBABILISTIC MOTION PLANNING IN COMPLEX ENVIRONMENTS FOR UNMANNED AERIAL VEHICLES

M.Sc. Thesis by

Emre KOYUNCU, B.Sc.

Department : **Mechatronics Engineering**

Programme : **Mechatronics Engineering**

JUNE 2008

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

PROBABILISTIC MOTION PLANNING IN COMPLEX ENVIRONMENTS FOR UNMANNED AERIAL VEHICLES

M.Sc. Thesis by

Emre KOYUNCU, B.Sc.

(518051010)

Date of Submission : 5 May 2008

Date of Examin : 13 June 2008

Supervisor : Asst. Prof. Dr. Gökhan İnalhan (İ.T.Ü.)

Members of the Examining Committee Prof. Dr. Çingiz Hacıyev (İ.T.Ü.)

Asst. Prof. Dr. Erdinç Altuğ (İ.T.Ü.)

JUNE 2008

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KARMAŞIK ÇEVRELERDE UÇAN İNSANSIZ HAVA ARAÇLARI İÇİN OLASILIKSAL YÖRÜNGE PLANLAMA

YÜKSEK LİSANS TEZİ

Elektrik Müh. Emre KOYUNCU

(518051010)

Tezin Enstitüye Verildiği Tarih : 5 Mayıs 2008

Tezin Savunulduğu Tarih : 13 Haziran 2008

Tez Danışmanı : Yrd. Doç. Dr. Gökhan İnalhan (İ.T.Ü.)

Diğer Jüri Üyeleri Prof. Dr. Çingiz Hacıyev (İ.T.Ü.)

Yrd. Doç. Dr. Erdinç Altuğ (İ.T.Ü.)

HAZİRAN 2008

ACKNOWLEDGEMENT

I am deeply indebted to Prof. Inalhan for providing me opportunity to do my M.Sc. under his supervision. He has been an exemplary researcher to me and he has been very patiently and diligently advising and guiding me over the last two years. My personality has greatly improved by his outstanding advisory and leadership. His insight and vision have been instrumental in completing this work.

I would like to thank Prof. Güvenç for his very kind helps and advises. His suggestions has been very important for me in the first year of the my M.Sc. research. I would like also to thank Prof. Ertuğrul. I really enjoyed many insightful and fruitful discussions and talks with her.

I would like to special thank N. Kemal Üre different from other members of the Controls and Avionics Laboratory. His collaboration and our academic discussions have been really enjoyable for me. But my research life on CAL probably would have been harder and boring without Miraç Aksugür, Serdar Ateş, Melih Fidanoğlu, Can Kurtuluş and A. Cemil Tural. I am very grateful to these friends have made my times at CAL so memorable. I would also like to thank people from neighbor laboratory; Barış Toktamış and Ozan Haktanır. I never forget the our pleasant Sunday walks and conversations about the photographs. I would never think of my life without İsmail Gerzeli, Serhat Turan, Gürcan Güray, Emre Uncuoğlu and Eren Kayaoğlu. We have countless memories with together and I can never forget them.

Finally, I can not thank my family enough for hanging in there with me. Without their non-ending strong spiritual and financial support, it would have been impossible for me accomplish all of my study and research. Every time, hearing supportive voice of my father, concerned voice of my mother and lovely voice of my sister have made me strong much more and I can never repay them for the love and care that they have shown me.

June 2008 Emre KOYUNCU

TABLE OF CONTENTS

ABBREVATIONS

EST	:	Expansion Space Tree
PRM	:	Probabilistic Road Map
RRT	:	Rapidly-Exploring Random Tree
SBL	:	Single-Query Bi-Directional PRM with Lazy Collision Checking

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLES

$\boldsymbol{C}$	:	Configuration space
$\boldsymbol{C}_{free}$	:	Configuration free space
$\boldsymbol{C}_{obs}$	:	Configuration obstacle space
$\boldsymbol{G}$	:	Roadmap graph
$\boldsymbol{E}$	:	Edge members of the roadmap
$\boldsymbol{q}$	:	Configurations of the vehicle
$\boldsymbol{q}_{init}$	:	Start configuration of the space
$\boldsymbol{q}_{goal}$	:	Goal configuration of the space
$\boldsymbol{q}_{rand}$	:	Random selected configuration
$\boldsymbol{\Omega}$	:	Control space
$\boldsymbol{T}$	:	Tree graph
$\boldsymbol{u}$	:	Control variables of the vehicle
$\boldsymbol{V}$	:	Vertex members of the roadmap
$\boldsymbol{x}$	:	States of the vehicle
$\boldsymbol{\chi}$	:	States space
$\boldsymbol{\chi}_{ric}$	:	State inevitable collision space

PROBABILISTIC MOTION PLANNING IN COMPLEX ENVIRONMENTS FOR UNMANNED AERIAL VEHICLES

SUMMARY

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.

In this work, we consider the design of a probabilistic trajectory planners for an unmanned aerial vehicles flying in a dense and complex city-like environment and we suggest a real-time implementable two-step planner strategy. Our general motion planning strategy, as a first step, is aimed at finding a connectivity path between the initial and the goal point. To decrease the computational time, we disregard the vehicle's dynamic constraints and use RRT method for searching only the configuration space of vehicle. The resulting connecting path is converted into flight milestones through a line-of-sight segmentation. Remaining points that we will call as way points generally appear in entering and exiting regions of the narrow passages that are formed between the obstacles. This gives an advantage to segment the hard path planning problem to a series of small path generating problems on these locations.

After this first step we suggest two methods to create *Dynamically Feasible Path*, first one that we called *Mode Based PRM Planner* is suitable for agile unmanned aerial vehicles that their maneuvers can be define with distinct modes.This mode-based structure is also well suited designing a systematic flight control system. 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. 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. 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.

Our second path planning method that we called *Probabilistic B-Spline Planner* is more suitable for generating constant acceleration flight paths that pass through milestones are founded in the first step. These time-scalable constant

acceleration flight paths approximate unmanned helicopter closed-loop dynamics under trajectory control. In this strategy, every consecutive way points belongs to connectivity path re-connected with B-Spline curves and these curves repaired probabilistically thanks to local support property of B-Spline curves with some repairing methods to obtain dynamically feasible path.

Numerical simulations for 2D and 3D environments indicate the ability of these methods to provide real-time solutions in dense and complex environments for the aerial vehicles.

KARMAŞIK ÇEVRELERDE UÇAN İNSANSIZ HAVA ARAÇLARI İÇİN OLASILIKSAL YÖRÜNGE PLANLAMA

ÖZET

Genel kinodinamik hareket planlayıcılar dinamik sistemlerin parametre uzaylarının en az iki katı olan durum uzaylarının boyutu arttıkça, en az onun eksponensiyel katı kadar artan çözüm zamanlarına ihtiyaç duyarlar. Bu yüzden gerçek uygulamalarda kinodinamik planlayıcılar ancak küçük durum uzaylarına sahip sistemler için uygulanabilirdirler.

Bu çalışmada, yoğun ve karmaşık şehir benzeri çevrelerde uçabilen hava araçlarını göze alarak, olasılıksal yörünge planlayıcılar tasarlamak istemekteyiz ve bunun için gerçek zamanlı uygulanabilir iki kademeli yaklaşım önermekteyiz. Genel yaklaşımımızda ilk adım olarak amacımız, başlangıç ve hedef noktalar arasında bağlantı yörüngesi bulmaktır. Çözümleme zamanını azaltmak için, bu adımda araç dinamik sınırlamaları gözardı edilir ve aracın sadece parametre uzayının taranması için RRT arama metodu kullanılır. Görüş-doğrultusu filtrelemesi ile elde edilen bağlantı yörüngesi uçuş noktalarına dönüştürülür. Sonuç olarak geriye kalan yol noktaları olarak adlandırdığımız noktalar çoğunlukla engeller tarafından oluşturulmuş dar geçitlerin giriş ve çıkış bölgelerinde görülecektir. Bu yaklaşım, bu karmaşık büyük yörünge planlama problemini küçük seri yörünge planlama problemlerine çevirme kolaylığı sağlayacaktır.

Bu ilk adımdan sonra, Dimanik-Uygulanabilir Yörünge üretebilmek için iki ayrı yöntem önermekteyiz. Mod-Tabanlı PRM Planlayıcı olarak adlandırdığımız bu ilk yapı, manevraları ayrık modlarla tanımlanabilecek atak insansız hava araçları için uygun olan bir yöntemdir. Bu mod-tabanlı yapı sistematik uçuş kontrol tasarımı için de oldukça uygun bir yaklaşımdır. Bu yaklaşım, a) temel atak-manevra uçuş modlarının kontrolü ve b) nerdeyse tüm atak manevraların yaratılabileceği uçuş modlarını kullanarak yörünge planlama problemlerinin birleştirilmesine dayanır. Bu türden kontol ve kapalı-çevrim modlarını kullanarak yörünge planlama yaklaşımı sonucunda önemli ölçüde kontrol değişkenleri uzayı boyutu, dolayısıyla taranılan uzayın boyutu küçülmektedir. Daha önceden elde edilmiş bağlantı yörüngesini oluşturan noktalar arasında, bu ayrık uçuş modu seçimlerini kullanarak, dinamik-yapılabilir uçuş yörüngeleri üretebilmek için Tek-sorgulu Olasılıksal Yol Haritası (PRM) aramaları kullanılır. Burada dar geçit problemine temas etmek için, tekdüze-olmayan dağıtık yaklaşma alanları ile yaklaşılan hedef noktasından daha sonra gelecek hedef noktasına yönelmesini sağlayan araç durum değişkenlerini seçen bir yöntem izlenecektir.

Olasılıksal B-Spline Planlayıcı olarak adlandırdığımız ikinci yörünge planlama yaklaşımı ise ilk adımda elde edilen yol noktalarından geçen sabit ivmeli uçuş yörüngeleri üretmek için daha uygun bir yöntemdir. Bu zaman boyutu değiştirilebilir sabit ivmeli uçuş yörüngeleri, kapalı çevrim dinamiğe sahip insansız

bir helikopterin yörünge kontrolüne benzemektedir. Bu yaklaşımda, bağlantı yörüngesi üzerinde bulunan her bir yol noktası B-Spline yörünge eğrisi ile bağlanır ve bu eğriler yerel-etki özellikleri sayesinde bazı olasılıksal onarma yöntemleri ile dinamik-yapılabilir yörüngeler elde edebilmek için yeniden düzenlenir.

Yaptığımız 2B ve 3B çevrelerdeki benzetimler, bu yöntemlerin karmaşık ve yoğun geometrili çevreler için gerçek-zamanlı uygulabilir olduklarını göstermektedir.

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.

It is noted in [1] that 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. For example [1] suggested 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.

In this work, we suggest a real-time implementable two-step planner strategy. Our general motion planning strategy, as a first step, is aimed at finding a connectivity path between the initial and the goal point. Since, this phase needs to add minimal computational time; it should be rapid and have the ability to give a quick solution to the connectivity problem. Therefore, in this phase, RRT algorithm is used because of its well quick spreading ability. RRT algorithm can

rapidly search the space in a short time because of its well extension property. To decrease the computational time, we disregard the vehicle's dynamic constraints and use RRT for searching only the configuration space of vehicle. After this first phase connectivity path is refined with tiny line-of-sight transition algorithm to filter long detours. Remaining points that we will call as way points generally appear in entering and exiting regions of the narrow passages that are formed between the obstacles. This gives an advantage to segment the hard path planning problem to a series of small path generating problems on these locations.

After this first step 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 flight control system. Our second feasible path method that we called *Probabilistic B-Spline Planner* is more suitable for generating constant acceleration flight paths that pass through milestones are founded in the first step. These time-scalable constant acceleration flight paths approximate unmanned helicopter closed-loop dynamics under trajectory control.

For the our first *Mode Based PRM Planner* approach, a dynamically feasible path between the way-points (and their possible neighborhood relaxations) is created with a single-query probabilistic roadmap 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. All process of this method is illustrated in Fig. 1.1.

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,

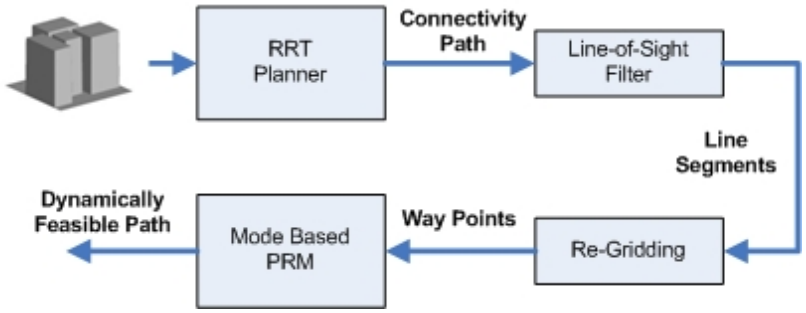

Figure 1.1: Dynamically Feasible Path Process Creation with Mode Based PRM Planner

the size of the search is limited to only partial flight segments. Both of these factors contribute significantly in reduced computation time.

In the our *Probabilisitic B-Spline Planner* approach, B-Spline method is used for generating constant acceleration flight paths that pass through these milestones. In face of collisions, the milestones locations are probabilistically adjusted to eliminate the collisions and the accelerations are limited in time scale to allow dynamic achievability. These steps are illustrated in Fig. 1.2

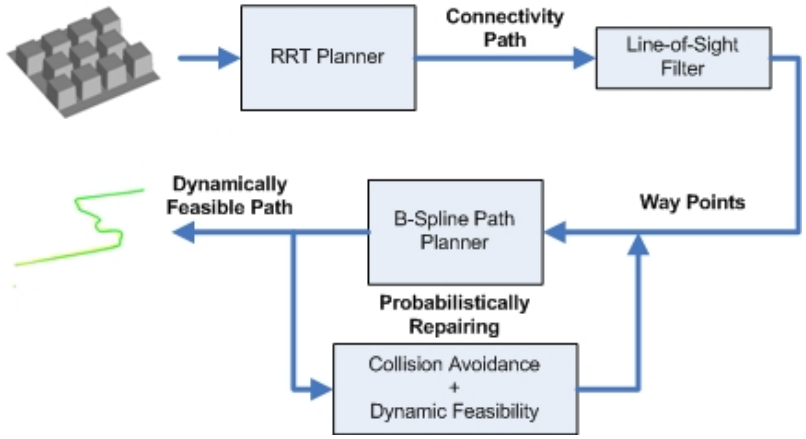

Figure 1.2: Creating Dynamically Feasible Path Process with Probabilistically B-Spline Planner

In comparison, our this method utilizes both the probabilistic and the deterministic aspect of both of these approaches to obtain a real-time implementable two-step planner strategy. In the creating dynamically feasible path step of our this strategy, for every consecutive way points, third-order B-Spline curves are generated deterministically and checked for collision and dynamic feasibility between them. These third-order B-spline curves present constant inertial acceleration paths with breakpoints. If the generated curve

is not feasible, the probabilistic repairing is achieved by randomized waypoint expansion on the connecting line path and the unit flight time is expanded to limit the accelerations within controllable regime. Since B-Spline curves have local support property, these repairing processes can be made on local interest of path.

On this thesis, firstly we gave related key works about sampling-based motion planning in Chapter 2. On Chapter 3, we focus on dealing with complex environment problem, firstly key works are given and our finding connectivity path contribution is explained. On Chapter 4, Our two creating dynamically feasible path method is described respectively. ITUCAL Testbed is presented on Chapter 5. and the conclusions are discussed in Conclusion section. Detailed informations about B-Spline Curves and Mode-Based Maneuvering can be found in Appendixes section to facilitate understanding behaviors of planners that we presented.

2. SAMPLING BASED MOTION PLANNING FRAMEWORK

For developing 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 aerial vehicles.

On this chapter, we firstly gave the previous related key works to understand the sampling based motion planning phenomenon.

2.1 Roadmap-Based Planners

Roadmap-based planners are typically used as multi-query planners that generally have two phases; a learning phase and a querying phase. In the learning phase, a roadmap is grown incrementally with sample points in the space until the roadmap has reached a sufficient level of coverage. In the query phase, planner try to connect the initial point and the goal point to nodes that exist in the roadmap. If there is a path that connect the initial point to the goal point through the roadmap, then a solution is returned.

Well known Probabilistic Roadmap (PRM) planner [2] is a good representation of roadmap based planners. Similarly, the main idea of the PRM is constructing a roadmap that is an undirected graph $G = (V, E)$ in probabilistic way. The nodes in V are set of vehicle configurations (or robot) chosen by some sampling strategies in C_{free} as illustrated in Fig 2.1. The edges in E represents a collision-free paths that connect nodes locally in roadmap. In primitive form of PRM, local planner connects the nodes by straight lines. Roadmap construction process can be seen in Algorithm 2.1 is shown below.

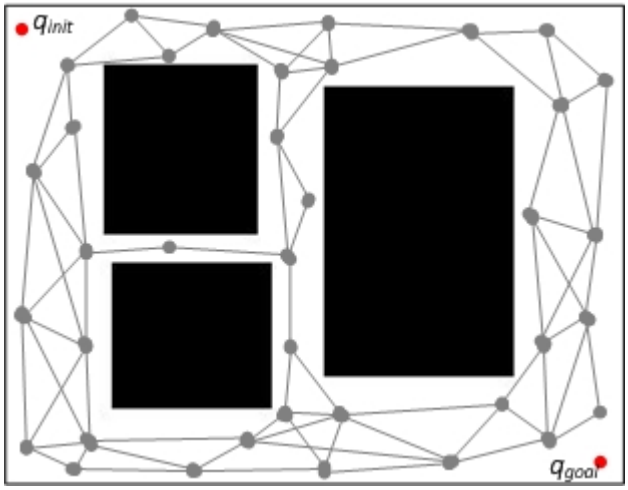

Figure 2.1: Construction of a roadmap with PRM Planner

Algorithm 2.1: Build Roadmap

$V \leftarrow \{\}, E \leftarrow \{\}$
while $|V| < n$ **do**
  $q \leftarrow$ a sampled configuration in C_{free}
  $V \leftarrow V \cup \{q\}$
$N_q \leftarrow$ a set of closest neighbors of q chosen from V
forall $q' \in N_q$ **do**
  $E \leftarrow E \cup \{(q, q')\}$

In query phase, PRM planner first tries to connect initial configuration q_{init} and goal configuration q_{goal} to two member nodes in the roadmap. If q_{init} and q_{init} can be connect to any q and q' respectively, then sequence of edges are searched in E that obtain connection between q and q' as illustrated in Fig 2.2. There are several versions of the PRM, but they all use the same underlying concepts. Query phases steps can be seen in Algorithm 2.2.

There is two important challenges of this method which first one is how to sample configurations that will increase the coverage of the roadmap and second one is how to connect nodes in the roadmap [3]. Therefore, new strategies are developed to deal with these problems. We will give detailed explanations about the improvements to deal with first problem.

PRM Planer locally tries connect nodes and checks collisions of edges between them. Actually, most of the connection edges are not used in result path. Therefore, in Lazy-PRM algorithm [4] checks collisions of edges only if they need in evaluation of solution path. This approach minimize the number of

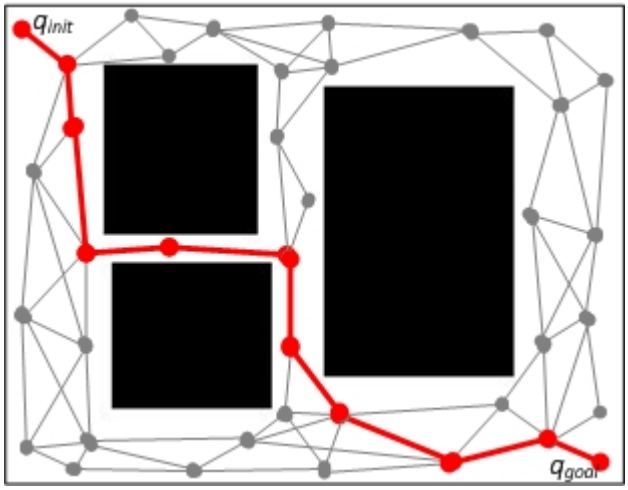

Figure 2.2: Query Phase Strategy of PRM

collision checking and hence significantly decreases running time of the planner. If a collision with the obstacles occurs, the corresponding nodes and edges are removed from the roadmap and planner tries to find new collision-free path. This approach is illustrated in Fig. 2.3.

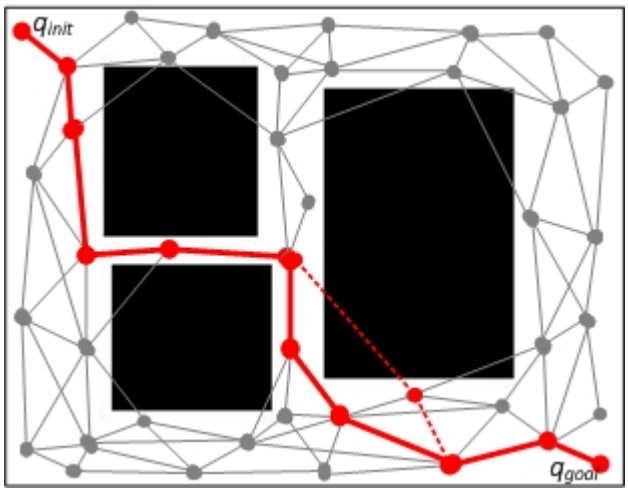

Figure 2.3: Repairing Strategy for Lazy-PRM planner

Solution paths obtained with a PRM planner are typically jagged and quite long [3]. Therefore, some smoothing processes can be applied to solution path. In [5], connection strategy allows cycles in the roadmap to shorten the paths between the configurations.

2.2 Tree-Based Planners

In these planners, main data set is a typically a tree. Tree-based planners focus on to solve a query as fast as possible. The basic idea is that an

Algorithm 2.2: Query Phase

$N_{q_{init}} \leftarrow$ a set of closest neighbors of q_{init} chosen from V
$N_{q_{goal}} \leftarrow$ a set of closest neighbors of q_{goal} chosen from V
$V \leftarrow \{q_{init}\} \cup \{q_{goal}\} \cup V$
set q' is the closest neighbour of q_{init} in $N_{q_{init}}$
repeat
  if $(q_{init}, q') \subset C_{free}$ **then**
    $E \leftarrow \{(q_{init}, q')\} \cup E$
  else
    set q' is the next closest neighbor of $\{q_{init}\}$ in $N_{q_{init}}$
until *a connection is found*
set q' is the closest neighbour of q_{goal} in $N_{q_{goal}}$
repeat
  if (q_{goal}, q') *is in* C_{free} **then**
    $E \leftarrow \{(q_{goal}, q')\} \cup E$
  else
    set q' is the next closest neighbor of $\{q_{goal}\}$ in $N_{q_{goal}}$
until *a connection is found*
$Path \leftarrow ShortestPath(q_{init}, q_{goal}, G)$

initial samples (start or goal configurations) are the root of the tree and newly generated samples are connected to nodes already existing in the tree. Most popular representatives of tree-based planners are Expansive-Spaces Trees (ESTs), Single-Query Bi-Directional PRM with Lazy Collision Checking (SBL) and Rapidly-Exploring Random Trees (RRTs).

2.2.1 Expansive-Space Trees (EST)

Expansive Space Trees [6] (EST acronym was later begun to use) were initially developed as a single-query planner that covers the space rapidly. In this method, planner first selects a configuration q existing in main tree T and then samples a random configuration q_{rand} near by q. If planner can connect the q_{rand} and q with a collision free edge, then the q_{rand} is added to vertices of T and (q,q_{rand}) is added to the edges of T. The pseudo code of EST given in Algorithm 2.3 shown below.

The good performance of EST relies on the ability to avoid oversampling any region of C_{free}. Therefore, choosing configuration from T is made with probability function $\pi_T(q)$. $\pi_T(q)$ function is biased toward configurations whose neighborhoods are not dense. To measure the density of neighborhood, each

Algorithm 2.3: Build EST Algorithm

$V \leftarrow \{q_{init}\}, E \leftarrow \{\}$
repeat
  $q \leftarrow$ a sampled configuration in T with probability $\pi_T(q)$
  $q_{new} \leftarrow$ a random collision free configuration from neighborhood of q
  if *(q,q_{rand}) in* C_{free} **then**
    $V \leftarrow V \cup \{q_{new}\}$
    $E \leftarrow E \cup \{(q, q_{new})\}$
until *connection is found between* q_{init} *and* q_{goal}

configurations q in the T assigned with weight $w_T(q)$ that counts the number of configuration within neighborhood of q. For example, $\pi_T(q)$ can be defined to be inversely proportional to $w_T(q)$.

2.2.2 Single-Query Bi-Directional PRM with Lazy Collision Checking (SBL)

SBL [7] generates two EST trees T_{init} and T_{goal} begin from q_{init} and q_{goal} respectively. SBL generates new nodes similar to EST but does not immediately check collisions between the nodes. The connections is only checked when it will be part of the solution path that connects the two T_{init} and T_{goal} trees. This approach results in decreasing on run time.

2.2.3 Rapidly-Exploring Random Trees(RRT)

The RRT algorithm works by extending a tree from a given root. In each step of algorithm, RRT planner uses two step; selection and propagation. In selection step, a random configuration q_{rand} is sampled in C_{free}. Then, the nearest configuration q_{near} to q_{rand} is found in existing tree T. In propagation step, q_{near} is extending toward q_{rand} using with specific strategy. If this newly generated configuration q_{new} extended from q_{rand} and their edge (q_{near},q_{new}) is in collision free, then q_{new} is added to the vertices of T and (q_{near},q_{new}) is added to edges of T. The pseudocode of the RRT algorithm can be seen in simple Algorithm 2.4. Extension strategy of the RRT algorithm is illustrated in Fig. 2.4.

One of the bottlenecks of RRT method, in environments like a bug-trap, most of the randomly selected samples will fall out of the trap and it will cause that RRT algorithm mostly returns with fail as presented in [8]. To overcome this

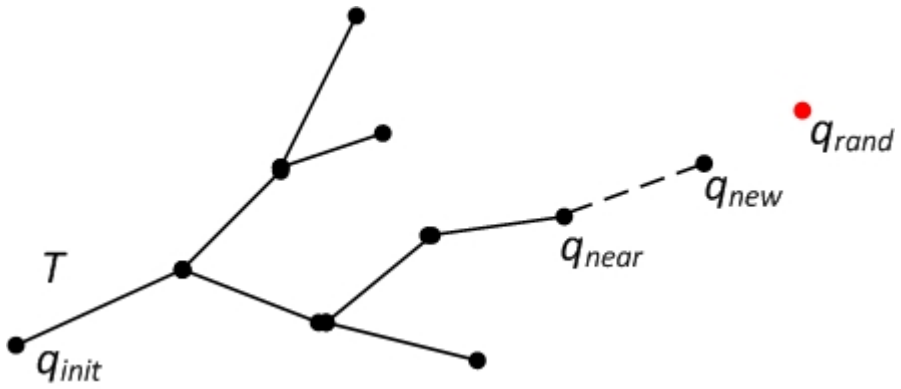

Figure 2.4: Extension Strategy of RRT Tree

Algorithm 2.4: Built RRT Algorithm

$V \leftarrow \{q_{init}\}, E \leftarrow \{\}$
repeat
  $q_{rand} \leftarrow$ a randomly chosen collision free configuration
  $q_{near} \leftarrow$ closest neighbor of q_{rand} in T
  $q_{new} \leftarrow Extend(q_{near})$ toward q_{rand}
  if (q_{near}, q_{new}) *in* C_{free} **then**
    $V \leftarrow V \cup \{q_{new}\}$
    $E \leftarrow E \cup \{(q_{near}, q_{new})\}$
until *connection is found between* q_{init} *and* q_{goal}

problem, every nodes existing in the tree is assigned with a radius. If randomly selected configuration is further away than the specified radius, another sample is selected until the the configuration fall in the radius attached to its nearest node. These radius values are set from workspace-dependent constant. To adapt the value of the radius according to some other constant that less sensitive to the workspace, in [9], the radius is increased when every successful connection is done and decreased when every connection gives a failure.

2.2.4 Using Multiple Tree

Single-query planners can be processed with multiple trees. It gives a potential parallel execution to planners [3]. For example, in commonly used bi-directional search, one of the tree is propagated from initial configuration while other one is propagated from goal configuration toward the each other as seen in [10]. In detailed, for bi-directional RRT, planner first tries to extend one of the tree toward to q_{rand} and new q_{new} configuration is obtained. Then, the closest configuration of q'_{near} is selected from other tree and it is extended toward q_{new} with q'_{new}. Until

the certain connection is found, this algorithm is repeated with swapping the two tree. This method is illustrated in Fig. 2.5.

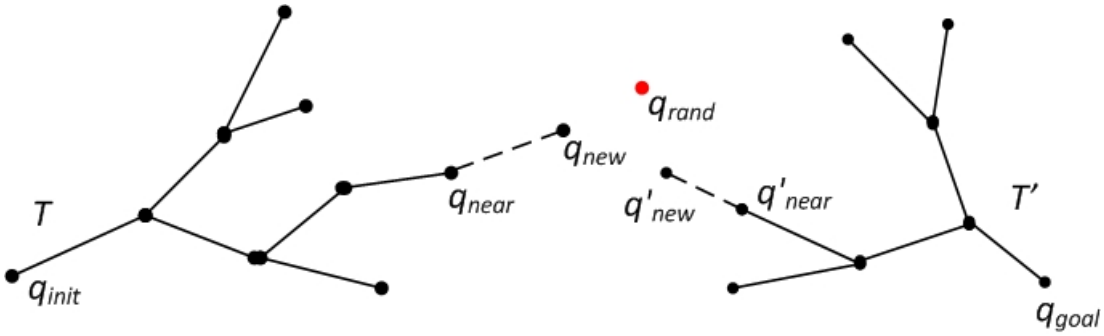

Figure 2.5: Building Bi-directional Search for RRT trees

2.3 Adapting the Planning Algorithms to Kinodynamic Motion Planning Problems

In the path planning problem, first intention is to compute a collision free path ignoring the system dynamics. In side of controlling perspective, controllers should generate appropriate control signals that will implement desired path. However, produced solution path can not be execute by controller and it may be infeasible for the controller. Therefore, sampling-based motion planners, especially tree-based planners, are adapted to solve path planning problem considering the kinodynamic and physical constraints in the last few years. The main idea behind kinodynamic sampling-based motion planning, is to search a state-space χ of the vehicle [3]. A state of the vehicle is simply defined $x = (q, \dot{q})$, for given a configuration $q \in C$. In kinodynamic planing, the goal idea is to plan with the tree-based planners in the state space instead of the configuration space.

2.3.1 State-Space Formulation

For detailed explanation of state space, we consider a vehicle whose motion is explained by an equation of the form:

$$\dot{x} = f(x,u), \tag{2.1}$$

where $x \in \chi$ is the vehicle's state, $\dot{x}$ is its derivative relative to time and $u \in \Omega$ is a control input. The set χ and Ω are the vehicle's state-space and control-space

respectively. We assume that χ and Ω are bounded of dimension n and m that is $(m \leq n)$ respectively. Eq.(2.1) can represent both nonholonomic and dynamic constraints. A nonholonomic system is constraint by k independent non-integrable scalar equations of the form $F_i(q,\dot{q}) = 0, i = 1,2,...,k$, where q and $\dot{q}$ represents the vehicle configuration and velocity respectively. Using the Lagragian Mechanics, the dynamics equations can be represent by a set of equations of the form $G_i = (q,\dot{q},\ddot{q})$. Let $x = (q,\dot{q})$ defines as vehicle's state, we can rewrite the dynamic equations of the form $F_i(x,\dot{x}) = 0$ for $i = 1,...,m$ which as in the nonholonomic case [11].

2.3.2 Obstacles in the State Space

There are interesting differences between finding collision-free paths in C versus in the state-space χ [12]. The path planning problem is to find a continuous path into $C_{free} = C \backslash C_{obst}$ where C_{obst} represents configurations within regions are blocked by obstacles. For planning in the χ, χ_{obst} can be simply defined by declaring $x \in \chi_{obst}$ if and only if $q \in C_{obst}$ for $x = (q,\dot{q})$ but another interesting phenomena exist: *the region of inevitable collision* that its detailed explanation is given in [12]. It can be defined as χ_{ric} represents the set of states in which the vehicle is either in collision or it can not do anything to avoid collision. In other words, there is no control inputs over any time interval that can be supply to avoid collision for the states lie in the χ_{ric}. Therefore, it can be choose to define $\chi_{free} = \chi \backslash \chi_{ric}$. This phenomena explains part of challenge of the kinodynamic planning problem.

2.3.3 Planning under Kinodynamic Constraints

To adapt the planning under kinodynamic constraints that explained above, motion planners need some modifications.

In RRT like planner, with similar method is previously described, planner samples random control inputs and tries to apply them for some amount of time to extent current state toward to newly random selected state [12]. The control input u can be chosen at random or be selected by trying all possible inputs in time interval Δt and choosing the one that can extend the current state toward sampled state

as close as possible as illustrated in Fig. 2.6. Therefore, any path on the tree is a collision free and feasible path of the vehicle.

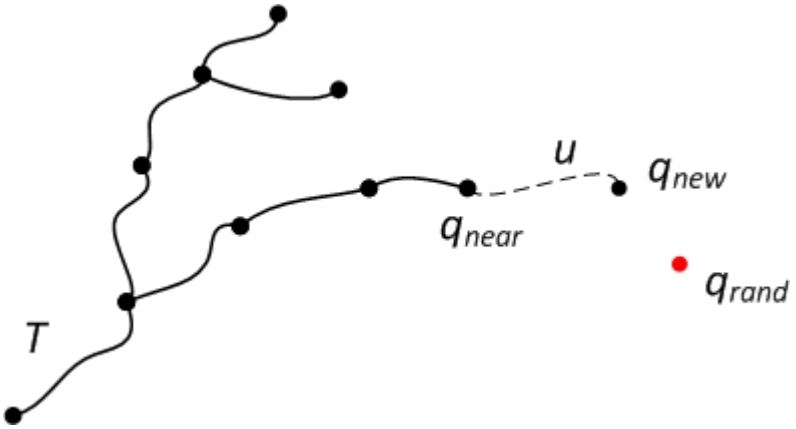

Figure 2.6: Kinodynamically Extension of RRT Tree

In the similar way, EST planner is modified to plan in *state x time space* as presented in [13]. The planner picks a state node already existing in tree and tries to extend it with applying control input that sampled for some fixed amount time interval at random as illustrated in Fig 2.7. Moreover, in [13], trajectory replanning is discussed if unexpected changes are occur in the environment that conflicts with the current trajectory.

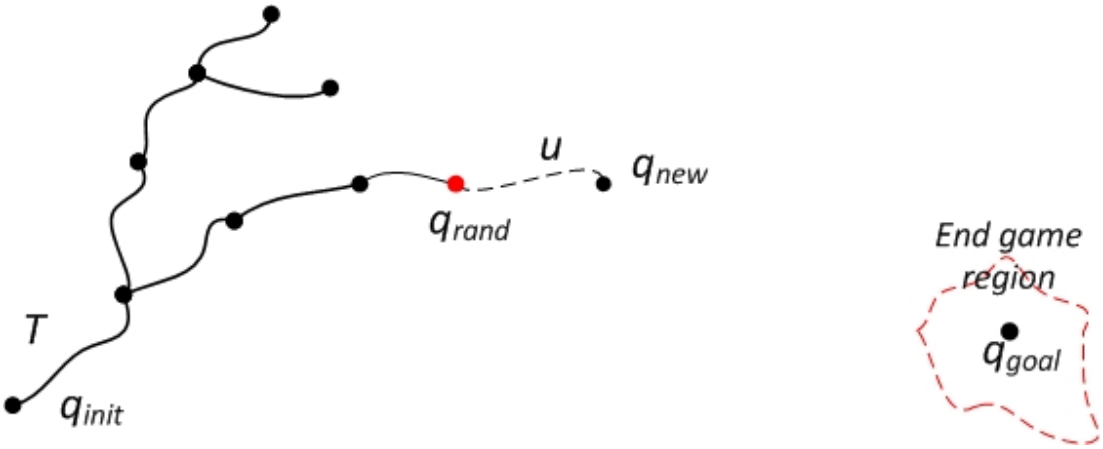

Figure 2.7: Kinodynamic Extension Strategy of EST tree

In [13], one issue that called *endgame connection* is emphasized. The kinodynamic planning does not allow reach to goal state exactly. Therefore, goal state should be expanded to the *endgame region*. Some ways are suggested in [13] to create such an endgame region.

On the following chapter, we will discuss the dealing with complex environment issue of the sampling based planner. Firstly, key works about improvements of

sampling based path planners for complex environments will be given and then our suggestion will be presented to deal with this problem.

3. DEALING WITH COMPLEX ENVIRONMENT PROBLEM: FINDING CONNECTIVITY PATH IN COMPLEX 3D ENVIRONMENTS

The motion planning problem is significantly more challenging when planners must be run in the complex environments that have difficult and narrow passages. Therefore, some works have been recently focus on improving the performance of the motion planners in complex environments.

On this chapter, we firstly gave the key works on improving effectiveness of the motion planners in difficult regions and then we present the our strategy that increase the running performance in the complex environments for the further steps of the our planners.

3.1 Improvements for Dealing with Complex Environments

Choosing suitable sampling strategy significantly effects the performance of the motion planners [14]. Therefore recent works focused on improving sampling strategy with using workspace information to able to cover important areas such as narrow passages.

To increase the chances to finding solution path in complex environments, as this thesis focused on, some sampling strategies try to increase sampling density around narrow passages.

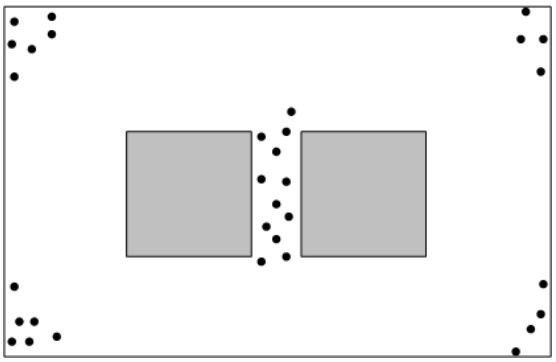

Figure 3.1: Bridge-test Sampling Strategy

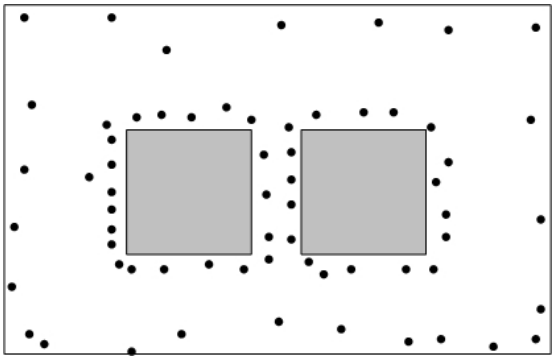

Figure 3.2: Gaussian Sampling Strategy

For example, the bridge-test method presented in [15] uses information of colliding samples. If sampled two points x and x' are in collision and their midpoint x_m is not in collision, it is added to roadmap. Hence, sampling density significantly rises in passages as seen in Fig. 3.1. Some sampling methods tries to increase sampling density near the obstacles. In [16], Gaussian sampling strategy is used to sampling biased near the obstacles as illustrated in Fig. 3.2 from information that is obtained by initial uniform sampling. In [17], similarly, ***OBPRM*** generates uniform samples and then for each configuration q_{in} found in collision, planner generates a random direction v and finds a collision free configuration q_{out} in the direction v. Finally, planner searches to find free configuration q closest to the surface of the obstacle and q is added to the roadmap. Some hybrid sampling strategies are used to select the most effective sampler according to sampled region [18].

Some strategies tries to divide space into regions according to initial uniform sampling information [19]. According to density of occurring collision in regions, regions are defined as $free$, $narrow$, $surface$, $blocked$, etc. The region specific samplers are then used to further sampling.

Increasing number of nodes in the roadmap also increases solution time in query phase, therefore, some strategies tries to decrease dimension of the roadmap. Visibility-based PRM [20] method tries to keep small the number of nodes in produced roadmap with visibility phenomena. Visibility-based PRM planner only adds nodes to roadmap within sampled free configuration set if these can not be connected to any existing node (out of visibility) or can be connected at least two existing nodes. This approach can be seen in Fig. 3.3.

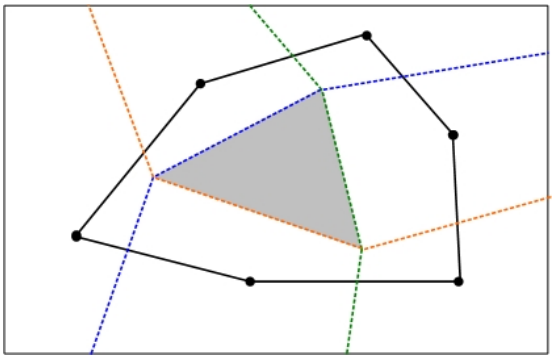

Figure 3.3: Visibility-Based Sampling Strategy

Some gradual motion planning methods that have resemblances to our strategy are recently proposed to solve complex problems such as planning on cluttered environment. These methods based on firstly solving relaxed form of the problem and then approximate solution is refined to solve original problem with some "repairing" methods. In [21], roadmap is initially generated by allowing some penetration. Later, "milestones" are carried to collision-free space. In Iterative Relaxation of Constraints (IRC) method [22], firstly relaxed version of problem is solved and then this coarse solution is used as a guide to solve original problem iteratively. Finding approximate solution phenomena is also seen in Lazy PRM [23], C-PRM [24], and Fuzzy-PRM [25].

3.2 Finding Connectivity Path in Complex 3D Environments

In motion planning problem for complex environments, first major question is whether there is solution path or not. Planning algorithm in generally try to find solution path in predefined number of iterations. It is hard to say when planners should stop searching or change the searching method (i.e. switching to a more complex planner etc.) or begin repairing algorithms. Especially in real time applications, planners should be able give a reliable answer in minimal permitted time slot. Also, finding an obstacle free geometrical path does not necessarily mean that a dynamically feasible path exists - though this is the case for helicopters that can do point to point navigation with hover. For vehicles that have high dimensional state-space, for example a aerial vehicle, searching in high dimensional state-space consumes long computational time to find the feasible path.

In our strategy, for finding connectivity path, RRT algorithm is used because of their rapid spreading ability as the first step. RRT is a considered as being an efficient algorithm to search even high dimensional spaces that has both global and differential constraints [26]. However, one of the important drawbacks of using RRT as a stand-alone planner is biasing of the distribution of milestones toward the obstacle regions if the configuration space has large obstacles. In this phase, we are only motivated by RRT's good property to obtain connectivity path that can be tracked during flight. Our strategy does not focus on feasibility in this part of the path planner. Therefore, RRT algorithm is used searching configuration-space C of vehicle with primitive maneuvers such as level and climbing flight and changing instantaneous heading direction. Construction of connectivity path algorithm is given in Algorithm 3.1 and illustrated in Fig. 3.4.

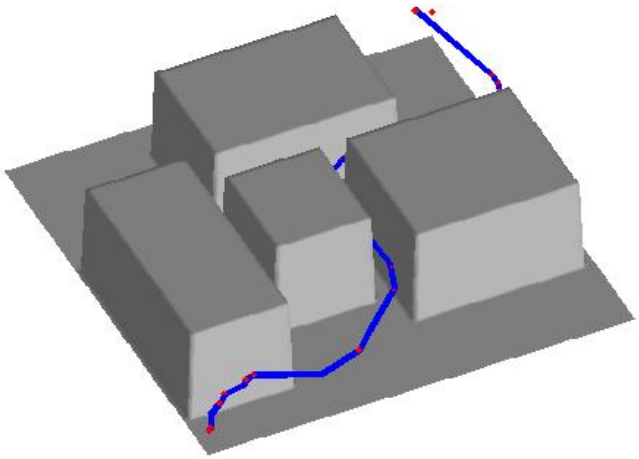

Figure 3.4: Construction of Connectivity Path

In this algorithm, our strategy resembles a bidirectional RRT-based planner but, in our algorithm only 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 select points, with specialized selection method of RRT, the nearest point already within the τ tree to the sampled random point (in Line 4) and the nearest point to the goal point is selected (in Line 11) respectively. *Generate* function generates new point m_{new} on the direction of the selected nearest points m_{near} at random selected

Algorithm 3.1: Goal Biased RRT Algorithm

input : q_{init}, q_{goal}
output: *connectivity path*

```
1  τ ← q_init and i ← 1
2  repeat
3      Select random point m_rand in C
4      Select neighbor point m_near of m_rand in τ
5      Generate m_new from m_near toward m_rand
6      Generate e_new from m_near to m_new
7      if e_new is in C_free then
8          τ ← m_new and i + 1
9          if m_new is in end region then
10             break with success
11     Select neighbor point m_near of q_goal in τ
12     Generate m_new from m_near toward q_goal
13     Generate e_new from m_near to m_new
14     if e_new is in C_free then
15         τ ← m_new and i + 1
16         if m_new is 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 can be gone back from end region to initial point in τ
```

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 vehicles 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 τ tree as shown in Line 8 and 15. If τ 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 [13].

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 "open way" in this part; we can simply remove the points that cause these detours. Therefore, we perform a simple *filter algorithm* that uses the line-of-sight implementation after finding the *connectivity path*. As can be seen in Fig. 3.5, the key milestones frequently appear in while entering and exiting field of narrow passages and hard regions. Hence, these *guard points* also

indicate where hard regions are beginning and the direction of the passage. This simple phase is illustrated in Algorithm 3.2.

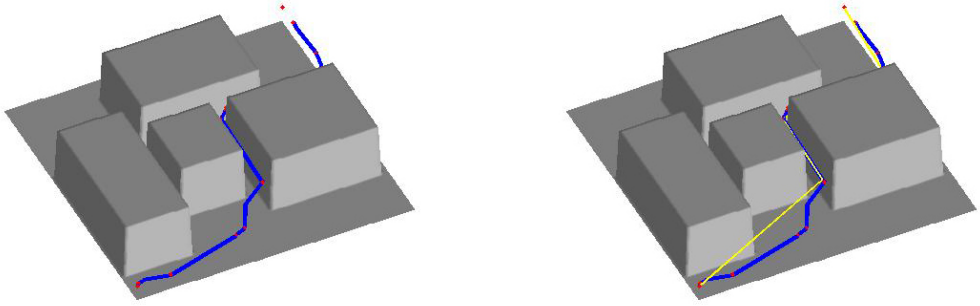

Figure 3.5: 3D Demonstration of Line-of-Sight Filter

Algorithm 3.2: Line-of-Sight Filtering

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

In this phase of the algorithm, a simple iteration checks whether selected point can be connected with previous points in *connectivity path* with a line segment without colliding with any obstacles. 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 subsequently the search continues from this point on. Search process runs until the last point of *connectivity path* is reached.

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, especially with a PRM further 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

predescribed threshold; *long*. This term is defined as distance metric value and is chosen according to density of environment. This Re-Gridding can be seen in Algorithm 3.3.

Algorithm 3.3: Re-Gridding

input : *visible points*
output: *way points*
1 *way points*$\leftarrow$ *visible points*
2 **for** $m_i \leftarrow m_2$ **to** *last way point* m_m **do**
3 **if** *distance from* m_{i-1} *to* m_i *is long* **then**
4 *way points* $\leftarrow$ *middle point* of m_{i-1} and m_i

Note that, this phase does not be needed in the following Probabilistic B-Spline Planner while may be needed in the Mode-Based PRM Planner.

On the following chapter, we will suggest the two approach to create dynamically feasible trajectories between these way points obtained in "finding connectivity path" phase that presented in this chapter. We will call the first one as *Mode Based PRM Planner* and the second one will be called as *Probabilistic B-Spline Planner*.

4. CREATING A DYNAMICALLY FEASIBLE PATH IN COMPLEX 3D ENVIRONMENTS

After the finding connectivity path, 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 flight control system. Our second feasible path method that we called *Probabilistic B-Spline Planner* is more suitable for generating constant acceleration flight paths that pass through milestones are founded in the first step. These time-scalable constant acceleration flight paths approximate unmanned helicopter closed-loop dynamics under trajectory control. We presented these two approaches on the following sections respectively.

4.1 Dynamically Feasible Path with Mode-Based PRM Algorithm

Our Mode-Based PRM approach [27] 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 [1]

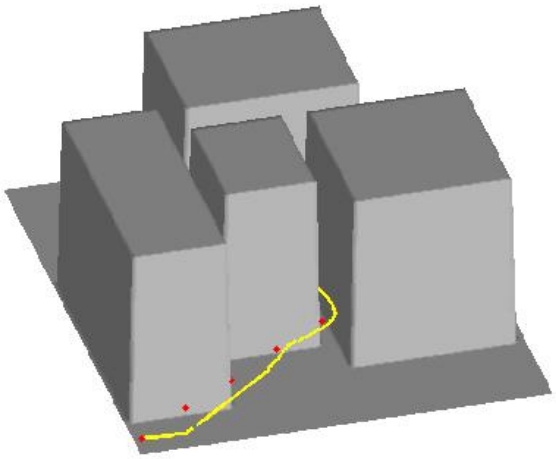

Figure 4.1: Complete Solution of Mode-Based PRM

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.

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 [28]. Note that this modal approach can also be used for trajectory optimization for multiple vehicles [29]. [1] 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

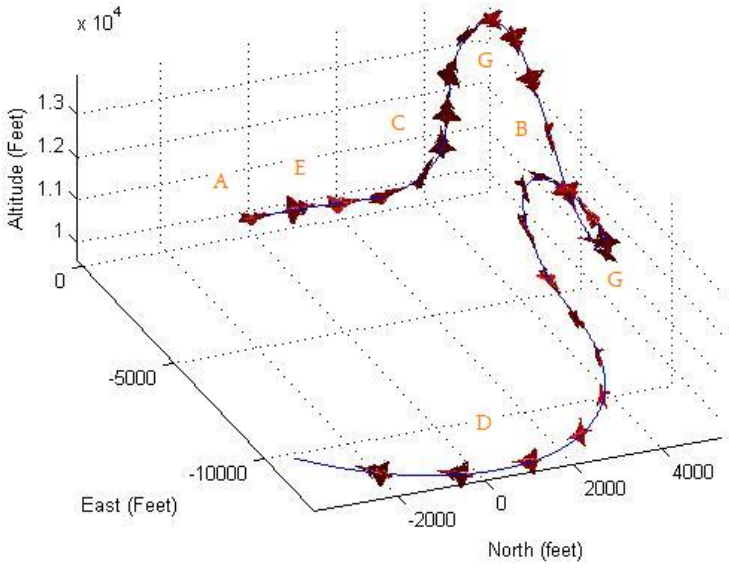

Figure 4.2: Nonlinear Control of an Aggressive Maneuvering of F-16 over Flight Modes Using the Full-Envelope Dynamic Model

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 [30]. 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. 4.2. Detailed description about the mode-based maneuvering phenomena will be given in Appendix A.

Previous chapter 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 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.

Algorithm 4.1: 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 ← 0
25 until last way point is reached
```

This part of our planning strategy is an extension of single-query PRM algorithm that given as Algorithm 4.1. 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.

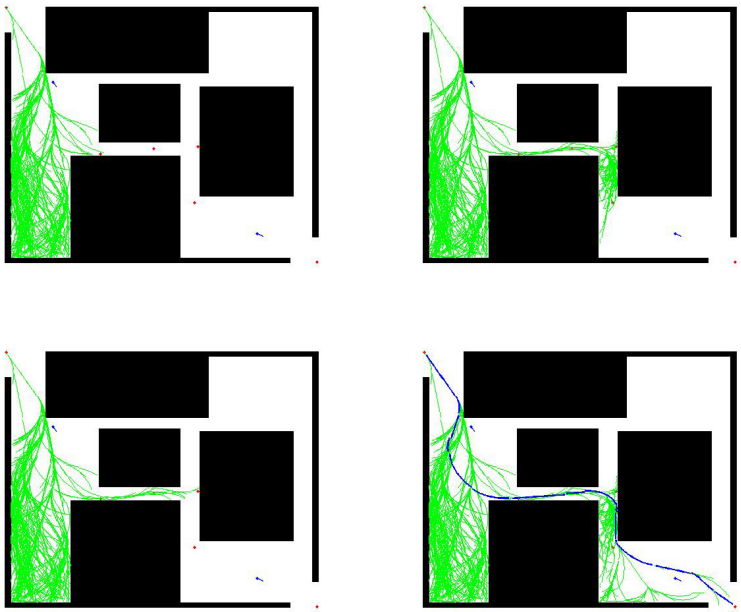

Figure 4.3: 2D Demonstration of Mode-Based PRM Searching

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. 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.

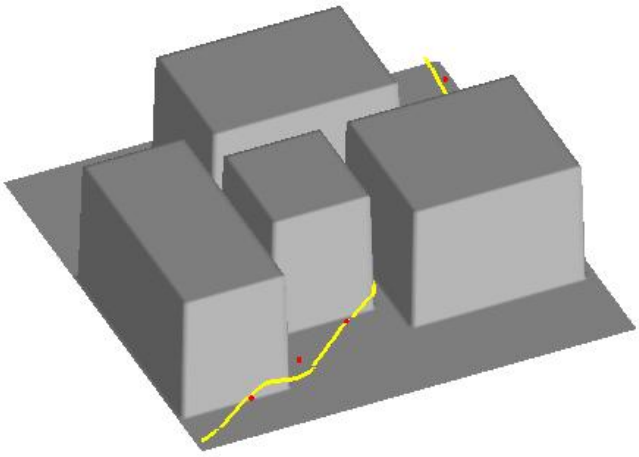

Figure 4.4: Complete Solution of Mode Based PRM Planner

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. 4.3.

Simulation Results

We tested the performance of our method on some dense environments in varying ratio of obstacle space to all work space. The computational times of the all phases of the algorithm are illustrated in Table 4.1 for 3D single-narrow-passage problem, city-like environment, mostly-blocked environment and challenging circuitous-tunnel problem as all seen in Fig. 4.8. All the experiments were conducted on a 3.00 GHz Intel Pentium(R) 4 processor and the average results are obtained over 10 runs.

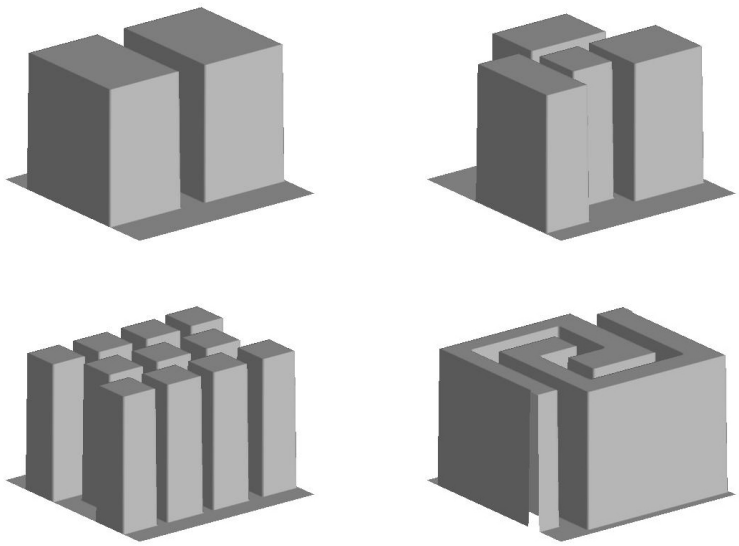

Figure 4.5: Test Environments: Single-narrow Passage, City-like Environment, Mostly Blocked Environment and Circuitous-Tunnel

As can be seen in simulation results, total times mostly based on Mode-Based Planner phase. 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.

Table 4.1: Mode-Based Path Planer Construction Times (Seconds)

		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

4.2 Dynamically Feasible Path with Probabilistic B-Spline Planning Algorithm

Our implementation of the goal-biased RRT algorithm provides a flight path that is represented with *way points.* Filtering the *connectivity path* with straight line segment results in a simple and implementable piecewise flight plan. However, this flight plan is not a fast agile and continuous motion plan – a desirable feature in many urban unmanned helicopter applications. After obtaining the *way points* on the environment, many deterministic and sampling based path planner methods can be used to find the dynamically feasible path between the *way points.* Generated paths must be able connect on the *way points* dynamically. Moreover, the paths 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 [31] and basic requirements is given in Appendix B.

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

$$C(u) = \sum_{i=0}^{n} P_i N_{i,k}(u) \quad 0 \leq u \leq u_{max} \tag{4.1}$$

The coefficients P_i in 4.1 are called control points.

B-Spline curves have been used in many dynamic path planning and control problem implementations. In [32], dynamic trajectory is generated with the minimum travel time for two-wheeled-driven type mobile robot. In [33] and

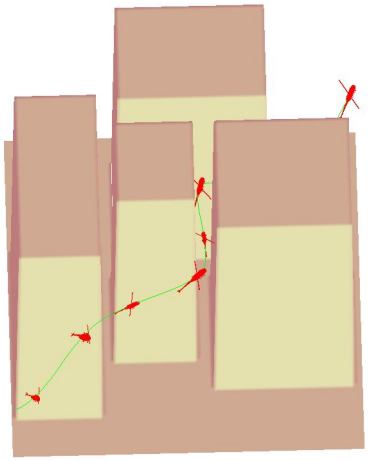

Figure 4.6: Complete Solution of B-Spline Based Algorithm for Unmanned Helicopter in City-like Environment

[34], visibility-based path is modified to continuous feasible path via B-Spline curves. Using the well local support property of B-Spline curves, real-time path modification methods are proposed for multiple mobile robots in [35] and robot manipulators in [36].

In this step of our strategy, for every consecutive way points, third-order B-Spline curves are generated deterministically and checked for collision and dynamic feasibility between them. These third-order B-spline curves present constant inertial acceleration paths with breakpoints. If the generated curve is not feasible, the probabilistic repairing is achieved by randomized waypoint expansion on the connecting line path and the unit flight time is expanded to limit the accelerations within controllable regime. Since B-Spline curves have local support property, these repairing processes can be made on local interest of path.

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 and ending direction of the curve by inserting an extra control points after starting point and before the ending point. These directions are defined according to way points' orientations that the B-Spline curve is generated between them [37].

In our path planning algorithm, we choose generate third-order path B-Splines (quadratic polynomials) to obtain constant inertial acceleration. This approximates the constant acceleration flight paths that can be tracked via a closed-loop helicopter system. For generating B-Spline path segments between consecutive way points, one control point is added after the initial way point at random selected d_1 distance on heading-direction of initial point and one control point also is added before the goal point at random selected d_2 distance on heading-direction of goal way point. Hence, B-Spline algorithm begins with four control point initially. We note that the number of control points can be incremented by the algorithm to repair the spline. For our order-three B-Spline curves, we use specific nonuniform knot vector form $\mathbf{U} = [\ 0\ \ 0\ \ 0\ \ \ldots\ \ 1\ \ 1\ \ 1\ ]$ to obtain the coincidence between the first and last control points and the first and the last points of B-Spline curves respectively. Detailed information about this effect can be found in [31]. We choose using $[0,1]$ interval for parameter u such that it represents unit-time scale [38]. This property is later used to allow acceleration feasibility via time scaling (i.e. expanding the time horizon of the maneuver) Overall B-Spline path planning algorithm can be demonstrated in Algorithm 4.1.

This algorithm tries to find dynamically feasible B-Spline curve for every consecutive *way points* in a loop which begins in Line 2 and runs until the last *way point* is connected with a feasible path. For every loop, as shown in Line 5, initial way point is selected as first control point and goal way point also is selected as end control point. To obtain the initial control polygon, second and third control points are selected as mentioned way before and depicted in Line 6 and 7. Algorithm begins with four control point, thus, $\mathbf{P}_{\mathbf{n}new}$ is null initially. Knot vector as used in our implementation is shown in Line 8. $\mathbf{u}_{\mathbf{n}new}$ vector starts initially having one member for the third-order spline and should be chosen in the unit time$[0,1]$ interval. We choose this interior knot vector uniformly spaced although the knot vector is still nonuniform. B-Spline curve is evaluated in Line 9 and then the collision and dynamic feasibility checked in Line 10.

Dynamic feasibility is checked according to first and second derivations of the evaluated B-Spline that is composed from quadratic polynomials. First derivative

Algorithm 4.2: Path Planning with B-Spline

input : **g** way point vector
output: *dynamically feasible path*

$i \leftarrow 1$
repeat
  $\mathbf{P_{new}} \leftarrow \{\}$ and $\mathbf{u_{new}} \leftarrow$ *initial* u_{new}
  repeat
    $P_0 \leftarrow g_{ith}$ and $P_m \leftarrow g_{(i+1)th}$
    Select P_1 that d_1 distance from P_0 on positive direction of $g_{(i+1)th}$
    Select P_{m-1} that d_2 distance from P_m on negative direction of $g_{(i+1)th}$
    $\mathbf{P} \leftarrow [P_0\ P_1\ \mathbf{P_{new}}\ P_{m-1}\ P_m]$ and $\mathbf{U} \leftarrow [0\ 0\ 0\ \mathbf{u_{new}}\ 1\ 1\ 1]$
    Evaluate B-Spline curve with parameter u; in interval $[0,1]$
    Check collision and feasibility
    if *collision occurs or spline is not feasible* **then**
      Change location of P_1 and P_{m-1} and $m+1$
      if $m > M$ *max iteration number* **then**
        Change location of $g_{(i+1)th}$ in its small region, $n+1$ and $m=0$
        if $n > N$ *max iteration number* **then**
          $\mathbf{P_{new}} \leftarrow P_{new}$ as new control point around collision or unfeasibility, update knot vector, $k+1$ and $n=0$
          if $k > K$ *max iteration number* **then**
            break with *fail*
    else
      $i+1$
    if *fail occurs* **then**
      Evaluate path with **Mode-Based PRM**
  until *last way point is reached with success*
until *path can be evaluated between way points*

of curves gives velocity lines and second derivative of curves gives constant accelerations. If velocities and accelerations of evaluated B-Spline curves are within interval that can be produced by the vehicle, it is considered as dynamically feasible path. Derivations of the B-Splines can be evaluated with equation seen below;

$$C(u)^{(j)} = \sum_{i=0}^{n} P_i N_{i,p}^{(j)}(u)\, 0 \leq u \leq u_{max} \tag{4.2}$$

If feasibility (both geometric feasibility and dynamic feasibility) cannot be obtained, a repairing method is implemented hierarchically.

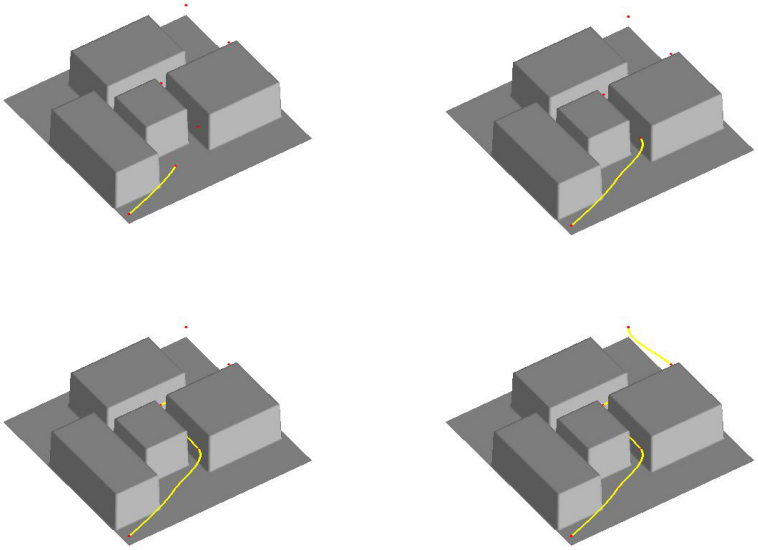

Figure 4.7: 3D Demonstration of Probabilistic B-Spline Search

For reparing, firstly, locations of P_1 and P_{m-1} control points are carried randomly on same tangent directions as shown in Line 12 and algorithm evaluates curves again in main loop. It changes shape of control polygon, hence, the beginning and the end velocity of the path and the constant accelerations are changed. After predefined number of trying, if spline can not be repaired, goal way point is carried to new collision-free location that small distance from its own location as shown in Line 14. If spline still cannot be repaired, P_{new} new control point is added in $\mathbf{P}_{\mathbf{n}new}$ vector around region that collision or unfeasibility occurred and interior knot vector $\mathbf{u}_{\mathbf{n}new}$ is updated according to number of control point. During this process if the acceleration and the velocity limits are exceeded of the helicopter, the time scale of the maneuver is expanded to allow feasibility as the time scale is arbitrary to complete the maneuver. A typical solution is shown in Fig. 4.7.

If all these repairing methods cannot achieve to find a collision free feasible path, Mode Based PRM planner is begun by algorithm. Detailed information about the Mode Based PRM planner can be found in previous section.

Simulation Results

We tested the performance of our method on some environments in varying ratio of obstacle-space to all work space and types. The computational times of the all phases of the algorithm are illustrated in Table 4.2 for 3D single-narrow-passage

Table 4.2: Probabilistic B-Spline Path Planner Construction Times (Seconds)

		Connectivity Path Planner	Filtering Phase	B-Spline Planner	Total Time
Single-Narrow Passage	time	0.432 s	0.033 s	0.551 s	1.017 s
	std	0,438 s	0.011 s	0.086 s	0.466 s
City-Like Environment	time	1.133 s	0.038 s	0.794 s	1.967 s
	std	1.171 s	0.008 s	0.121 s	1.185 s
Mostly-Blocked Environment	time	5.051 s	0.057 s	4.361 s	9.468 s
	std	3.346 s	0.012 s	6.143 s	6.584 s
Circuitous Tunnel Problem	time	120.885 s	0.079 s	1.802 s	122.741 s
	std	91.716 s	0.009 s	0.262 s	91.741 s

problem, city-like environment, mostly-blocked environment and challenging circuitous-tunnel problem as all seen in Fig. 4.8. All the experiments were conducted on a 3.00 GHz Intel Pentium(R) 4 processor and the average results are obtained over 50 runs.

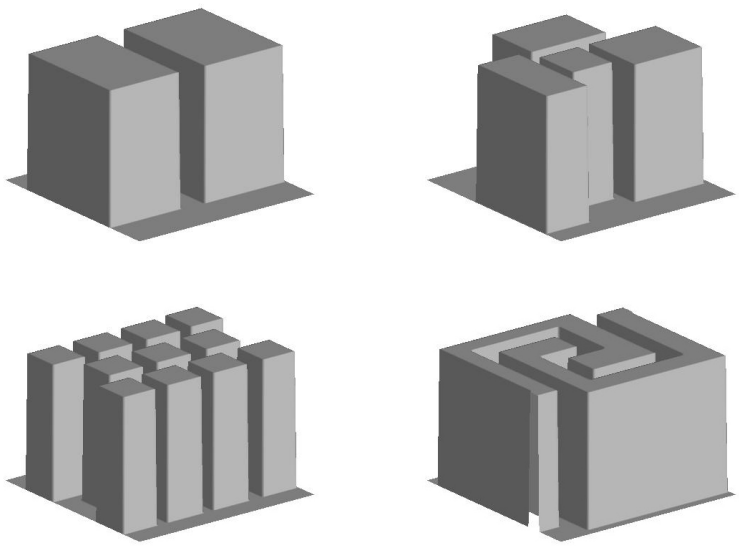

Figure 4.8: Test Environments: Single-narrow Passage, City-like Environment, Mostly Blocked Environment and Circuitous-Tunnel

Increasing complexity of the environment, as shown in 4.2, 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 environments, computational time of the B-Spline based planner phase is also rises. However, this rising rate does not grow exponentially like RRT based planner as seen in the circuitous-tunnel example. Computational times mostly based on Finding Connectivity Path Planner. The solution times

suggest that our method will be applicable for real-time implementations as the solution time is favorably comparable to implementation times.

On the next chapter, hardware and software architecture of ITUCAL Robotic Testbed will be presented to implement the motion planning algorithms in real-time and test their performance in practice with mobile robots that have no complex dynamics.

5. ITUCAL ROBOTIC TESTBED

On this chapter, we presented our robotic testbed that was build in İstanbul Technical University, Controls and Avionics Laboratory. We firstly gave the specifications, hardware and software architecture of ITUCAL Robotic Testbed. Then we explained the working steps and gave running time table of steps for the example implementations.

Figure 5.1: ITUCAL Robotic Testbed

5.1 Robotic Testbed Architecture

Our robotic testbed is formed of the 3x2.5 m movement platform for the robots. The positions, orientations and color ids of the robots are measured by an vision positioning system that is mounted on top of the room. Every robot carries two circular stickers one of them takes place center of the robot while other one takes place on heading of the robot and these stickers have own colors for every robot to define their ids. Position and heading direction is measured thanks to center

sticker and heading sticker respectively. General appearance of the testbed can be seen in Fig. 5.1.

Communications among the robots, the vision system and planner PC are implemented with radio Ethernet. All system architecture can be seen in Fig. 5.2.

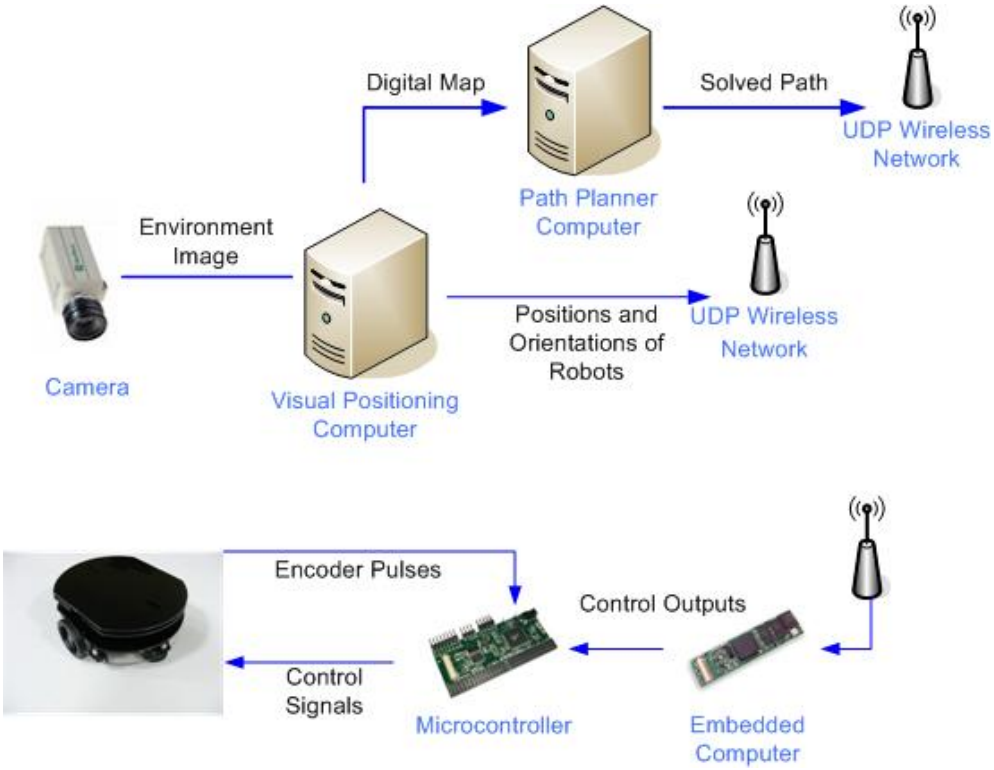

Figure 5.2: ITUCAL Robotic Testbed System Architecture

5.1.1 Visual Positioning System and Path Planning Process

Camera that mounted top of the room supplies to get positions and heading directions of the robots. Taken images are processed by Vision Positioning PC that have 2.4 GHz Pentium III processor and 2 GByte memory. To process the images, MATROX Image Library is used on it. Processed images as seen in Fig. 5.3 are sent in 7Hz to Path Planner Unit where solution paths are solved and evaluated path broadcasted through wireless UDP network.

5.1.2 Mobile Disc Robots

Testbed robots are two wheeled differential drive, circular shaped mobile robots that had been produced in ITU Controls and Avionics Laboratory. The robots

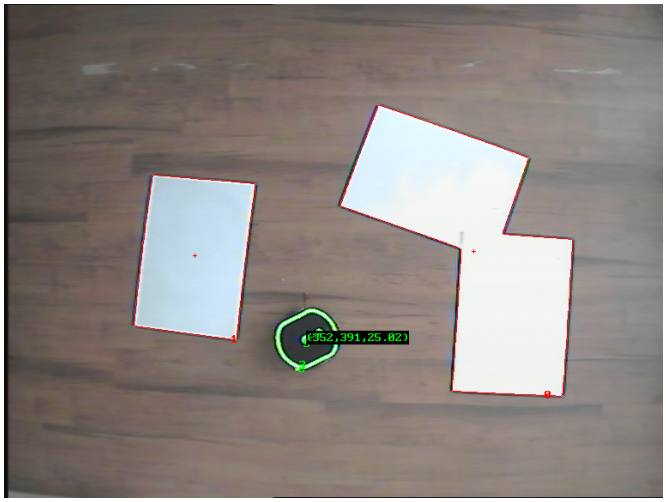

Figure 5.3: Example processed environment image, detection obstacles and robots

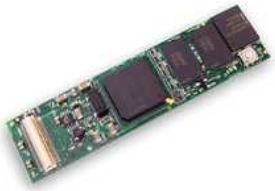

Figure 5.4: Gumstix Embedded Linux Computer of the robots

have 26 cm diameter and their wheels have 8 cm diameter. Robot controller is performed on a 400 MHz - Intel XScale® PXA255 real-time Linux computer (Gumstix) that seen in Fig. 5.4. Control signals transfered to Atmega128 microcontroller (Robostix) that seen in Fig. 5.5 to produce PWM signals. Robostix also counts optic encoder pulses for controlling angular velocities of the motors. 12V, 70 rpm motors are driven by 2A motor drivers. Their Li-Po batteries can provide power for about 30 minutes to robots without recharging.

We can define kinodynamic equations of the robots that are illustrated in Fig. 5.6:

$$\begin{aligned} \dot{x} &= \frac{R}{2}.(w_r + w_l).\cos\Phi \\ \dot{y} &= \frac{R}{2}.(w_r + w_l).\sin\Phi \\ \dot{\Phi} &= \frac{R}{L}.(w_r - w_l) \end{aligned} \tag{5.1}$$

where x and y represents the coordinate of the robots' center and Φ represents the heading angle. R is a diameter of wheels and L is diameter of the robots.

Figure 5.5: Robostix Microcontroller of the robots

Figure 5.6: Two-wheeled Differential Drive Mobile Robot

We can choose w_r and w_l as control variables that represent angular velocities of right and left wheels.

5.1.3 Performing Control on Robots

Trajectory controlling algorithm runs on Embedded Linux computers that are located on the robots (Fig. 5.4). While reference path is received from Path Planner PC, current position and orientation of robot is received from Visual Positioning PC. According to position and orientation errors, trajectory controller evaluates angular velocities of the motors that leads the robot to track reference path as outer control loop. Evaluated angular velocities that drive to follow the solution paths are sent to Atmega128 microcontroller unit(Robostix) seen in Fig. 5.5 as reference control variables through UART port. Robostix also counts the pulses of the optic encoders to evaluate current angular velocities of motors. Received reference angular velocities and current angular velocities are compared and new PWM signals are produced with PID controllers as inner control loop and then sent to motor drivers. All these control loops demonstrated in Fig. 5.7.

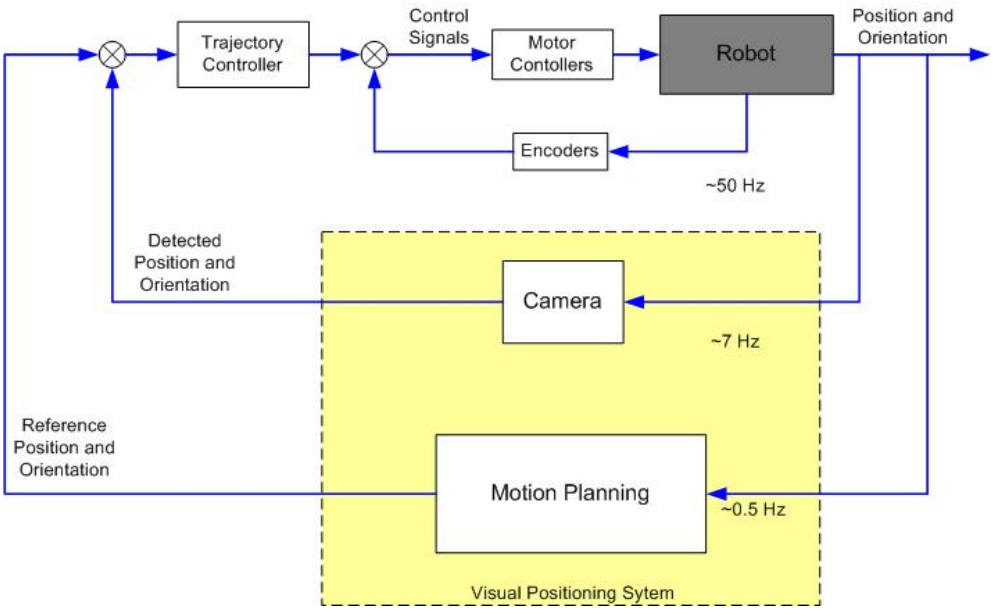

Figure 5.7: Control loop demonstration of the ITUCAL Robotic Testbed

In construction of this robotic testbed, it is aimed that founding general multi-purpose robotic platform. Illustrated implementations have been performed for single mobile robot but goal intention is to extend this platform step by step to multi-agent implementations also includes indoor aerial vehicles. Example result demonstration for path planning implementation can be seen in Fig. 5.8

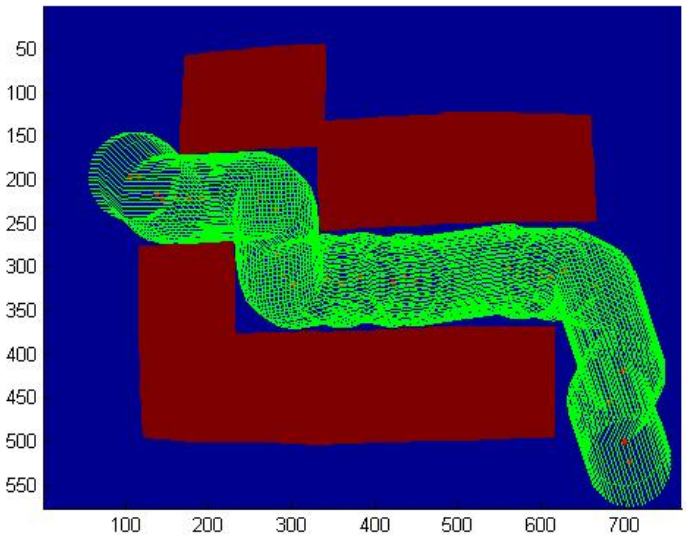

Figure 5.8: Example result path of a path planning implementation for single robot

6. CONCLUSION

There is two major problem in motion planning of dynamical systems. The first problem is the planning in the complex environments and the second is the designing real-time implementable kinodynamic planners in practice for vehicles that have complex dynamics such aerial vehicles.

In our strategy, to deal with complex environment problem, we suggested the finding connectivity path as the initial step. We observed that because of the our initial simplifications and fast spreading ability of the RRT method, this connectivity path generally can be obtain in very short run time. With this approach, defining way points on this connectivity path give us a advantage to divide the big and single-complex problem to serial small-pieced problems on these location. In the further steps, continuity is certainly supplied between these discrete solutions. Furthermore, in the our creating dynamically feasible path planners, we suggest some probabilistic selection approaches and re-shaping ways to deal with complex environment like tunneling effect and B-Spline trajectory repairing methods.

In 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 our creating dynamically feasible path method with the Mode-Based PRM planner, 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 two 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

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. These factors contribute significantly in reduced computation time. 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 creating dynamically feasible path method with the B-Spline Based Planner, a real-time implementable two-step planner strategy is implemented for obtaining 3D real-time flight-path generation for an unmanned helicopter flying in a dense and complex environment. To obtain a fast agile and continuous motion plan we used a B-Spline method to generate probabilistically refined constant acceleration flight paths that pass through way points while providing collision-free and dynamically achievable flight paths. The numerical implementations suggests that the method has real-time properties that can be used for unmanned helicopters operating in urban environments.

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.

REFERENCES

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

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

[3] **Tsianos, K.I.**, **Sucan, I.A. and Kavraki, L.E.**, 2007. Sampling-based robot motion planning: Towards realistic applications, *Computer Science Review*, **1**, 2–11.

[4] **Bohlin, R. and Kavraki, L.**, 2000. Path Planning Using Lazy PRM, Proceedings IEEE International Conference on Robotics & Automation.

[5] **Nieuwenhuisen, D. and Overmars, M.H.**, 2004. Useful cycles in probabilistic roadmap graphs, Proceedings IEEE International Conference on Robotics & Automation, pp. 446–452.

[6] **Hsu, D.**, **Latombe, J.C. and Motwani, R.**, 1999. Path Planning in Expansive Configuration Spaces, *International Journal Computational Geometry & Applications*, **4**, 495–512.

[7] **Sánchez, G. and Latombe, J.C.**, 2001. A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking, Proceedings International Symposium on Robotics Research.

[8] **Yershova, A.**, **Jaillet, L.**, **Simeon, T. and LaValle, S.M.**, 2005. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain, Proceedings IEEE International Conference on Robotics and Automation.

[9] **Jaillet, L.**, **Yershova, A.**, **LaValle, S.M. and Simeon, T.**, 2005. Adaptive Tuning of the Sampling Domain for Dynamic-Domain RRTs, Proceedings IEEE International Conference on Intelligent Robots and Systems.

[10] **Kuffner, J.J. and LaValle, S.M.**, 2000, RRT-Connect: An Efficient Approach to Single-Query Path Planning, Appears in Video Proceedings of IEEE Int'l Conf. on Robotics and Automation.

[11] **Barraquand, J. and Latombe, J.C.**, 1993. Nonholonomic Multibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles, *Algorithmica*, **10**, 121–155.

[12] **LaValle, S.M. and Kuffner, J.J.**, 1999. Randomized Kinodynamic Planning, Proceedings IEEE International Conference on Robotics and Automation, pp. 473–479.

[13] **Hsu, D.**, **Kindel, R.**, **Latombe, J.C. and Rock, S.**, 2001. Randomized Kinodynamic Motion Planning with Moving Obstacles, **B.R. Donald**, **K.M. Lynch and D. Rus**, editors, Algorithmic and Computational Robotics: New Directions, A.K. Peters, Wellesley, MA.

[14] **Choset, H.**, **Burgard, W.**, **Hutchinson, S.**, **Kantor, G.**, **Kavraki, L.E.**, **Lynch, K. and Thrun, S.**, 2005. Principles of Robot Motion: Theory, Algorithms, and Implementation, MIT Press.

[15] **Hsu, D.**, **Jiang, T.**, **Reif, J. and Sun, Z.**, 2003. The Bridge Test for Sampling Narrow Passages with Probabilistic Roadmap Planners, Proceedings IEEE International Conference on Robotics & Automation.

[16] **Boor, V.**, **Overmars, M.H. and van der Stappen, A.F.**, 1999. The Gaussian Sampling Strategy for Probabilistic Roadmap Planners, Proceedings IEEE International Conference on Robotics & Automation, pp. 1018–1023.

[17] **Amato, N.M.**, **Bayazit, O.B.**, **Dale, L.K.**, **Jones, C. and Vallejo, D.**, 1998. OBPRM: An Obstacle-Based PRM for 3D Workspaces, Proceedings Workshop on Algorithmic Foundations of Robotics, pp. 155–168.

[18] **Hsu, D.**, **Sánchez-Ante, G. and Sun, Z.**, 2005. Hybrid PRM sampling with a cost-sensitive adaptive strategy, Proc. IEEE Int. Conf. on Robotics & Automation, pp. 3885–3891.

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

[20] **Siméon, T.**, **Laumond, J.P. and Nissoux, C.**, 2000. Visibility-based probabilistic roadmaps for motion planning, *Advanced Robotics*, **14(6)**, 477–493.

[21] **Hsu, D.**, **Kavraki, L.E.**, **Latombe, J.C.**, **Motwani, R. and Sorkin, S.**, 1998. On Finding Narrow Passages with Probabilistic Roadmap Planners, **P.K. Agrawal**, **L.E. Kavraki and M. Mason**, editors, Robotics: The algorithmic perspective, A.K. Peters, Natick, MA, pp. 141–153.

[22] **Bayazit, O.B.**, **Xie, D. and Amato, N.M.**, 2-6 Aug. 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*, 3433–3440.

[23] **Bohlin, R. and Kavraki, L.E.**, 2001. A Randomized Algorithm for Robot Path Planning Based on Lazy Evaluation, **P. Pardalos**, **S. Rajasekaran and J. Rolim**, editors, Handbook on Randomized Computing, Kluwer Academic Publishers, pp. 221–249.

[24] **Song, G.**, **Miller, S. and Amato, N.**, 2001. Customizing PRM roadmaps at query time, *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, **2**, 1500–1505 vol.2.

[25] **Nielsen, C. and Kavraki, L.E.**, 2000. A Two-Level Fuzzy PRM for Manipulation Planning, volume 3, IEEE Press, pp. 1716–1722.

[26] **LaValle, S.**, 2002. From dynamic programming to RRTs: Algorithmic design of feasible trajectories, **A. Bicchi**, **H.I. Christensen and D. Prattichizzo**, editors, Control Problems in Robotics, Springer-Verlag, pp. 19–37.

[27] **Koyuncu, E.**, **Ure, N.K. and 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*.

[28] **Ghosh, R. and Tomlin, C.**, 2000. Nonlinear inverse dynamic control for mode-based flight, *AIAA Guidance, Navigation and Control Conference*.

[29] **Inalhan, G.**, **Stipanovic, D. and Tomlin, C.**, 10-13 Dec. 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.

[30] **Schouwenaars, T.**, **Feron, E. and How, J.**, 2004. Hybrid model for receding horizon guidance of agile autonomous rotorcraft, *IFAC Symposium on Automatic Control*.

[31] **Piegl, L. and Tiller, W.**, 1997. The NURBS book (2nd ed.), Springer-Verlag New York, Inc., New York, NY, USA.

[32] **Komoriya, K. and 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*, 398–405.

[33] **Eren, H.**, **Fung, C.C. and Evans, J.**, 1999. Implementation of the spline method for mobile robot path control, *Instrumentation and Measurement Technology Conference, IMTC'99. Proceedings of the 16th IEEE*, **2**, 739–744 vol.2.

[34] **Munoz, V.**, **Ollero, A.**, **Prado, M. and Simon, A.**, 8-13 May 1994. Mobile robot trajectory planning with dynamic and kinematic constraints, IEEE International Conference on Robotics and Automation, pp. 2802–2807 vol.4.

[35] **Paulos, E.**, 1998. On-line Collision Avoidance for Multiple Robots Using B-Splines, Technical Report CSD-98-977, UC, Berkeley.

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

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

[38] **Vazquez, G.B.**, **Sossa, A.H. and Diaz-de Leon, S.J.L.**, 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.

[39] **Ure, N.K. and Inalhan, G.**, 2008. Design of a Full-Envelope Nonlinear Flight Controls for an Aggressive Maneuvering F-16 Aircraft, Technical report, ITU, Controls and Avionics Laboratory.

A. MODE-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:

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

Primary modes are the most basic building blocks of arbitrary maneuver sets. In order to sequence these modes efficiently, it is defined 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 [39], it is 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. 4.2 for a complex maneuver.

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

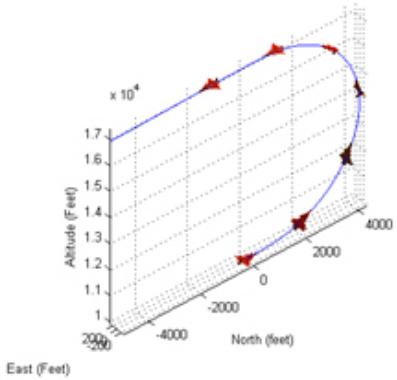

Figure A.1: Controlled Immelmann Turn Using Flight Mode-Based Control Structure

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, Φ,θ,ψ are roll, pitch and yaw Euler angle set, P,Q,R are body angular rates. With the total velocity corresponding toV_T, and aerodynamic angles (angle of attack and side slip angle) are α and β.

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`$\left(r_{loop},\dot{\psi}_d\right)$

E. `Stability Axis Roll` $\left(P_w,\Phi_f\right)$

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 it is specified the 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. A.1 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. A.1. Note that Re-orientation and 3D spin modes have not been included for simplicity of presentation.

B. B-SPLINE CURVES

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

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

The coefficients P_i are called control points. The B-Spline basis functions $N_{i,k}$ are given by the Cox De Boor recursion;

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

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

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 $U = \begin{bmatrix} u_0 & \dots & u_{max} \end{bmatrix}$ is called the knot vector. Frequently, the knot points are referred to as the break points on curve [26]. B-Spline basis function $N_{i,k}$ is zero outside the interval $[u_i, u_{i+k}]$ 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.

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

CIRCULUM VITAE

Emre Koyuncu was born in Ankara in 1984. He received his B.Sc. degree in Electrical Engineering from İstanbul Technical University in 2005. He attend Mechatronics Engineering Master Programme in 2005 and he has been working as a researcher at Controls and Avionics Laboratory at İstanbul Technical University since 2007.