Trajectory planning for aerial vehicles with constraints

Pawit Pharpatara

THESE DE DOCTORAT
DE L'UNIVERSITE PARIS-SACLAY,

Préparée à "Université d'Évry-Val-d'Essonne"

ÉCOLE DOCTORALE N° 580 STIC

Sciences et Technologies de l'Information et de la Communication

Spécialité de doctorat : AUTOMATIQUE

par

M. Pawit PHARPATARA

"Planification de trajectoire sous contraintes d'aéronef"

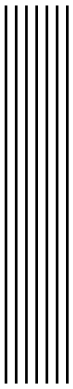
"Trajectory planning for aerial vehicles with constraints"

Thèse preésentée et soutenue à l'Onera Palaiseau en salle NA-00-63, le 22 Septembre 2015

Composition du Jury:

Président du jury:	Véronique PERDEREAU	Prof. à l'UPMC, Paris
Rapporteurs:	Isabelle FANTONI	Directrice de Recherche CNRS Labo. Heudiasyc, CNRS, UTC, Compiègne
	Krzysztof KOZLOWSKI	Prof., Chair of control au Poznan University of Technology, Poland
Examinateur:	Romain PEPY	Ingénieur de recherche (PhD) au MBDA
Encadrant:	Bruno HÉRISSÉ	Ingénieur de recherche (PhD) à l'ONERA
Directrice de thèse:	Yasmina BESTAOUI	Prof. à l'Université d'Évry-Val-d'Essonne

06/2015 Pawit Pharpatara



Abstract

The focus of this PhD thesis is on the trajectory planning module as a part of autonomous aircraft system. Feasible trajectories for aircraft flying in environment cluttered by obstacles are studied. Since aircraft dynamics are complex, nonlinear and nonholonomic; trajectory planning for such systems is very difficult and challenging.

Rapidly-exploring Random Tree or RRT path planner is used as a basis to find a feasible trajectory. The advantage of this algorithm is that it does not consider only the complete vehicle model but also the environment. Two algorithms are developed to find a feasible and optimal solution. The RRT algorithm, combined with a preprocessing of the exploration space, is used for a complete realistic model of the system. However, this approach does not consider any optimal criteria. In order to consider performance criteria, the RRT* algorithm is used based on a simplified model with the help of the artificial potential field as a heuristic to improve the convergence rate to the solution.

The algorithms are simulated in an application of hypersonic aerial vehicles, for example, interceptor missiles flying in high altitude. This makes the aerodynamically controlled aircraft have less maneuverability since the air density decreases exponentially with altitude. 3D shortest paths are developed and used as a metric. Therefore, a feasible and optimal trajectory is obtained efficiently. With these results, real-time constraints will be easier to verify if the algorithm is implemented on board the vehicle. In future work, replanning will be considered to improve the performance of the algorithm in case of dynamic environment or changes in the mission.

Keywords: Path planning, Aerial robotics, Complex environment, RRT algorithm, Artificial potential fields

Résumé

Le sujet de cette thèse porte sur la planification de trajectoire pour un aéronef autonome. Les trajectoires d'aéronefs se déplaçant dans un environnement encombré par des obstacles sont étudiées. La dynamique des aéronefs étant complexe, non linéaire, et non holonome, la planification de trajectoire de ce type de systèmes est un problème très difficile.

L'algorithme Rapidly-exploring Random Tree, ou RRT, est utilisé comme planificateur de base. L'avantage de cet algorithme est qu'il permet de considérer des modèles d'aéronefs complets dans un environnement complexe. Deux algorithmes sont développés pour trouver une solution faisable et optimale. Pour un modèle complet, L'algorithme RRT avec un prétraitement de l'espace d'état est utilisé dans le cas d'une prise en compte du modèle complet du système. Cependant, cette méthode ne considère pas de critères optimaux. Pour y remédier, l'algorithme RRT* est utilisé pour un modèle simplifié du système avec l'aide de champs de potentiels artificiels comme heuristique pour améliorer le taux de convergence vers la solution.

Les algorithmes sont simulés pour une application d'aéronefs hypersoniques, comme par exemple des missiles intercepteurs volants à haute altitude. Les aéronefs ont donc moins de manœuvrabilité parce que la densité de l'air diminue exponentiellement avec l'altitude. Les chemins les plus courts en 3D sont développés et utilisés comme une métrique. Des trajectoires réalisables et optimales sont obtenues efficacement. A partir de ces résultats, les contraintes de temps réel à bord du véhicule seront plus faciles à vérifier. Dans les travaux futurs, la replanification sera considérée pour améliorer la performance de l'algorithme en cas d'environnement dynamique ou de changements dans la mission.

Mots clés: Planification de trajectoire, Robotique aérienne, Environnement complexe, Algorithme RRT, Champs de potentiels artificiels

Acknowledgment

First of all, I would like to express my heartfelt gratitude to my thesis committee members. I gratefully thank Prof. Krzysztof Kozłowski and Prof. Isabelle Fantoni for accepting to be my reviewers of this thesis. I also give my sincerely thanks to Prof. Véronique Perdereau and Dr. Romain Pepy to review this research.

I would like to express my sincerest gratitude to my respectable Supervisor, Prof. Yasmina Bestaoui, and Adviser, Dr. Bruno Hérissé, whose guidance and encouragement were very valuable during my PhD research.

My thanks will be out of tune, if I do not express my thanks to Prof. Samia Bouchafa, who always followed up with my PhD research every year.

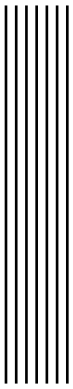
I thank Onera and Université d'Évry-Val-d'Essonne as a part of Université Paris-Saclay for giving me a financial support and an opportunity to do this PhD research.

I take this opportunity to express my gratitude to Lindsay Custodio to be an English proof-reader of this thesis.

My heartfelt thanks to my PhD colleagues at Onera who have always been with me during my PhD research.

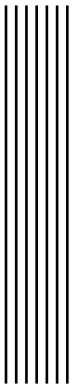
I will be failing with my responsibility as a son if I do not acknowledge the love, support and care of my mother, father, and my siblings.

Last but not least, my final thanks are reserved for my grandfather who motivates me to continue my PhD study. Without him, this PhD thesis will not be possible.



Contents

List of Figures



Nomenclature

Abbreviation

APF	Artificial Potential Field
APN	Augmented Proportional Navigation
BPN	Biased Proportional Navigation
ECEF	Earth-Centered Earth-Fixed coordinate system
ENU	East-North-Up coordinate system
FIFO	First-In First-Out
GA	Genetic Algorithm
GPN	Generalized Proportional Navigation
GPOPS	General Purpose Optimal Control Software
IPN	Ideal Proportional Navigation
LIFO	Last-In First-Out
LOS	Line-Of-Sight
LQG	Linear-Quadratic-Gaussian
LQR	Linear-Quadratic Regulator

MIMO	Multi-Input-Multi-Output
MPC	Model Predictive Control
NED	North-East-Down coordinate system
OPN	Optimal Proportional Navigation
PIP	Predicted Intercept Point
PN/PNG	Proportional Navigation Guidance
PPN	Pure Proportional Navigation
PRM	Probabilistic RoadMap path planner
PRM	Probabilistic Roadmap Planner Method
QFT	Quantitative Feedback Theory
RRT	Rapidly-exploring Random Tree
RRT*	Optimal Rapidly-exploring Random Tree
SISO	Single-Input-Single-Output
SMC	Sliding Mode Control
TPN	True Proportional Navigation
UAV	Unmanned Aerial Vehicle
UCAV	Unmanned Combat Air Vehicle
US-76	U.S. Standard Atmospheres 1976

Constants

ω_0	Angular frequency
ζ	Damping ratio

Scalar variables

α_{com}	Commanded angle of attack
α_{d}	Desired angle of attack
$\alpha_{\text{max}}^{\text{stb}}$	Maximum angle of attack at the actuators limit

$\alpha_{\max}^{\text{struct}}$	Maximum angle of attack at the structural limit of the missile
χ	Azimuth angle
γ	Flight path angle
λ	Longitude
$\mathcal{X}(t)$	Collision-free trajectory at time t
Ψ	Geocentric latitude
ρ	Atmospheric density
θ,ϕ,ψ	Euler angles
a	Acceleration norm
C_{D}	Drag force coefficient
C_{L}	Lift force coefficient
h	Altitude
m	Vehicle mass
p_{atm}	Atmospheric pressure
Q	Dynamic pressure
S	Surface of reference
T_{atm}	Atmospheric temperature
v	Speed

Mathematical operation

$\times$	Cross product
$\mathbf{f}(\cdot)$	State transition function
$\|\cdot\|$	Euclidean norm
$\|\cdot\|$	Absolute value
$^\top$	Transpose

Set

$\mathbb{X}_{\text{free}}$	Collision-free exploration space
$\mathbb{X}_{\text{goal}}$	Objective exploration space
$\mathbb{X}_{\text{obs}}$	Obstacle space
$\mathbb{U}$	Admissible control input space
$\mathbb{X}$	State space

Column vector variables

$\boldsymbol{\tau}_{\text{net}}$	Net torque
$\boldsymbol{\omega}$	Rotational velocity
$\boldsymbol{\xi}$	Position
$\mathbf{e}_1^{\text{b}},\mathbf{e}_2^{\text{b}},\mathbf{e}_3^{\text{b}}$	Unit vectors in body-fixed coordinate system
$\mathbf{e}_1^{\text{ecef}},\mathbf{e}_2^{\text{ecef}},\mathbf{e}_3^{\text{ecef}}$	Unit vectors in ECEF coordinate system
$\mathbf{e}_1^{\text{ned}},\mathbf{e}_2^{\text{ned}},\mathbf{e}_3^{\text{ned}}$	Unit vectors in NED coordinate system
$\mathbf{f}_{\text{net}}$	Net force
$\mathbf{a}$	Acceleration
$\mathbf{g}$	Gravity of the Earth
$\mathbf{u}$	Control input
$\mathbf{v}$	Linear velocity
$\mathbf{x}$	State variable

Part I

Introduction

1 Introduction

The development of human technology has grown very rapidly from the ground to water, water to air, and air to space. It had been an impossible dream for the human to fly in the sky until the world's first successful manned airplane was built by the Wright brothers in 1903. Since then, aircraft technologies have grown rapidly. Aircraft are machines or vehicles capable of flying by using the support of the atmosphere, mainly the air. Aircraft are not only airplanes but can also be any flying machines such as kites, hot air balloons, airships, gliders, missiles, and drones or unmanned aerial vehicles (UAVs). Then, technologies evolve from manual to autonomous system. Autonomous systems are systems that can operate by themselves without any human interventions so that the mission in the uninhabited areas can be done flawlessly. Thus, the study of autonomous aircraft is an interesting subject nowadays.

The general mission of aircraft is defined by a starting location to a destination such as going from one city to another. A trajectory between two vehicle states can be found under several different types of constraints (variation of air density, wind velocity, different flight stages, obstacles, etc.) and some uncertainties in state measurement or in dynamic model. The environment or the mission can also change any time during the flight. Thus, as a part of an autonomous mission, the trajectory generation module is very important. Once a feasible solution is known, the vehicles such as military drones (unmanned combat air vehicles also known as UCAVs) will be able to complete the mission successfully. The trajectory generation module can be computed offline before the mission starts and then gives the obtained solution to the vehicles. In the even better scenario, the trajectory generation module can be embedded on board the vehicle. Consequently, the vehicles are

capable of finding the solution by themselves during the mission in case of any perturbations, any changes in mission or in a dynamic environment. This increases choices of decisions to make, *i.e.* the reactivity of the vehicles increases, which makes the vehicles work more efficiently.

In this thesis, we are interested in the trajectory generation for UAVs whose mission is to travel from one state to another. It is very similar to the rendezvous problem in navigation.

1.1 Trajectory generation for aerial vehicles

1.1.1 Motivations

Trajectory generation is an important part of any autonomous system involving motions. Originally, trajectory generation is widely studied in ground robotics and manipulator systems. As time passes, more and more studies in trajectory planning are extended to underwater and aerial robotics.

In aeronautics and aerospace, the trajectory generation starts with a prelaunch phase of the vehicles. A generated trajectory should not consider only the aircraft states (position, orientation, and speed), departure and arrival points, but also aerodynamic constraints (maneuverability, gravity, air density, etc.) and constraints induced by obstacles. These make the problem of trajectory generation very challenging.

Trajectory generation has to face both static and dynamic environments, for example, radar detection systems, cooperating and non-cooperating vehicles, as well as the constraints imposed on the system. Moreover, if the objective task changes during the mission, the trajectory has to be re-configured with no performance loss. Consequently, such embedded algorithms that can be very demanding in terms of numerical computations also need to consider real-time constraints. Then, it is of significance to solve a trajectory planning problem with a good computational efficiency.

1.1.2 Difficulties in trajectory planning in a complex environment

In most of aerial applications, autonomous aerial vehicles are operated aerodynamically using the forces caused by the air pressure applied on the control surface of the vehicles and by propulsion. For example, air density, atmospheric pressure, and temperature decrease exponentially with altitude. Since a lot of aerial vehicles can fly in a wide range of altitude, their maneuvering capability also decreases exponentially with altitude. Moreover, obstacle avoidance is a difficult problem since obstacles can induce state constraints to the system. These state constraints make problems very difficult to solve with many methods. The

nature of obstacles can be both static and dynamic such as no-fly zones, physical obstacles, cooperated or non cooperated aerial vehicles, and some perturbations in the environment. For example, at low altitude, the perturbations can be caused by the wind or the weather conditions, while, at higher altitude, these perturbations decrease and then can be less observed.

These make the trajectory planning for aerial vehicles complex and nonlinear problems with constraints. Thus, the trajectory planning in such system is very challenging and interesting.

1.2 Scientific context of this thesis

Several methods have been studied using UAV flight controls: linear flight control theories, nonlinear flight control theories, guidance laws, numerical methods, etc. However, the problem cannot be solved completely. Even if some methods such as trajectory optimization can solve and find an optimal solution, it is time consuming to solve complex and nonlinear problems using realistic system with obstacle constraints. Most of the solutions are based either on the simplified system or under other restrictive hypotheses.

On the other hand, in robotics, trajectory planning have been very popular recently. Trajectory planning methods are used in many fields such as in biology, marketing and even in video games. The aim of trajectory planning is to find a feasible path from an initial state to a goal state while avoiding obstacles. Trajectory planning methods are interesting because some of them can find a solution while considering the complete system and the environment cluttered with obstacles. This is what is missing from the previously mentioned methods. However, the solution found by these methods are rarely optimal, or in the best case suboptimal.

In this thesis, trajectory planning algorithms are studied by combining trajectory planning methods with optimal control theory. The trajectory planning method is used in order to consider complex systems and environment models in the calculation. Then, the optimal control theory is used to improve the quality of the solution found by trajectory planning method regarding a specified criterion.

1.3 Thesis objectives

The objective of this thesis is to find an algorithm(s) that can find an optimal trajectory for aerial vehicles flying from one state to another in a complex environment cluttered by obstacles. The nature of the obstacles can be, for example, radar detection systems, cooperating and non-cooperating vehicles, etc. Thus, trajectory planning in such environment is a

complex, nonlinear, nonholonomic problem that is difficult to solve using only the classical UAV control laws.

In this thesis an application of hypersonic aerial vehicles such as the interceptor missiles, is considered as a case study. Since the aerial vehicle flies in the earth atmosphere, the environment can be considered heterogeneous in the sense that the air density decreases with altitude. This makes the aerodynamically controlled aircraft such as missiles have less maneuverability at higher altitude. The trajectory planning algorithms developed in this thesis should be able to find a trajectory from one state to another state successfully while considering these constraints.

1.4 Solution approach

The basic approach of trajectory planning for aerial vehicles is the waypoints planning. Waypoints planning is widely used in many domains. However, it can lead to an unfeasible trajectory if all constraints of the system are not considered or the path following algorithm cannot follow the waypoints. In this thesis, instead of finding waypoints for the aircraft to follow, directly finding a feasible trajectory that can be followed by path following algorithms without any difficulty can be more interesting since the path following is not needed to be used after. Two approaches of solving this problem are proposed.

The first approach is to find a feasible trajectory using a realistic model. This approach does not require any path following algorithms for the aircraft to execute the path since the obtained trajectory is the real feasible trajectory itself. Control input sequence used to execute this trajectory is also obtained at the same time as the obtained trajectory. A preprocessing of the exploration space is introduced to improve the convergence rate to the solution of the algorithm. However, it is difficult to prove the optimality according to a specified criterion of the solution using this method.

The second approach is proposed using a simplified model instead of using a realistic model since it is easier to prove the optimality of the solution. The optimal RRT or RRT* algorithm which has an asymptotic optimality property, *i.e.* almost-sure convergence to an optimal solution, is used as a basis path planner. The aim of this approach is to find a feasible and easy to follow reference trajectory using path following algorithms. Heuristic using the Artificial Potential Field or APF is proposed to increase the convergence rate of the algorithm.

1.5 Thesis contributions

- This thesis focuses on trajectory planning of a single unmanned aerial vehicle (UAV) for a rendez-vous problem. The methodology for the trajectory planning is divided into two approaches. In the first approach, the trajectory planning algorithm tries to find a realistic trajectory for UAVs together with a sequence of control inputs. The advantage of this approach is that missions are surely achieved by the vehicles in reality. However, it is very difficult to prove the optimality of obtained trajectories using the complete model of vehicle and environment. The second approach is proposed, by using a simplified model; optimal trajectories can be found and proven. These trajectories are used as references for the vehicles to follow using trajectory tracking and path following algorithms.

- The trajectories are generated using different methods. Guidance laws are used to generate trajectories to find a realistic trajectory in the first approach, while the shortest path based on Dubins-like model is used to generate trajectories in the second approach.

- In both 2D and 3D application, the shortest path based on Dubins-like model is developed so that they can adapt to a heterogeneous environment where the turning radius of the vehicle is not constant and is decreasing with altitude. The shortest path length is also used as a metric function to determine a distance between two vehicle states in space.

- A framework using an algorithm with asymptotic optimal property is used in order to achieve an optimal trajectory.

- Two approaches are proposed to improve the convergence rate of the algorithm to the optimal solution: preprocessing of the exploration space and integration of Artificial Potential Field or APF.

1.6 Disseminations, presentations from this thesis

1.6.1 International conferences papers

- Pharpatara, P., Hérissé, B., Pepy, R., and Bestaoui, Y. (2013). Sampling-based path planning: a new tool for missile guidance. *In Proceedings of the IFAC Symposium on Automatic Control in Aerospace*, pages 131-136, DOI:10.3182/20130902-5-DE-2040.00091.

- Pharpatara, P., Pepy, R., Hérissé, B., and Bestaoui, Y. (2013). Missile trajectory shaping using sampling-based path planning. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2533-2538, DOI: 10.1109/IROS.2013.6696713.

- Pharpatara, P., Hérissé, B., Pepy, R., and Bestaoui, Y. (2015). Shortest path for aerial vehicles in heterogeneous environment using RRT*. *In Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6388-6393, DOI: 10.1109/ICRA.2015.7140096.

- Pharpatara, P., Hérissé, B., and Bestaoui, Y. (2015). 3D-shortest paths for a hypersonic glider in a heterogeneous environment. *In Proceedings of the IFAC Workshop on Advanced Control and Navigation for Autonomous Aerospace Vehicles*, pages 186-191.

1.6.2 International journal papers

- Pharpatara, P., Hérissé, B., and Bestaoui, Y. (2015). 3D trajectory planning of aerial vehicles using RRT*. *IEEE Transactions on Control Systems Technology*, submitted.

1.7 Organization of the thesis

This thesis is divided into five parts. Part I is the general introduction to trajectory generation for aerial vehicles and the problem formulation of this thesis. Part II contains two chapters concerning the state of the art. One chapter is about the UAV flight controls and another is about the trajectory planning methods used in robotics. The contributions of this thesis are included in part III and IV. All of the results presented in these parts are obtained from MATLAB simulations. Part V includes all necessary calculations from both literature reviews and contribution work. They are all detailed below:

- In part I, the trajectory generation for aerial vehicles is firstly explained along with problems, the proposed approaches, and the importance of this. Then, the general dynamics of a UAV and the environment are explained along with necessary constraints in mathematical form. Then, the problem formulation is presented.

- In part II, the state of the art of the existing methods are presented in two chapters separated by domain of applications.

  - In chapter 3, some existing UAV flight controls are presented and explained with their advantages and drawbacks. The classical control laws including linear and nonlinear control laws and specific guidance laws are simple and easy to

implement on board the vehicle but they cannot anticipate the future changes in missions and environment. The optimal control theory can be used to improve the performance of the classical control laws. However, the same problem persists. The numerical methods can find an optimal solution for problems but they need a good initial guess to make the methods converge to the optimal solution. Sometimes, they also require a lot of computational effort for a complex and nonlinear problem with constraints. These make them not suitable for the real time implementation. The artificial intelligent methods are introduced to improve the performance of the existing methods by combining or training knowledge to the machine. These methods require good know-hows suitable for each problem. Most of these methods are calculated based on the restrictive vehicle model. They do not consider the environment model or the future conditions in the calculation either. In the next chapter, trajectory planning methods used in robotics are presented to help integrating the complete system model and environment into the calculation.

- In chapter 4, basic trajectory planning methods used in robotics are presented to introduce interesting ideas. Some of methods are suitable for the low dimensional problems such as A* and its variants for the discrete system. The potential field path planning is suitable for a continuous system but it is quite difficult to be implemented with a complex and nonlinear system. The sampling-based path planning methods are the most interesting ones, especially the Probabilistic RoadMap path planner (PRM) and the Rapidly-exploring Random Tree (RRT). The PRM is a multi-query path planner which is suitable for finding paths for several robots at the same time, while the RRT is a single-query path planner. Since the RRT is a single-query path planner, it can find a solution while searching the state space at the same time unlike the PRM. Thus, the RRT is chosen to be used as a path planner in this thesis.

- In part III, trajectory planning using a realistic model is studied. The algorithm framework is generally presented and the performance is demonstrated using an interceptor missile application.

  - In chapter 5, the RRT algorithm is used as a basis path planner. It is applied in an interceptor missile application including both boost phase and midcourse phase. The path calculations based on the optimal control theory, namely Dubins' paths, are used as a metric. Moreover, the classical guidance laws, such as proportional navigation and kappa guidance, are used to build an exploration

tree. The results are satisfying compared to the ones obtained by using solely the classical guidance laws. The results show that, while the classical guidance laws have a difficulty to find solutions for some problems, the RRT algorithm is capable of finding ones. However, no optimal criterion is considered and the computational time can be long.

- In chapter 6, preprocessing methods of the exploration space are introduced to improve the results obtained from the previous chapter. The artificial potential fields (APF) and the Dubins' paths in a heterogeneous environment are proposed for the preprocessing. However, It is difficult to find a suitable APF function since the system is complex, nonlinear, and nonholonomic. Then, the Dubins' paths, in a heterogeneous environment, are used in to preprocess the exploration space. Then, the same algorithm used in the previous chapter is launched. This framework is then tested with an interceptor missile application during the mid-course phase. Apparently, the algorithm converges to a solution more rapidly. Moreover, the obtained solutions appear to be smoother and shorter than the previous ones. Yet, no global optimal criterion is verified in this method. In addition, the reduced exploration space also makes the RRT lose its main advantage, the exploration of high-dimensional problem without a need of approximations. Thus, another approach is proposed in the next part.

- In part IV, trajectory planning using a simplified model is studied. The aim is to find a reference trajectory for the UAVs to follow using some path following algorithms. The algorithm framework is also generally presented and the performance is demonstrated using a hypersonic aerial vehicle application.

  - In chapter 7, the optimal RRT or RRT* algorithm with its asymptotic optimal property is explained. The motion planning framework is developed using the RRT* algorithm as a basis path planner. Moreover, the APF is integrated in the algorithm as a heuristic to accelerate the convergence rate to the optimal solution. In hypersonic aerial vehicle applications such as interceptor missiles, the framework is simulated in both 2D and 3D applications. Dubins' paths in heterogeneous environment are used as metric and node expansion method. In 2D application, only the RRT* algorithm is used. The aim is to show the capability of the RRT* algorithm in finding a solution close to the optimal solution while avoiding obstacles within permitted time. In 3D application, the complete framework is simulated. The 3D Dubins' paths in a heterogeneous environment are developed and used based on a Dubins-like aerial vehicle model. As a result,

this framework shows good performance in finding a feasible and optimal trajectory while avoiding obstacles in 3D plane. However, the obtained trajectories are based on the simplified but yet close to real system model. Even though, they are not totally executable by the real system, they are very close to the real executable trajectory. Hence, they can be used as reference trajectories which facilitate the path following algorithm using, for example, MPC algorithm.

- The overall conclusions are made. Moreover, some perspectives and ideas are given for the future work.
- In part V, all mathematical calculations are separated from the main body of this thesis and shown in this part.

2 Unmanned aerial vehicle modeling for path planning

In order to solve trajectory planning problems of UAVs, the system model of the UAV and of the environment must be studied and defined. In this paper, the introduction to the UAV system model is presented such as the coordinate systems used in aeronautics and aerospace. Then, the dynamics of the UAV and the environment model are described in details. Finally, the problem statement containing system modeling and problem formulation is described.

2.1 System modeling

Before going into the UAV equations of motion and dynamics, several coordinate systems used in aeronautics and aerospace domains are explained.

2.1.1 Coordinate Systems

Four orthogonal-axes systems are usually defined to develop the appropriate equations of motion of aerial vehicles:

1. The *inertial* frame, which is fixed in space for which Newton's laws of motion are valid.

2. An *Earth-Centered* frame that rotates with the Earth.

3. An *Earth-Surface* frame that is parallel to the Earth surface, whose origin is at the vehicle center of gravity defined in two ways: north, east and down (NED) directions or east, north and up (ENU) directions.

4. A *body-fixed* frame that is conventional to the body of the vehicle. The center of this frame is at the center of gravity of the vehicle, and its components are forward, out of the right side, and down for NED coordinate and forward, out of the left side, and up for ENU coordinate.

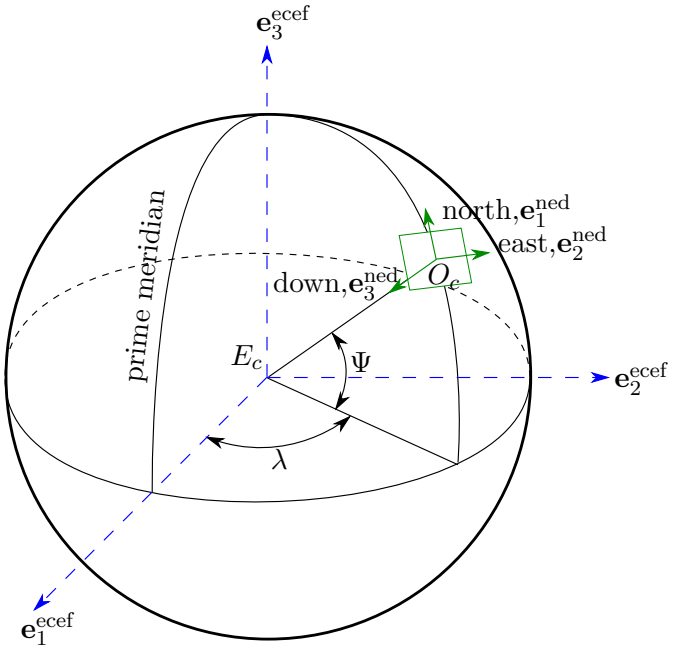


Figure 2.1: Illustration of ECEF and local NED coordinates

Figure 2.1 illustrates relation between Earth-Centered Earth-Fixed (ECEF), which is equivalent to inertial coordinate system, and North-East-Down (NED) coordinate system. The ECEF coordinate system is presented by the blue dashed line while the NED coordinate system is presented by the green solid line.

The ECEF coordinate system ($\mathbf{e}_1^{\text{ecef}}$,$\mathbf{e}_2^{\text{ecef}}$,$\mathbf{e}_3^{\text{ecef}}$) has its origin at the centre of the Earth E_c and rotates with the Earth. The $\mathbf{e}_1^{\text{ecef}}$-axis passes through the equator at the prime meridian. The $\mathbf{e}_3^{\text{ecef}}$-axis passes through north pole but it does not exactly coincide with the instantaneous Earth rotational axis. The $\mathbf{e}_2^{\text{ecef}}$-axis can be determined by the right-hand rule to be passing through the equator at 90° longitude. The NED coordinate system is defined from a plane tangent to the Earth surface at a specific location O_c. By convention, the north axis is labeled $\mathbf{e}_1^{\text{ned}}$, the east $\mathbf{e}_2^{\text{ned}}$ and the down $\mathbf{e}_3^{\text{ned}}$. We can transform the ECEF coordinate system into the local NED coordinate system by using a local reference point. In our case, if the location of the control station is located at (x_0^{ecef},y_0^{ecef},z_0^{ecef}) in ECEF

coordinate and a UAV at $(x_{\mathrm{m}}^{\mathrm{ecef}},y_{\mathrm{m}}^{\mathrm{ecef}},z_{\mathrm{m}}^{\mathrm{ecef}})$ in ECEF coordinate then the vector pointing from the control station to the UAV in the NED frame is:

$$\begin{bmatrix} x^{\mathrm{ned}} \\ y^{\mathrm{ned}} \\ z^{\mathrm{ned}} \end{bmatrix} = \begin{bmatrix} -\sin\lambda\cos\Psi & -\sin\lambda\sin\Psi & \cos\lambda \\ -\sin\Psi & \cos\Psi & 0 \\ -\cos\lambda\cos\Psi & -\cos\lambda\sin\Psi & -\sin\lambda \end{bmatrix} \begin{bmatrix} x_{\mathrm{m}}^{\mathrm{ecef}} - x_0^{\mathrm{ecef}} \\ y_{\mathrm{m}}^{\mathrm{ecef}} - y_0^{\mathrm{ecef}} \\ z_{\mathrm{m}}^{\mathrm{ecef}} - z_0^{\mathrm{ecef}} \end{bmatrix} \tag{2.1}$$

where λ is the longitude and Ψ is the geocentric latitude.

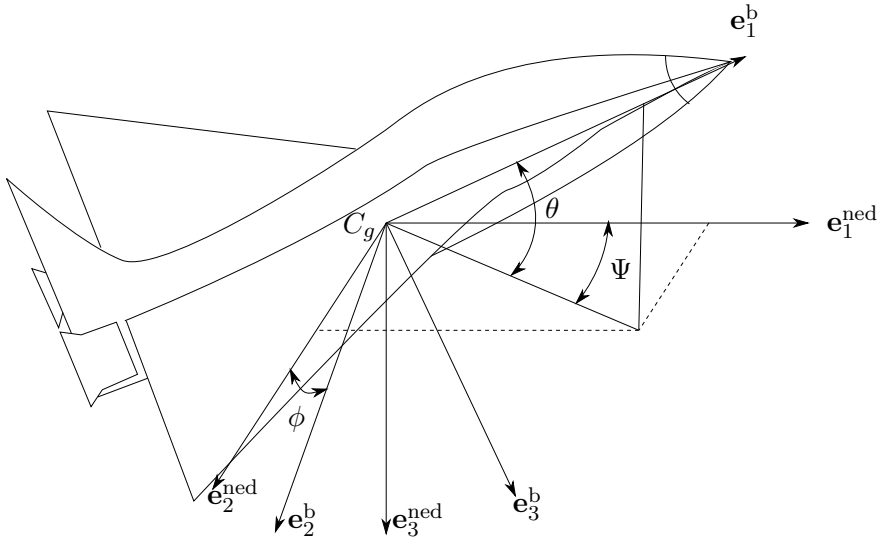


Figure 2.2: UAV body-fixed coordinates

The UAV body-fixed coordinate system, shown in figure 2.2, is denoted $(\mathbf{e}_1^{\mathrm{b}},\mathbf{e}_2^{\mathrm{b}},\mathbf{e}_3^{\mathrm{b}})$. It is fixed with respect to the UAV at its center of gravity C_g and moves with the UAV. The positive $\mathbf{e}_1^{\mathrm{b}}$-axis coincides with the forward direction of the UAV. The $\mathbf{e}_2^{\mathrm{b}}$-axis is to the right of the missile. The positive $\mathbf{e}_3^{\mathrm{b}}$-axis points down according to right-hand rule. This coordinate system is similar to NED coordinate system.

Positions on the local NED coordinate system $(x^{\mathrm{ned}}, y^{\mathrm{ned}}, z^{\mathrm{ned}})$ can also be transformed into the missile body-fixed system $(x^{\mathrm{b}}, y^{\mathrm{b}}, z^{\mathrm{b}})$ by using:

$$\begin{bmatrix} x^{\mathrm{b}} \\ y^{\mathrm{b}} \\ z^{\mathrm{b}} \end{bmatrix} = C_{\mathrm{ned}}^{\mathrm{b}} \begin{bmatrix} x^{\mathrm{ned}} \\ y^{\mathrm{ned}} \\ z^{\mathrm{ned}} \end{bmatrix}, \tag{2.2}$$

on the other hand

$$\begin{bmatrix} x^{\mathrm{ned}} \\ y^{\mathrm{ned}} \\ z^{\mathrm{ned}} \end{bmatrix} = C_{\mathrm{b}}^{\mathrm{ned}} \begin{bmatrix} x^{\mathrm{b}} \\ y^{\mathrm{b}} \\ z^{\mathrm{b}} \end{bmatrix}, \tag{2.3}$$

where

$$C^{\mathrm{b}}_{\mathrm{ned}} = C^{\mathrm{b}}_{\mathrm{ned_x}} C^{\mathrm{b}}_{\mathrm{ned_y}} C^{\mathrm{b}}_{\mathrm{ned_z}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix},$$

and

$$C^{\mathrm{ned}}_{\mathrm{b}} = {C^{\mathrm{b}}_{\mathrm{ned}}}^{-1} = (C^{\mathrm{b}}_{\mathrm{ned}})^T$$

$$= \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix},$$

where (θ,ϕ,ψ) are euler angles , $C^{\mathrm{b}}_{\mathrm{ned}}$ is the transformation from NED to body-fixed coordinate system and vice-versa for $C^{\mathrm{ned}}_{\mathrm{b}}$.

Please note that ambiguities (or singularities) can result from using the above transformation (i.e., as θ, ϕ, $\psi \to \pi$). Therefore, in order to avoid these ambiguities, the ranges of the Euler angles (θ, ϕ, ψ) are limited as follows:

$$-\pi \leqslant \phi < \pi \text{ or } 0 \leqslant \phi < 2\pi,$$
$$-\pi \leqslant \psi < \pi,$$
$$-\pi/2 \leqslant \theta < \pi/2 \text{ or } 0 \leqslant \psi < 2\pi.$$

2.1.2 Dynamics

It is assumed that the UAV has six degrees of freedom (6-DOF). The six degrees of freedom consist of three translations and three rotations along and about the aircraft axes ($\mathbf{e}^{\mathrm{b}}_1$, $\mathbf{e}^{\mathrm{b}}_2$, $\mathbf{e}^{\mathrm{b}}_3$). These motions are illustrated in figure 2.3, the linear velocity is denoted $\mathbf{v} = (v_x, v_y, v_z)$ and the angular velocity is denoted $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$. In compact form, the dynamics of the translation and the rotation of a rigid body with mass m may be expressed by the following equations (see [Sio04] for more details):

$$\text{Translation} : \mathbf{f} = m\dot{\mathbf{v}} + \boldsymbol{\omega} \times m\mathbf{v}, \tag{2.4}$$

$$\text{Rotation} : \boldsymbol{\tau} = \mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}, \tag{2.5}$$

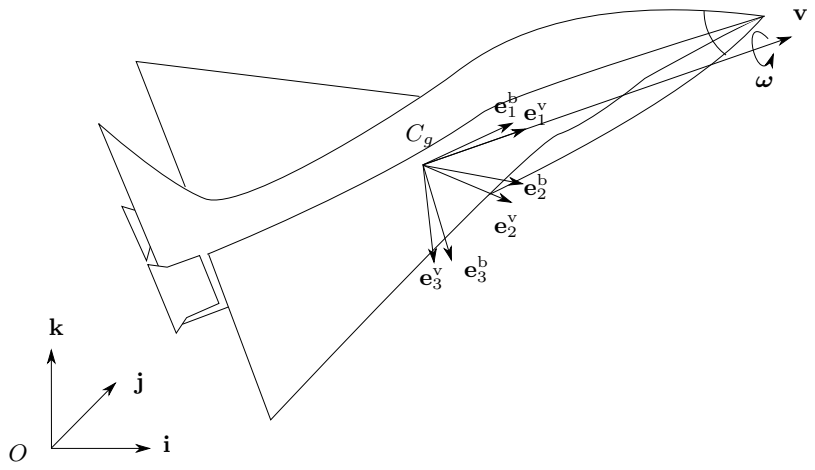


Figure 2.3: Motions of a UAV

where $\mathbf{f}_{\text{net}} = (F_x, F_y, F_z)$ is the net force on the UAV, $\boldsymbol{\tau} = (\tau_x, \tau_y, \tau_z)$ is the net torque of these forces about its center of mass C_g, the dot indicates differentiation with respect to time, and $\mathbf{J}$ is the UAV inertia tensor which is constant in the body-fixed frame.

In general, UAVs are left-right symmetric. By assuming that the y-axis is orthogonal to the UAV plane of symmetry, the inertia tensor $\mathbf{J}$ is defined as

$$\mathbf{J} = \begin{bmatrix} J_{xx} & 0 & -J_{xz} \\ 0 & J_{yy} & 0 \\ -J_{xz} & 0 & J_{zz} \end{bmatrix}.$$

According to equations (2.4) and (2.5), the dynamics of the UAV is:

$$\frac{1}{m}\mathbf{f} \equiv \frac{1}{m}\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} \dot{v_x} + \omega_y v_z - \omega_z v_y \\ \dot{v_y} + \omega_z v_x - \omega_x v_z \\ \dot{v_z} + \omega_x v_y - \omega_y v_x \end{bmatrix}, \tag{2.6}$$

$$\boldsymbol{\tau} \equiv \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} J_{xx}\dot{\omega_x} - J_{xz}\dot{\omega_z} + \omega_y\omega_z(J_{zz} - J_{yy}) - \omega_x\omega_y J_{xz} \\ J_{yy}\dot{\omega_y} + \omega_x\omega_z(J_{xx} - J_{zz}) + (\omega_x^2 - \omega_z^2)J_{xz} \\ J_{zz}\dot{\omega_z} - J_{xz}\dot{\omega_x} + \omega_x\omega_y(J_{yy} - J_{xx}) + \omega_y\omega_z J_{xz} \end{bmatrix}. \tag{2.7}$$

The net force includes aerodynamic forces, trust, and gravity.

2.1.3 Simplified UAV equations of motion for path planning

Since the complete equations of motion are very complex. They can be simplified for the path planning purpose by using the vehicle as a rigid point mass model with kinematic path constraints.

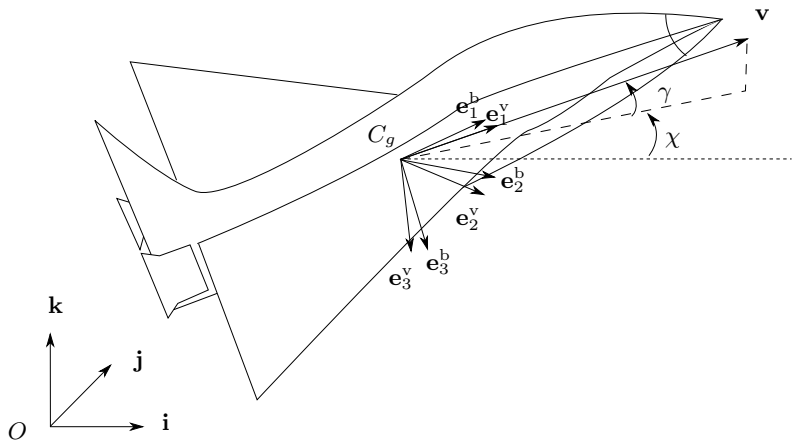


Figure 2.4: Illustration of the flight path angle γ and azimuth angle χ

In figure 2.4, denote γ and χ the orientation w.r.t $\mathcal{I}$ of the velocity $\mathbf{v}$, where γ is the flight path angle, *i.e.* the angle between the vehicle axis $\mathbf{e}_1^{\mathrm{b}}$ and the horizontal plane $(\mathbf{i}, \mathbf{j})$, and χ is the azimuth angle, *i.e.* the angle from the horizontal axis $\mathbf{i}$ to the projection of the velocity $\mathbf{v}$ on horizontal plane $(\mathbf{i}, \mathbf{j})$. There are four principal forces acting on the UAV: the weight $m\mathbf{g}$ where $\mathbf{g}$ is the gravity of the Earth, thrust $\mathbf{T} = (T_1, T_2, T_3)$, aerodynamic force $\mathbf{f}_{\mathrm{a}} = (F_{a_1}, F_{a_2}, F_{a_3})$, and other perturbation forces $\mathbf{f}_{\mathrm{p}} = (F_{p_1}, F_{p_2}, F_{p_3})$. The standard model in [Tré12] is used as a basis with the hypothesis that the mission can be accomplished in shortest time. Thus, the Earth rotation has very few effects on the vehicle so it is neglected. The equations of motion of a point mass model aircraft can be formulated as

$$\begin{cases} \dot{x} = v\cos\gamma\cos\chi, \\ \dot{y} = v\cos\gamma\sin\chi, \\ \dot{z} = v\sin\gamma, \\ \dot{v} = -g\sin\gamma + \dfrac{F_{a_1}}{m} + \dfrac{F_{p_1}}{m} + \dfrac{T_1}{m} \\ \dot{\gamma} = -\dfrac{g}{v}\cos\gamma + \dfrac{F_{a_3}}{mv} + \dfrac{F_{p_3}}{mv} + \dfrac{T_3}{mv}, \\ \dot{\chi} = \dfrac{F_{a_2}}{mv\cos\gamma} + \dfrac{F_{p_2}}{mv\cos\gamma} + \dfrac{T_2}{mv\cos\gamma}, \end{cases} \tag{2.8}$$

where x, y, z are positions, v is the vehicle speed, $g = ||\mathbf{g}||$, $\gamma \in [-\pi/2, \pi/2]$ and $\chi \in [-\pi, \pi]$. The aerodynamic forces F_{a_1}, F_{a_2}, and F_{a_3} including the drag force $\mathbf{f}_{\mathrm{D}}$ and the lift forces $\mathbf{f}_{\mathrm{L}_2}$ and $\mathbf{f}_{\mathrm{L}_3}$ are expressed as follows:

$$\begin{aligned} \mathbf{f}_{\mathrm{D}} &= -\frac{1}{2}\rho(z)SC_D v_a^2 \mathbf{e}_1^{\mathrm{v_a}}, \\ \mathbf{f}_{\mathrm{L}_2} &= \frac{1}{2}\rho(z)SC_{L_2} v_a^2 \mathbf{e}_2^{\mathrm{v_a}}, \\ \mathbf{f}_{\mathrm{L}_3} &= -\frac{1}{2}\rho(z)SC_{L_3} v_a^2 \mathbf{e}_3^{\mathrm{v_a}}, \end{aligned}$$

where C_D is the drag coefficient, C_{L_2} and C_{L_3} are the lift coefficients, S is the surface of reference, $\rho(z)$ is the density of air and v_a is the magnitude of the air velocity $\mathbf{v}_\mathrm{a}$ expressed as $\mathbf{v}_\mathrm{a} = \mathbf{v} - \mathbf{v}_\mathrm{w}$ where $\mathbf{v}_\mathrm{w}$ is the wind velocity (see figure 2.5). $\mathbf{e}_1^{\mathrm{v_a}}$, $\mathbf{e}_2^{\mathrm{v_a}}$, and $\mathbf{e}_3^{\mathrm{v_a}}$ are the basis vectors related to $\mathbf{v}_\mathrm{a}$. Note that if $\mathbf{v}_\mathrm{w} = 0$, $\mathbf{e}_2^{\mathrm{v_a}} = \mathbf{e}_2^{\mathrm{v}}$, $\mathbf{e}_3^{\mathrm{v_a}} = \mathbf{e}_3^{\mathrm{v}}$ and the drag and lift coefficients depend on the angle of attack α [Sio04], *i.e.* the angle between the vehicle axis $\mathbf{e}_1^{\mathrm{b}}$ and the air velocity $\mathbf{v}_\mathrm{a}$ (see figure 2.5).

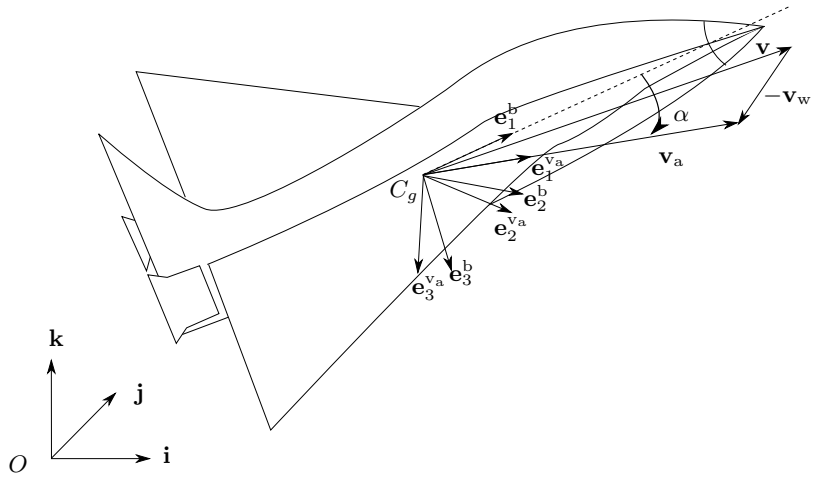


Figure 2.5: Illustration of the air velocity $\mathbf{v}_\mathrm{a}$ and the angle of attack α

For the control of the vehicle, the aerodynamic forces $\mathbf{f}_{\mathrm{L}_2}$ and $\mathbf{f}_{\mathrm{L}_3}$ as well as the propulsion forces (T_1, T_2, T_3) are used.

2.1.3.1 Control Input

The desired angle of attack α_d of the UAV is required in order to calculate the control input $\mathbf{a}$, the acceleration which is normal to the vehicle velocity, equivalent to aerodynamic lift divided by mass. The desired angle of attack α_d is decided by calculating the following angles of attack [Sio04]: commanded angle of attack α_com, maximum angle of attack at the actuators limit $\alpha_\mathrm{max}^\mathrm{stb}$, and maximum angle of attack at the structural limit of the UAV $\alpha_\mathrm{max}^\mathrm{struct}$. Then, α_d is the lowest of all three angles, *i.e.* $\alpha_\mathrm{d} = \min(\alpha_\mathrm{com}, \alpha_\mathrm{max}^\mathrm{stb}, \alpha_\mathrm{max}^\mathrm{struct})$.

1. Commanded angle of attack α_com

   The commanded angle of attack is the angle of attack related to the normal acceleration of the UAV in velocity frame $\mathcal{V}$. The commanded angle of attack can be obtained by iteratively solving this equation:

   $$a = ||\mathbf{a}|| = \frac{QS}{m} C_\mathrm{L}(\alpha_\mathrm{com}) + \frac{||\mathbf{T}^\perp||}{m} \sin\alpha_\mathrm{com}$$

   where $Q = \frac{1}{2}\rho v^2$ is the dynamic pressure, $\mathbf{T}^\perp$ is the propulsive force, containing T_2 and T_3 from system (2.8), perpendicular to the vehicle velocity axis.

Here, a is the aerodynamic lift acceleration which is computed from the commanded acceleration a_{com} by

$$a = a_{\text{com}} - I_{\text{g}}\mathbf{g} \cdot \mathbf{e}_3^{\text{v}}$$

where a_{com} is computed by the guidance algorithm and I_{g} is zero if the input guidance parameter is zero or negative, and I_{g} is equal to one otherwise.

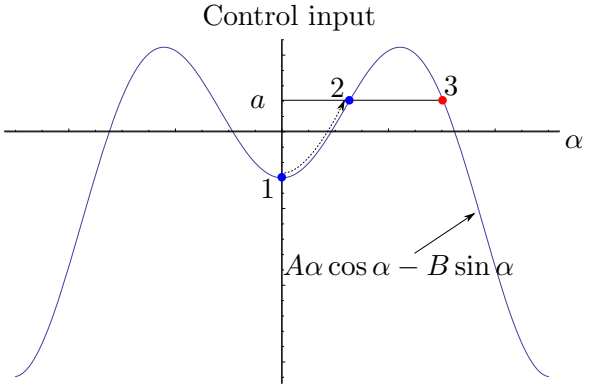


Figure 2.6: Relation between the control input a and the angle of attack α

The iteration starts by increasing α from zero gradually until the solution α for the command a is found. In figure 2.6, we start from moving the blue dot number 1 along the example graph $A\alpha\cos\alpha - B\sin\alpha$ until the solution $A\alpha\cos\alpha - B\sin\alpha = a$ is reached (blue dot number 2) but if the iteration continues, another solution, the red dot number 3, is found. In reality, only the first solution can maintain the stability of the UAV while the later solutions are found after the vehicle becomes unstable.

2. Maximum angle of attack at the actuators limit $\alpha_{\text{max}}^{\text{stb}}$ is the maximum angle of attack being able to maintain the velocity and stability of the vehicle. This value depends on altitude and speed of the UAV. It is usually given by wind tunnel experiments.

3. Angle of attack $\alpha_{\text{max}}^{\text{struct}}$, which yields the maximum normal acceleration $a_{\text{L}_{\text{max}}}$, is parametrised by structural limit of the UAV. It is related to the maximum effort that the UAV can take without deforming its structure. This angle of attack can be calculated by solving this equation:

$$a_{\text{L}_{\text{max}}} = \frac{QS}{m} C_{\text{N}}(\alpha_{\text{max}}^{\text{struct}}).$$

If $C_{\text{N}} = C_{\text{N}\alpha}\alpha$ is considered, then

$$\alpha_{\text{max}}^{\text{struct}} = \frac{m a_{\text{L}_{\text{max}}}}{QSC_{\text{N}\alpha}}.$$

Dynamic response: The dynamic response of α is also considered and modeled as a second order system:

$$\frac{\alpha}{\alpha_c} = \frac{1}{1 + \frac{2\zeta}{\omega_0}p + \left(\frac{1}{\omega_0}\right)^2 p^2}, \tag{2.9}$$

with a damping ratio ζ and an angular frequency ω_0.

This dynamic response represents a hierarchical controller whose inner loop stabilizes the rotational velocity of the UAV and the outer loop controls the dynamics of the angle of attack α [DSF00].

As the angle of attack α is a parameter using to control the UAV. The UAV cannot directly move in the sideways direction. Thus, the UAV system is nonholonomic.

2.2 Environmental modeling

2.2.1 US-76 model

The U.S. Standard Atmosphere has been the work of the U.S. Committee On Extension to the Standard Atmosphere (COESA) since 1953. There are several versions: 1953, 1958, 1962, 1966, and 1976. These models were published in book form jointly by the National Oceanic and Atmospheric Administration (NOAA), the National Aeronautics and Space Administration (NASA), and the U.S. Air Force. These models are obtained based on rocket and satellite data and perfect gas theory by cooperation of 30 U.S. organizations representing the government, industry, research institutions, and universities. The U.S. Standard Atmospheres 1976 (US-76 model) represent the atmospheric densities and temperatures from sea level to 1000 km. Below 32 km the U.S. Standard Atmosphere is identical with the Standard Atmosphere of the International Civil Aviation Organization (ICAO). The US-76 consists of single profile representing the idealized, steady-state atmosphere for moderate solar activity. The altitude resolution varies from 0.05 km at low altitudes to 5 km at high altitudes shown in figure 2.7. In the lower earth atmosphere (altitude < 35 km), density of air and atmospheric pressure decrease exponentially with altitude and approach zero at about 35 km.

The closed-form solutions of this model are as follows:
For $h > 25$ km (Upper Stratosphere):

$$\begin{aligned} T_{\text{atm}} &= -131.21 + 0.00299h, \\ p_{\text{atm}} &= 2.488 \left[\frac{T_{\text{atm}} + 273.1}{216.6}\right]^{-11.388}. \end{aligned}$$

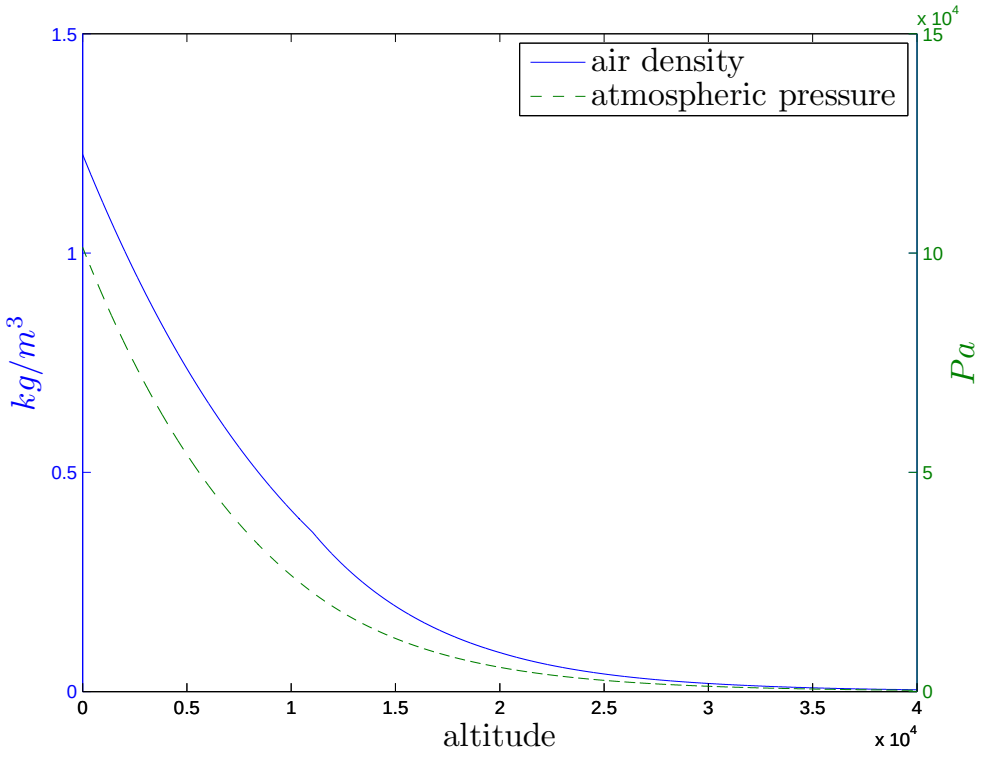


Figure 2.7: The variation of air density and atmospheric pressure with altitude: US-76 model

For 11 km $< h <$ 25 km (Lower Stratosphere):

$$T_{\text{atm}} = -56.46,$$
$$p_{\text{atm}} = 22.65 \exp^{-1.73-0.000157h}.$$

For $h <$ 11 km (Troposphere):

$$T_{\text{atm}} = 15.04 - .00649h,$$
$$p_{\text{atm}} = 101.29 \left[\frac{T_{\text{atm}} + 273.1}{288.08}\right]^{5.256}.$$

with $\rho = p_{\text{atm}}/(0.2869(T_{\text{atm}} + 273.1))$ where ρ is the air density (kg/m^3), p_{atm} is the atmospheric pressure (kPa), T_{atm} is the atmospheric temperature (°C) and h is the altitude (m).

2.2.2 Simplified environmental model

The simplified air density function can be expressed as

$$\rho(z) = \rho_0 e^{-z/z_r}, \tag{2.10}$$

where ρ_0 is the air density at standard atmosphere at sea level and z_r is the reference altitude.

As a consequence, the path curvature can be written in the same way as

$$c(z) = c_0 e^{-z/z_r}. \tag{2.11}$$

where c_0 is the maximum curvature at sea level.

2.3 Problem formulation

Let $\mathbf{x}(t) = (\boldsymbol{\xi}^\top, v, \gamma, \chi)^\top \in \mathbb{X} = \mathbb{R}^6$ be the measurable state of the system where $\boldsymbol{\xi} = (x, y, z)^\top$ be the vehicle position, v be the vehicle speed, γ be the flight path angle, χ be the azimuth angle, $\mathbf{u} \in \mathbb{U}$ be a control input, including aerodynamic and propulsion forces, in the set $\mathbb{U}$ of admissible controls. Then, the differential system (2.8) can be rewritten as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{2.12}$$

where $\mathbf{f}$ is the vehicle system model.

$\mathbb{X}$ is the state space. It is divided into two subsets. Let $\mathbb{X}_{\text{free}}$ be the set of admissible states. $\mathbb{X}_{\text{obs}} = \mathbb{X} \setminus \mathbb{X}_{\text{free}}$ is defined as the obstacle region. The initial state of the system is $\mathbf{x}_{\text{init}} \in \mathbb{X}_{\text{free}}$.

The trajectory planning algorithm is given a rendezvous set $\mathbb{X}_{\text{goal}} \subset \mathbb{X}_{\text{free}}$. In order to achieve its mission, the vehicle has to reach $\mathbb{X}_{\text{goal}}$ while avoiding obstacles and minimizing a performance criterion J defined as

$$J(t_0, t_f, \mathbf{u}) = \int_{t_0}^{t_f} f^0(\mathbf{x}(t), \mathbf{u}(t))\, dt + g(\mathbf{x}(t_f)), \tag{2.13}$$

where $f^0 : \mathbb{R}^6 \times \mathbb{R}^2 \to \mathbb{R}$ and $g : \mathbb{R}^6 \to \mathbb{R}$ are C^1 [Tré12]. Note that if the minimal time problem is considered, $f^0 = 1$ and $g = 0$ are chosen, and if the minimal maximum final speed is considered, $f^0 = 0$ and $g = -v(t_f)$ are chosen.

To sum up, the trajectory planning problem is to find a collision free trajectory $\mathbf{x}(t) : [0, t_f] \to \mathbb{X}_{\text{free}}$ with $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, that starts at $\mathbf{x}_{\text{init}}$, reaches the goal region $\mathbb{X}_{\text{goal}}$, *i.e.* $\mathbf{x}(0) = \mathbf{x}_{\text{init}}$ and $\mathbf{x}(t_f) \in \mathbb{X}_{\text{goal}}$ and minimizes the cost function J. The function $\mathbf{f}$ is defined by the dynamics of the UAV.

2.4 Conclusions

To conclude, the objective of this thesis is to find an optimal or near-optimal trajectory generation algorithm for rendez-vous problems of UAVs. This is a very challenging problem because:

- Some UAVs are hybrid systems such as missiles that have a propulsive phase and a non-propulsive phase
- The environment is complex and cluttered with obstacles
- The goal area must be reached with optimal criteria such as shortest distance or maximum velocity
- The possibility of replanning is required if there are some changes during the mission
- At the end, the method must be implementable in a real time system

In the next part, the state of the art of UAV flight controls and trajectory planning methods are presented.

Part II

State of the art

3 UAV flight controls

A flight control system is an important module for any aerial vehicles. The UAV flight control systems must provide such technologies or techniques that ensure that the UAVs can fulfill their missions. The missions of the UAVs can be to follow a desired trajectory in presence of external perturbations or to perform risky tasks under extreme flight conditions that are impossible for the manned aerial vehicles. Therefore, the UAV flight control systems must consider safety or mission critical, such as error in model and measurement, obstacle avoidance, etc. A classical UAV control system architecture is shown in figure 3.1. The general goal of the UAV is to track the desired trajectory or commands in presence of external and internal perturbations.

In this chapter, several general control techniques used to design the control systems for the UAV flight controls are presented. There are five main methods: linear flight control techniques, nonlinear flight control techniques, specific guidance laws, numerical methods, and artificial intelligent techniques.

3.1 Linear flight control techniques

The linear flight control techniques are used on linear systems which are unlikely for UAV systems. In order to apply these techniques for UAV systems, the dynamics of the vehicle must be linearized. The linear flight control techniques are widely used due to their simplicity, easy implementation, and associated metrics of stability and performance. Several linear control techniques are presented in this section.

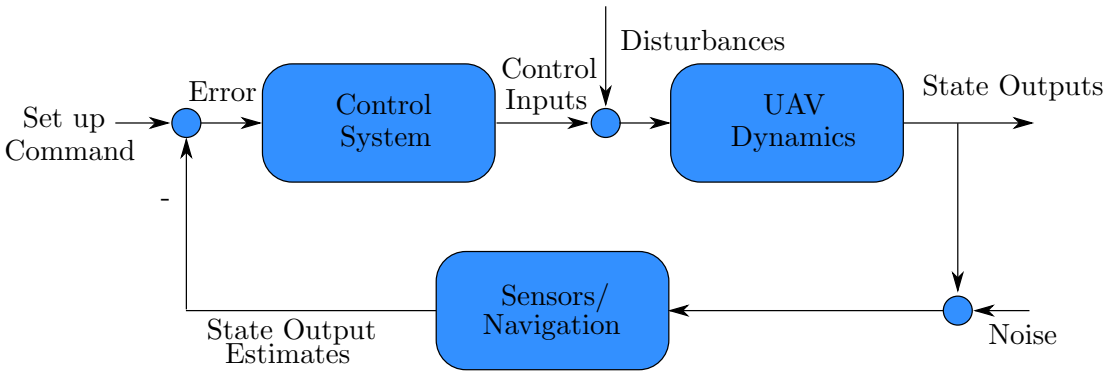


Figure 3.1: A classical UAV control system architecture

3.1.1 Classical and PID controllers

These controllers are used for single-input-single-output (SISO) loop which is very similar to the control structure in figure 3.1. Considering a SISO loop, three standard forms of the classical controllers can be used to design the control system:

- Proportional feedback: this controller is used to modify the gain of the system
- Integral feedback: the steady-state error is typically corrected by this controller
- Derivative feedback: this controller provides the feedback of the error change rate so that the future error can be anticipated

These three standard controllers can be combined with one another. The combination of three feedback controller is the standard PID controller. The robust PID controller is designed for a UAV flight control system in [KG11]. The PID controllers are easy to be implemented. However, the PID controllers have to balance the gains among all three controllers. This can impact the entire system resulting in settling time and oscillations for example.

3.1.2 Successive loop closure

If the dynamics of the vehicle is complex, the problems have to be decomposed to identify components of the dynamics that are controlled by each specific controller. For each controller, the classical controller is applied. Then, the successive loop closure is performed where the reference of each control loop comes from another control loop. An example of successive loop closure is the system decomposed into 3 controllers: position, velocity and attitude. The input of the system is the desired position which is given to the position controller plant. Then, the output of the position controller plant is used as a reference velocity for the next velocity controller plant and so on.

3.1.3 Multi-Input-Multi-Output control

In the actual UAV system, there are multiple sensors and actuators. This makes the system a multi-input-multi-output (MIMO) problem. Several techniques can be used to handle the additional complexity of the MIMO.

- **Full state feedback controller:** Linear-Quadratic Regulator (LQR) and Linear-Quadratic-Gaussian (LQG) controllers have been used in several applications of UAVs. The performances of the LQR and PID applied to an indoor micro quadrotor are analyzed and compared in [BNS04]. The LQR is also used together with the Kalman estimator to control the longitudinal dynamics of the UAV in [HV13]. In [MBB05], the LQG controller is used in a helicopter application.
- **Robust Control techniques:** H_2, H_∞, and Quantitative Feedback Theory (QFT) are the robust control techniques used for MIMO control problems. The combination of H_2 and H_∞ is used on the landing control of a flying-wing UAV in [WZS07]. The QFT and its application in UAV flight control can be found in [XY12].

The linear flight control techniques are very simple and easy to implement. The linear flight control techniques also works effectively for simple missions such as attitude-hold navigation, non-agile waypoint navigation, etc. Although the linear control techniques can still be used for linearized dynamics, they can only guarantee to be locally stable. The H_∞ can usually extend the stability area around the equilibrium point in exchange for a degraded performance. Thus, the desired performance may be difficult to achieve using the linear flight control techniques. Moreover, with a growth in aeronautic and aerospace technologies, the aerial vehicles become more capable of executing difficult missions involving significant model and environment uncertainties. Thus, the linear flight control techniques are not sufficient to solve these problems.

3.2 Nonlinear flight control techniques

Since the linear flight control techniques are not sufficient to solve the modern UAV problems, the nonlinear flight control techniques are introduced. In order to use these techniques, the UAV dynamics are basically considered nonlinear and can be written in a generic form as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{3.1}$$

where $t \in \mathbb{R}^+$ denotes the time, $\mathbf{x}(t) \in \mathbb{R}^n$ is the n-dimension state of system, and $\mathbf{u}(t) \in \mathbb{R}^m$ is the control inputs of m dimensions.

Several techniques can be used to solve the problems modeled with this dynamic.

3.2.1 Feedback-based control

- **Gain scheduling:** Gain scheduling approach [Lei99] can be classified in both linear and nonlinear control techniques. The gain scheduling treats linear or nonlinear problems by decomposing the problems into a finite number of linear or nonlinear subproblems. This approach tries to find a set of gains associating to each sub-model and applies the corresponding (linear or nonlinear) control law. However, the stability of the closed-loop control system must be guaranteed; for example, using Lyapunov function. Then, the control design task is to select the gain that satisfies the stability of the system. This results in switching between sub-models in the closed-loop control system.

- **Backstepping:** Backstepping [Kok92] is a recursive approach to stabilizing the systems that can be represented in cascade, *i.e.* control loops within control loops. The idea is very simple. Starting from the inner-most loop whose stabilizing control law and a Lyapunov function are known, an integrator is added to its input. Thus, a new stabilizing control law with a new Lyapunov function is explicitly designed for the augmented systems. It can be applied in several types of aerial vehicles [MH04, GHM08, JT08, RA05].

3.2.2 Sliding mode control

Sliding mode control or SMC [Zin90] is a variable structure control (VSC) methods. The nonlinear feed-back control has one or several discontinuities in the state space. The multiple control structures are designed so that, when its state crosses these discontinuities, the control structure is altered to a different control structure. The SMC is developed to deal with uncertainties. It stabilizes the dynamics by switching the control structure. Several UAV flight controls are designed based on this method such as the SMC based on the backstepping approach of a UAV type-quadrotor demonstrated in [BBT07] and the adaptive multiple sliding surface control for a missile guidance in [WHD10].

3.2.3 Model Predictive Control

Model Predictive Control or MPC [GPM89] is an advanced method combing open loop optimal control with feedback control. It is a multivariable control algorithm designing an admissible continuous control input by minimizing the error between the actual state of system and the reference model without violating any given constraints in a short time horizon. The main advantage of this method is that it optimizes a current horizon while

considering the future horizon, *i.e.* it is locally optimal. The applications of the MPC on the UAVs are shown in [BHPL06, YS12].

Note that there are some unmentioned techniques that can be used as a nonlinear flight control techniques; for example, singular perturbation techniques [Kok81, SOK84].

To conclude, the nonlinear control techniques work very well to solve the nonlinear problems of the UAVs. Each one of them has the advantages over the specific types of objectives. For example, the SMC is very good and robust to deal with uncertainties while the MPC can deal very well with problems with constraints imposed on the system. The advantage over the global optimization is that solutions can be obtained rapidly and is possible to implement to the real time system. However, some of them do not consider the future conditions and environment. Therefore, in a dynamic environment or environment cluttered with obstacle, a feasible trajectory might not be found by these methods. Even if some techniques such as the MPC can consider the future conditions and environment, they are only with in the local optimization horizon using a reference trajectory. If the reference trajectory is too far from the feasible trajectory, the MPC might have difficulties in finding a feasible trajectory and ends up in some obstacles. Thus, it is preferable that future conditions and environment are considered globally.

3.3 Specific guidance laws

Specific guidance laws are developed to solve specific problems such as missile guidance problems. In this section, some specific guidance laws are described. The interesting fact of these guidance laws is that they have closed-form solutions that are easy to implement on board the vehicles. As it is not easy to obtain the closed-loop control for nonlinear systems, a lot of open-loop guidance laws are designed to solve nonlinear problems. The objective of guidance is to direct the vehicle toward a given destination. In most of guidance laws, in order to view an object or destination, one must sight along a line to that object or destination. This line is called the line-of-sight (LOS) (see figure 3.2).

The LOS that passes through the objective of the guidance is an important concept of guidance. Most of the classical guidance laws use this LOS as basis for their calculation. Here, several guidance laws are explained.

3.3.1 Classical guidance

The classical guidance laws are guidance techniques which are a part of the postlaunch phase. In other words, they are the guidance laws applied directly to the aircraft during

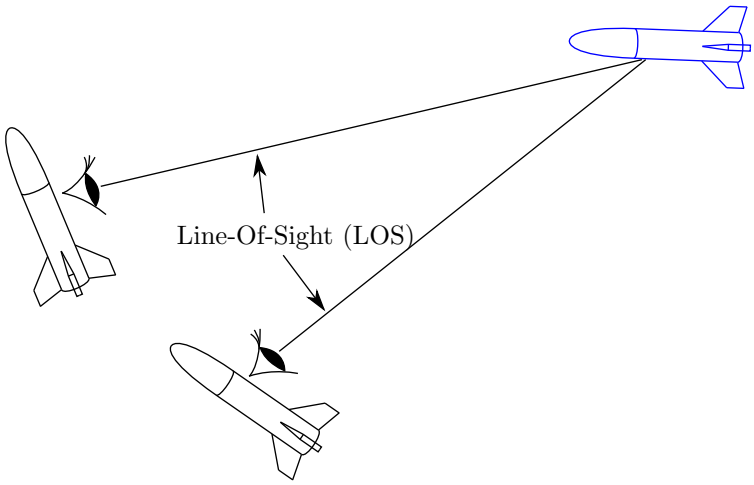


Figure 3.2: Illustration of line-of-sight (LOS)

the mission. There are some well-known guidance laws such as Beam rider guidance [Zar94], Command to line-of-sight guidance [Hea52], Pursuit guidance [Zar94], etc.

Proportional navigation guidance [Sio04] known as PN or PNG is a guidance law which is widely used in interceptor missile guidance in practice.

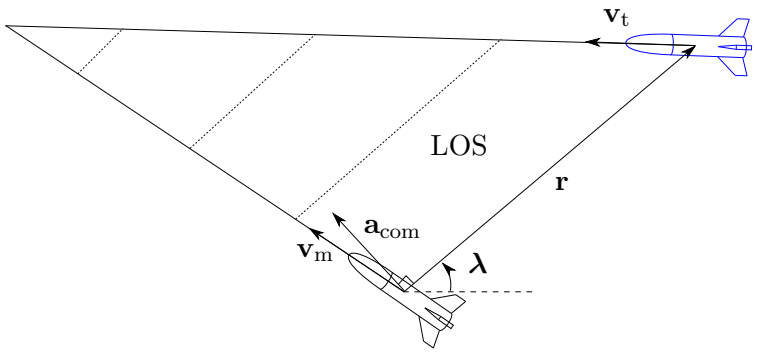


Figure 3.3: Illustration of proportional navigation

The basic philosophy behind PN is that the missile velocity vector should rotate at a rate proportional to the rotation rate of the line-of-sight (LOS-rate), and in the same direction. Specifically, the classical PN tries to nullify the heading error for intercepting the target. The PN is illustrated in figure 3.3. $\mathbf{v}_\mathrm{t}$ is the target velocity, $\mathbf{v}_\mathrm{m}$ is the missile velocity, $\boldsymbol{\lambda} = (\lambda_x, \lambda_y, \lambda_z)$ is the Euler angles of LOS, $\mathbf{r}$ is the vector representing the distance between the target and the missile, and $\mathbf{a}_\mathrm{com} = (a_{\mathrm{com}_x}, a_{\mathrm{com}_y}, a_{\mathrm{com}_z})$ is the commanded acceleration of the missile acting perpendicular to the instantaneous LOS. Mathematically, proportional navigation can be expressed as

$$\mathbf{a}_\mathrm{com} = N\mathbf{v}_\mathrm{c} \times \left(\frac{d\boldsymbol{\lambda}}{dt}\right) \tag{3.2}$$

where N is the navigation constant, $\mathbf{v}_\mathrm{c} = (v_{c_x}, v_{c_y}, v_{c_z}) = \mathbf{v}_\mathrm{m} - \mathbf{v}_\mathrm{t}$ is the relative velocity of the missile to the target.

After some straightforward calculation and some assumptions, the equation (3.2) can be rewritten as (see the demonstration in [Sio04])

$$\mathbf{a}_\mathrm{com} = N\frac{\mathbf{ZEM}}{t_\mathrm{go}^2} = N\frac{\mathbf{r} - \mathbf{v}_\mathrm{c}t_\mathrm{go}}{t_\mathrm{go}^2} \tag{3.3}$$

where $\mathbf{ZEM} = \mathbf{r} - \mathbf{v}_\mathrm{c}t_\mathrm{go}$ is the zero-effort-miss, *i.e.* the miss distance calculated at any time t assuming that the missile and the target do not maneuver or accelerate after the moment t until their collision at t_f, and $t_\mathrm{go} = t_f - t$ is the time-to-go until the end of flight.

By interpreting **ZEM** as a predicted intercept point (PIP), which can be calculated based on some knowledge, or assumptions, of the future motion of the target, the PN guidance law (3.3) can be considered as predictive guidance.

Proportional navigation and its variants have been treated extensively in the literature. The variations that are worth mentioning are: Pure Proportional Navigation (PPN) [Bec90] , Biased Proportional Navigation (BPN) [MC66] , True Proportional Navigation (TPN) [Gue76] , Generalized Proportional Navigation (GPN) [YY87] , Ideal Proportional Navigation (IPN) [YC92] , and Augmented Proportional Navigation (APN) [Zar94].

Among the mentioned variations of PN, the APN captures attention from many researchers. The APN is developed in order to be applied against the maneuvering target which is the general case of the interceptor missile. Thus, it is the PN that can be used for maneuvering target. It includes an additional term considering the acceleration of the target.

In the APN law, an addition term concerning the target acceleration $\mathbf{a}_\mathrm{t}$ is added to **ZEM** which is

$$\mathbf{ZEM} = \mathbf{r} - \mathbf{v}_\mathrm{c}t_\mathrm{go} + \frac{1}{2}\mathbf{a}_\mathrm{t}t_\mathrm{go}^2 \tag{3.4}$$

Please note that in reality, the target acceleration is not known *a priori*. Thus, the estimation of the target acceleration is required continuously during the flight.

The classical guidance laws are very simple to be implemented because of the closed-form solution. However, they do not consider the aerodynamic drag and some other external factors which can affect the maneuverability of the vehicle. This results in a loss of performance of the guidance laws.

3.3.2 Guidance based on optimal control theory

The typical dynamic system used in optimization problems takes form of general nonlinear differential equations:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) \tag{3.5}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is the state with n_x dimensions, $\mathbf{u} \in \mathbb{R}^{n_u}$ is the control input with n_u dimensions, and $\mathbf{p} \in \mathbb{R}^{n_p}$ is the additional system parameter with n_p dimensions.

The optimal control problem is to minimize the cost function of the system (cf. equation (2.13)):

$$J = \int_{t_0}^{t_f} f^0(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t)\, dt + g(\mathbf{x}(t), \mathbf{p}, t), \tag{3.6}$$

where $f^0 : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}$ are C^1.

The optimal control theory can be derived using the minimum principle of Pontryagin. In order to solve the optimal control problem using this method, the Hamiltonian is required and is defined as

$$H(\mathbf{x}(t), \boldsymbol{\lambda}(t), \mathbf{u}, \mathbf{p}, t) = \boldsymbol{\lambda}^\top(t) f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) + L[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t], \tag{3.7}$$

where $\boldsymbol{\lambda}(t)$ is a vector of the costate variables of the same dimension as $\mathbf{x}(t)$.

The optimal control theory aims to solve this problem with the given initial conditions, terminal conditions, path constraints, and the boundary conditions of the state or costate parameters. Usually, the closed-form solutions can be obtained only when the problem is simplified with less constraints. Here, several optimal control guidance laws with a closed-form solution are presented.

3.3.2.1 Optimal proportional navigation

The optimal control theory is used on the PN to find an optimal navigation gain N_{OPN} [Zar94, Waw02] of the optimal proportional navigation or OPN. Moreover, the dynamics of the first order is included in the study, *i.e.* $\frac{\alpha}{\alpha_d} = \frac{1}{1+\tau p}$ where α is the commanded angle of attack and α_d is its desired value. The optimal control theory tries to minimize the final miss distance and the energies. As a consequence, the optimal commanded acceleration of the missile is obtained.

$$\mathbf{a}_{\mathrm{com}} = N_{\mathrm{OPN}}(\varepsilon)(\mathbf{v}_{\mathrm{r}} \times \frac{d\boldsymbol{\lambda}}{dt} - k_1(\varepsilon)\mathbf{a}_{\mathrm{m}} + \frac{1}{2}\mathbf{a}_{\mathrm{t}}) \tag{3.8}$$

with

$$\varepsilon = t_{\mathrm{go}}/\tau$$
$$k_1(\varepsilon) = (e^{-\varepsilon} + \varepsilon - 1)/\varepsilon^2$$
$$N_{\mathrm{OPN}}(\varepsilon) = \frac{N(\varepsilon)}{\frac{6\mu}{\tau^3} + D(\varepsilon)}$$

where

$$N(\varepsilon) = 6\varepsilon^2(e^{-\varepsilon} + \varepsilon - 1)$$
$$D(\varepsilon) = 2\varepsilon^3 - 6\varepsilon^2 + 3 - 12\varepsilon e^{-\varepsilon} - 3e^{-2\varepsilon}$$
μ is a constant
$\mathbf{a}_\mathrm{m}$ is the actual acceleration of the missile
$\mathbf{a}_\mathrm{t}$ is the acceleration of the target

The OPN only concerns the first two terms of equation (3.8) while the optimal augmented proportional navigation is the entire equation (3.8) which includes the acceleration of the target. According to equation (3.8), if $\tau = 0$ and $\mu = 0$, we obtain the gain $N_{\mathrm{OPN}}(\varepsilon) = 3$ which is the optimal gain that is usually used for the proportional navigation (N in equation (3.3)). Please note that $\tau = 0$ implies that the dynamics of the angle of attack is perfect and $\mu = 0$ implies that the final miss distance is null.

3.3.2.2 Kappa guidance and optimal kappa guidance

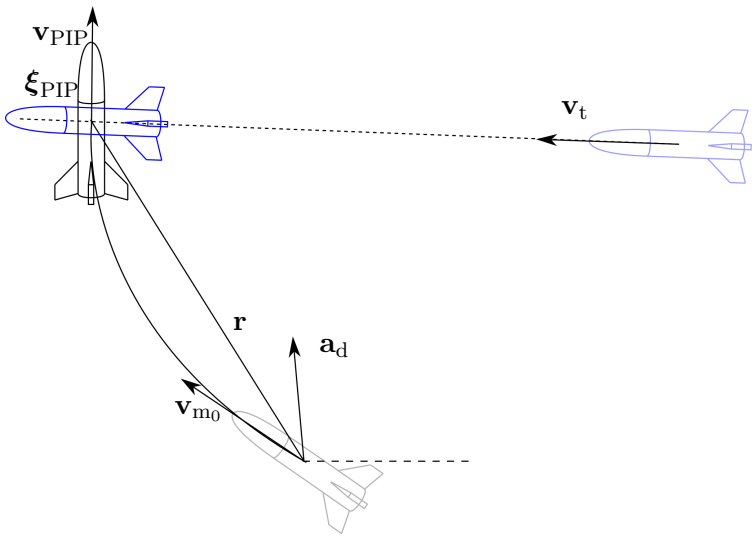


Figure 3.4: Illustration of kappa guidance

Kappa guidance [Lin91] is a guidance law designed using the optimal control theory to maximize the final speed of the vehicle by controlling the curvature of the vehicle trajectory. The kappa guidance is another famous law used in missile guidance.

The kappa guidance uses a form of proportional navigation guidance to ensure a minimization of the energy loss along with another term to direct missile velocity to the demanded terminal angle of approach. Thus, the constraint of the final approach angle is

applied to this guidance law. Then, the minimum principle of Pontryagin is used to derive the guidance solutions.

The explicit guidance is used as a basis of the kappa guidance. The desired commanded acceleration vector $\mathbf{a}_{\text{com}}$ of the explicit guidance can be expressed explicitly as

$$\mathbf{a}_{\text{com}} = \mathbf{a}_1 + \mathbf{a}_2, \tag{3.9}$$

with

$$\begin{aligned} \mathbf{a}_1 &= \frac{K_1}{t_{\text{go}}}(\mathbf{v}_{\text{pip}} - \mathbf{v}_{\text{m}_0}), \\ \mathbf{a}_2 &= \frac{K_2}{t_{\text{go}}^2}(\mathbf{r} - \mathbf{v}_{\text{c}} t_{\text{go}}), \end{aligned} \tag{3.10}$$

choosing $||\mathbf{v}_{\text{pip}}|| = ||\mathbf{v}_{\text{m}_0}||$, K_1 and K_2 are the gains.
The second term $\mathbf{a}_2$ is the proportional navigation term which nullifies the heading error to the PIP while the first term $\mathbf{a}_1$ is the shaping term which rotates the flight path angle so that the initial or current flight path angle $\gamma(\mathbf{v}_{\text{m}_0})$ converges to the final or desired flight path angle $\gamma_f(\mathbf{v}_{\text{m}_f})$.

In [Lin91], Lin also studied the optimal kappa guidance in order to find the optimal gains K_1 and K_2. The approximation of these gains are shown in appendix A. At the boundary conditions, the gain $K_1 = -2$ and $K_2 = 6$. These gains are usually used for the terminal missile guidance.

In general, kappa guidance is a PN with a trajectory shaping term which considers the final flight path angle. As kappa guidance is developed based on the PN, its performance also degrades sharply in the presence of rapidly maneuvering targets and large off-boresight angle launches.

3.3.2.3 Dubins' paths

Dubins' paths are developed based on a Dubins' car model. They are usually used in terrestrial mobile robotics. Even though Dubins' car does not have the exact same model as aerial vehicles, they are used in some aerial vehicle applications. Thus, it is interesting to study the Dubins' paths.

Instead of searching for a set of control inputs for the vehicle, Dubins supposes that the vehicles are always operated with the maximal control inputs in order to achieve the shortest path. With this hypothesis, the Dubins' paths are developed [Dub57] based on Dubins' car model. Dubins' paths are calculated using the optimal control method for 2-dimensional problems.

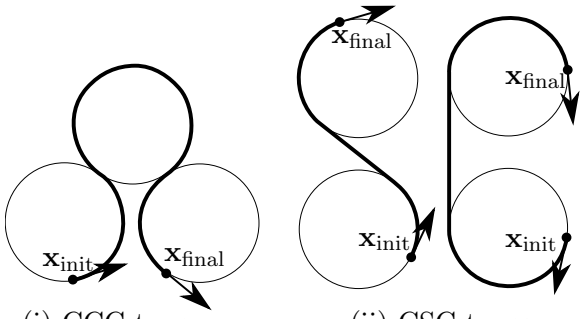
In the study of Dubins, the minimal length path is composed of two types of trajectories: an arc of circle (C) and a segment (S). This is studied and developed again by Boissonat's

team based on the minimal principle of Pontryagin [BCL91]. In their study, the system of Dubins' car is expressed as

$$\begin{cases} \dot{x} = \cos\theta, \\ \dot{z} = \sin\theta, \\ \dot{\theta} = u, \quad u \leqslant 1, \end{cases} \tag{3.11}$$

where $(x, z) \in \mathbb{R}^2$ is the vehicle position, $\theta \in \mathbb{R}$ is the path angle, and $u \in [-1, 1]$ is the control input.

The optimization problem for the Dubins' car is to minimize the total time of the trajectory, *i.e.* $J = t_f$. Boissonat's team has proved that the optimal path between two states cannot be any combinations of arcs of circle (C) and segments (S). The optimal path must be one of these two types: CSC (arc-segment-arc), CCC (arc-arc-arc) or their degenerate forms (C, S, CS, SC). They also state that, for the CCC path, the angle of the second arc must be superior to π. The CSC and CCC types are illustrated in figure 3.5.



(i) CCC type (ii) CSC types

Figure 3.5: Dubins' paths

The Dubins' paths are optimal and work very well for the 2-dimentional nonholonomic problems. However, there is a discontinuity of the control input at the section between C and S path, *i.e.* a rapid change of u from 1,-1 to 0 and vice versa. Thus, it is not suitable to be used as a guidance law. In the other hand, it can be used as a metric to determine the distance between two states of a nonholonomic robot/vehicle.

The guidance laws presented in this section have the closed-form solutions. Thus, they are very easy to be implemented in real-time systems. However, these guidance laws rely on some restrictive approximations such as very simplified model. Moreover, no future condition is considered such as the changes of control limitation or environment. For such complex systems and missions, the optimal problem needs to be considered globally.

3.4 Numerical methods

Certain trajectory optimization problems are conveniently solved by analytical optimal control such as Dubins' paths [Dub57]. However, if the system has complicated nonlinear dynamics of high dimension and many types of constraints, it is very difficult to obtain an analytical solution. Thus, the numerical solutions of the optimal control problem have been studied and developed. The numerical methods for solving optimal control problems can be divided into two main approaches [Rao09]: indirect and direct approaches, according to an approach of solving the problems shown in figure 3.6.

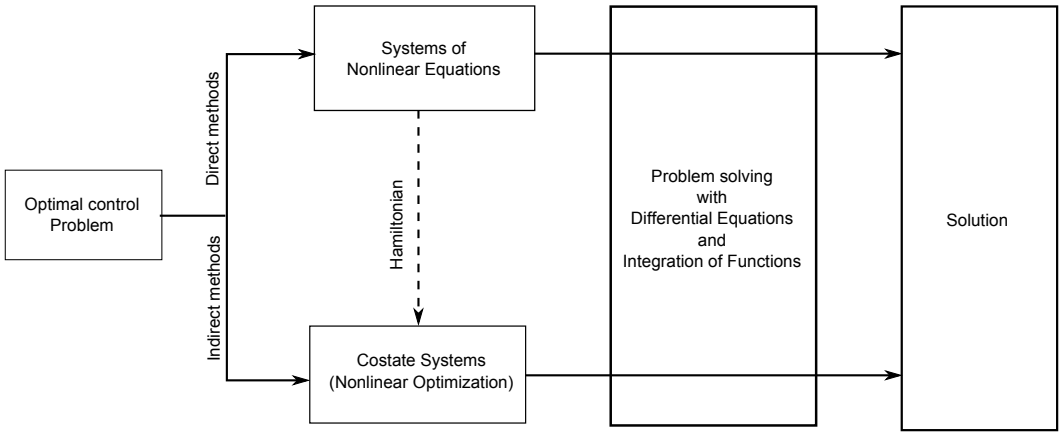


Figure 3.6: Diagram of direct and indirect methods

3.4.1 Indirect approaches

The indirect approaches solve the optimal control problem by solving the Hamiltonian (costate system). These approaches lead to a multiple-point boundary-value problem. This problem is solved to determine candidate optimal solutions called extremals. The extremals can be local minimum, maximum or saddle points. Each computed extremals are compared and the particular extremal with the lowest cost is chosen as an optimal solution. The indirect approaches give an accurate solution. However, there are three major problems [Shi08]. First, the analytical forms of the Hamiltonian and all necessary optimal control conditions must be expressed. Second, the good initial guesses are required. Third, since the initial guess is required, the domain of convergence can be very small.

3.4.2 Direct approaches

Unlike indirect approaches, direct approaches find a solution by means of discretion and parameterization so that the traditional continuous optimal control problem is converted

into parameters optimization problem. This reduces the programming complexity and makes direct approach widely used. Although the direct approaches are less accurate than indirect approaches, the fact that they are easier to implement, have a larger domain of convergence, and have reduced problem size makes them very attractive [Shi08]. The

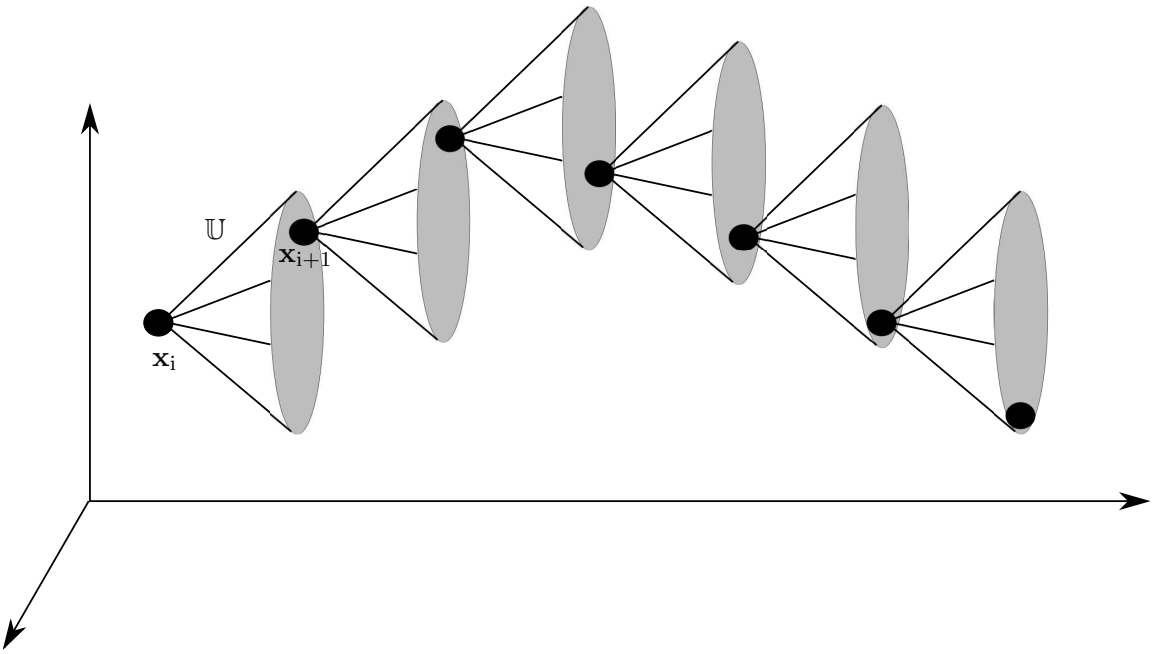


Figure 3.7: Schematic description of the differential inclusion approach

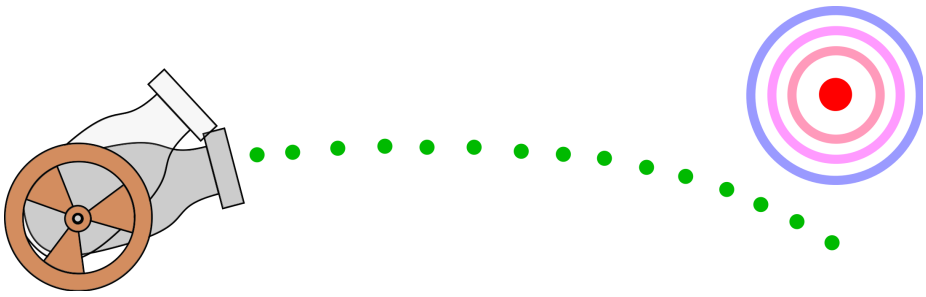
differential inclusion [Sey93] is a method using the direct approach. It enforces the system at each discrete node by applying inequality constraints on the state derivatives. These inequality constraints are obtained by substituting the upper and lower bounds on the control input into the equations of motion. When the inequality constraints are met, the states at $\mathbf{x}_{i+1}$ are said to lie in the attainable set at that node given the state values at an adjacent node $\mathbf{x}_i$ and the set of admissible controls $\mathbb{U}$ (see figure 3.7). The advantage given by differential inclusions is that it effectively eliminates the explicit dependence on control values at each node. This method was demonstrated on 1-D rocket ascent (Goddard problem [God19]) in presence of a dynamic pressure constraint in [Sey93]. It was also demonstrated on a flight maneuver simulation in [Rac]. However, these methods can become numerically unstable and the formulation can be problem dependent [Bet98].

3.4.3 Methods used in solving indirect and direct approaches

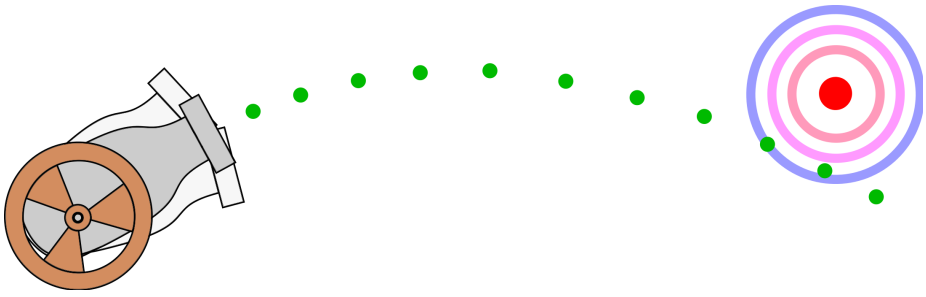
There are several methods in solving both indirect and indirect approaches.

3.4.3.1 Shooting and multi-shooting methods

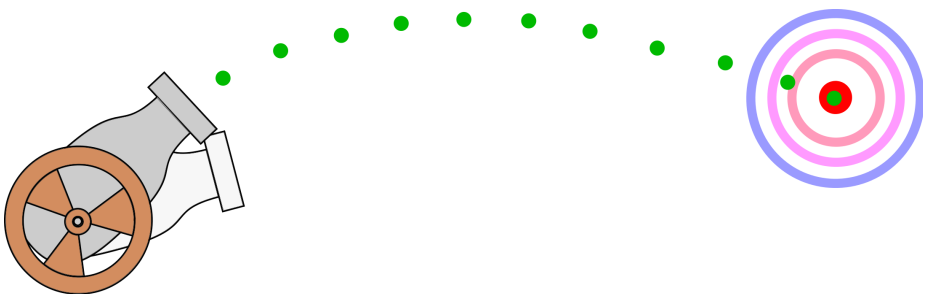
The idea of the shooting method is to reduce a boundary value problem to an initial value problem. In the typical shooting method [Kel76], an initial guess is made of the unknown boundary conditions at one end of the interval. By using this guess along with the known initial conditions, the system is integrated to the other end. Upon reaching the other end, the obtained terminal conditions are compared to the known terminal conditions. If the obtained terminal conditions differ from the known terminal conditions by more than a specified tolerance, the unknown initial conditions are adjusted and the process is repeated until the difference between the obtained terminal conditions and the known terminal conditions is less than the specified tolerance. This method is illustrated by using the analogy of a canon firing at a target in figure 3.8. The shooting method is very simple.



(a) First try



(b) Second try



(c) Last try

Figure 3.8: Schematic of shooting methods using the analogy of a canon firing at a target

However, there are significant numerical difficulties due to ill-conditioning of the system dynamics. As a result, errors made in the unknown boundary conditions will amplify as the dynamics are integrated in either direction of time. The shooting method possesses particularly poor characteristics when the optimal control problem is hyper-sensitive, *i.e.* when the time interval of interest is large in comparison with the time-scales of the system in a neighborhood of the optimal solution [Rao09].

The multi-shooting method [SB02] is proposed in order to overcome these difficulties. Instead of integrating the system over a single large time interval, the multi-shooting method applies the shooting method over each subinvertal divided from that single large time interval. It is the improved version of shooting method because the sensitivity to errors in the unknown initial conditions is reduced due to the integration over significantly smaller time intervals. Nevertheless, a sufficiently good initial guess is still required. This method is applied in the domain of missile in [Sub11, Tau02].

3.4.3.2 Collocation methods

Collocation methods enforce the equations of motion through quadrature rules or interpolation. An interpolating function (interpolant) is solved in such a way that it passes through the state values and maintains the state derivatives at the nodes spanning subinterval of time. Collocation methods can be divided into local collocation and global collocation. The local collocation is similar to the multi-shooting method that the time interval is divided into several subintervals while the global (orthogonal) collocation considers the problem globally.

Pseudospectral method is a global form of orthogonal collocation, *i.e.* the state is approximated using the global polynomial and collocation is performed at chosen points. The majority of pseudospectral methods use global orthogonal Lagrange polynomials as the interpolant while the different nodes are selected as the roots of the derivative of the polynomial such as Legendre pseudospectral method [EKR95], Chebyshev pseudospectral method [ER98].

General purpose optimal control software or GPOPS [RBD$^+$10] was developed in 2010 by Anil V. Rao and his team to solve multiple-phase optimal control problems using a Gauss pseudospectral method. This software is very well known tool in optimal control problem solving. The Gauss pseudospectral method has been studied for the anti-ship missile trajectory generation in [ZZZL11] by using GPOPS.

3.5 Artificial intelligence techniques

The concept of techniques in this group is to implant knowledge into a machine (algorithm) so that it can solve the problems by itself. It is more likely to create an artificial intelligence or a pattern to solve a problem. Techniques classified in this category are as followed.

3.5.1 Genetic algorithm

A genetic algorithm or GA [Whi] is inspired by the idea of the evolution of living things, *i.e.* Darwinian theory. It simulates the evolution of the initial population of random chromosomes. These chromosomes can reproduce themselves or mutate. Each chromosome may have more or less fitness, *i.e.* desirable objective values. Reproduction and mutation of the chromosomes ensure diversity and give the algorithm a non-zero chance of locating the global optimum of the objective function. The process is repeated for a given number of generations. Then, the individual with the most fitness is selected as a solution.

In trajectory optimization, the genetic algorithm has some attractive features over the gradient methods (direct and indirect methods). Because of its randomness in creating the candidate, it has higher probability of locating the global optimum than gradient-based methods when the cost function has multiple local extrema. However, also because of its randomness, there is no well-defined convergence criterion such as those used in the gradient methods (nonlinear programming). The genetic algorithm is applied to the missile guidance and control in [YA11].

3.5.2 Fuzzy logic

Fuzzy logic [Cox92] solves a problem by posing the conditional questions; for example, if *something* happens, then *do that*. Thus, the knowledge of how to solve each problem is required *a priori*. Fuzzy control [PY98] provides a formal methodology for representing, manipulating, and implementing human's heuristic knowledge about how to control a system. The most important thing in fuzzy logic is how to interpret the knowledge into the logic form. The fuzzy logic has been popular in the recent years and has been applied in the UAV in many ways.

The framework of fuzzy logic based UAV navigation and control can be found in [DVTK04]. It can be used to improve the performance of the existing guidance laws in [LM99, DDG04, GC95]. The combination of fuzzy logic and genetic algorithm on the missile guidance laws is studied in [CSW98]. Lately, the optimization of rules in the fuzzy logic is studied in [LY09].

3.5.3 Neural networks

The neural networks are first proposed in a logical calculus in [MP43]. The neural networks or the artificial neural networks are an information processing paradigm that is inspired by the simulation of the brain. The neural networks learn from experiences and examples. The neural networks solve the problem by combining information from their networks. This makes the neural networks the best in identifying patterns or trends in data. Thus, they are widely used in different domains such as sale forecasting, risk management, target marketing and even in UAV flight controls.

The neural networks are implemented with the existing control techniques to improve their performance like other artificial intelligent techniques. The neural networks applications can be found in [Zho02, ST02, ABSJM07, WC14] for missiles and in [DJ10, Efe11] for UAVs.

The artificial intelligent techniques are not primary techniques for UAV flight controls. They are often used to improve the performance of the existing control laws. Even if, the performance of the new control laws is improved, the improving control laws still inherit some drawbacks or basic properties from the original control laws.

3.6 Conclusion

The principal UAV flight control techniques are presented in this chapter. Linear control techniques are capable of solving simple missions for the local system model. However, difficult missions involving significant model and environment uncertainties are very difficult to solve using linear control techniques. Nonlinear control techniques are more suitable for these missions since they consider the nonlinearity of UAV systems. Some techniques of the nonlinear control techniques; for example the MPC, consider the optimal criteria for the problems. However, the optimal criteria are observed locally in the chosen horizon. The advantage over the global optimization is that solutions can be obtained rapidly and is possible to implement to the real time system. However, if there are some changes in missions or in dynamic environment, the UAVs could end up in the obstacles and the feasible trajectory might not be found. Thus, it is better to consider the problem globally. The guidance laws are very simple and easy to implement. Yet, the future conditions and environment are not considered. The numerical methods give the optimal results but it is time consuming to solve the problem with constraints induced by the obstacles. Even though the artificial intelligent techniques are very interesting, *a priori* knowledge of how to solve the problems is required.

Some of these techniques are interesting to use in the trajectory generation for UAVs.

Still, there are also some attractive techniques used in the robotic field that consider the complete vehicle model in the complex environment. Thus, they are interesting to be introduced in the UAV domain to find a feasible and optimal path for the UAVs in the next chapter.

4 Trajectory planning methods

Recently, autonomous vehicles perform more and more missions in complex environments such as exploration on hazardous terrain or in inaccessible areas. The ideal autonomous vehicles must have a capability to execute tasks without any human intervention in both decision and control. One of the problems is to find a collision-free path in a complex environment with either static or dynamic obstacles. Thus, some knowledge of the surrounding environment is required in order for the system to make a decision. Thus, trajectory planning is an interesting topic for autonomous systems.

In this chapter, several well-known path planning algorithms are generally presented. The organization of this chapter starts by introducing the problem statement of path planning. Frequently used notations are explained along with the objective for path planning algorithms. Then, path planning methods are shown and explained. Some illustrations are presented in order to understand the algorithms easily.

4.1 Problem statement

In path planning, a state or a configuration is denoted $\mathbf{x}$, and a set of all possible state is called a state space $\mathbb{X}$. In order to change from one state to another, the application of actions are chosen by the planner. When each action, denoted $\mathbf{u}$, is applied from the current state $\mathbf{x}$, a derivative with respect to time $\dot{\mathbf{x}}$ is produced as specified by a state transition function $\mathbf{f}$, *i.e.* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. Then, a new state $\mathbf{x}_{\text{new}}$ can be deduced from $\dot{\mathbf{x}}$. A set of all possible applications of actions at the state $\mathbf{x}$ is denoted $\mathbb{U}(\mathbf{x})$. The initial state is denoted $\mathbf{x}_{\text{init}}$.

The objective of path planning is to find a feasible path from the initial state $\mathbf{x}_{\text{init}}$ in the collision-free state space $\mathbb{X}_{\text{free}} \in \mathbb{X}$ to the goal area $\mathbb{X}_{\text{goal}} \in \mathbb{X}_{\text{free}}$ while avoiding obstacles denoted $\mathbb{X}_{\text{obs}} = \mathbb{X} \setminus \mathbb{X}_{\text{free}}$.

4.2 Path planning methods

There are many methods to find a path between $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}} \in \mathbb{X}_{\text{goal}}$. Currently, the path planning algorithms can be categorized as roadmap, cell decomposition, potential field, and sampling-based methods. In this section, most of the basic path planning methods are described along with their advantages and drawbacks. Some of them are presented with illustrations for better understanding of the methods.

4.2.1 Roadmap and cell decomposition methods

The idea of the roadmap is to construct a roadmap by fitting a graph to the state space while cell decomposition methods try to divide the state space into smaller cells. Then, a graph search algorithm is used to find a solution. In general, forward search algorithm is used. It searches the state space starting from the initial state $\mathbf{x}_{\text{init}}$ and other states waiting in queue to go to adjacent states. The search will continue until it reaches the goal state $\mathbf{x}_{\text{goal}}$ or until all states are investigated. Several well-known graph search algorithms are presented in this section.

- **Breadth first:** Breadth-first search [LaV06] uses the first-come first-serve principle to select states in queue, as known as First-In First-Out (FIFO) queue. This search guarantees that the first obtained solution will use the smallest numbers of iterations. However, a large memory space is required since the states stored in queue increase gradually with time. This search algorithm is used widely in electronic design automation.

- **Depth first:** Contrary to breadth-first search, depth-first search [Tar72] uses a Last-In First-Out (LIFO) queue. As a consequence, this search algorithm prefers to expand toward longer paths. This makes the search easily focus on one direction and completely miss large areas of search space. The iterative deepening depth-first search [Kor85] is developed to overcome this problem. The depth-first search algorithm is successfully used in chess programs.

- **Dijkstra:** Dijkstra's algorithm [Dij59, Dij65] is one of the first successful search algorithm which can find optimal paths. It is a special form of dynamic programming [Bel54]. Dijkstra's algorithm introduces the cost function to the search algorithm.

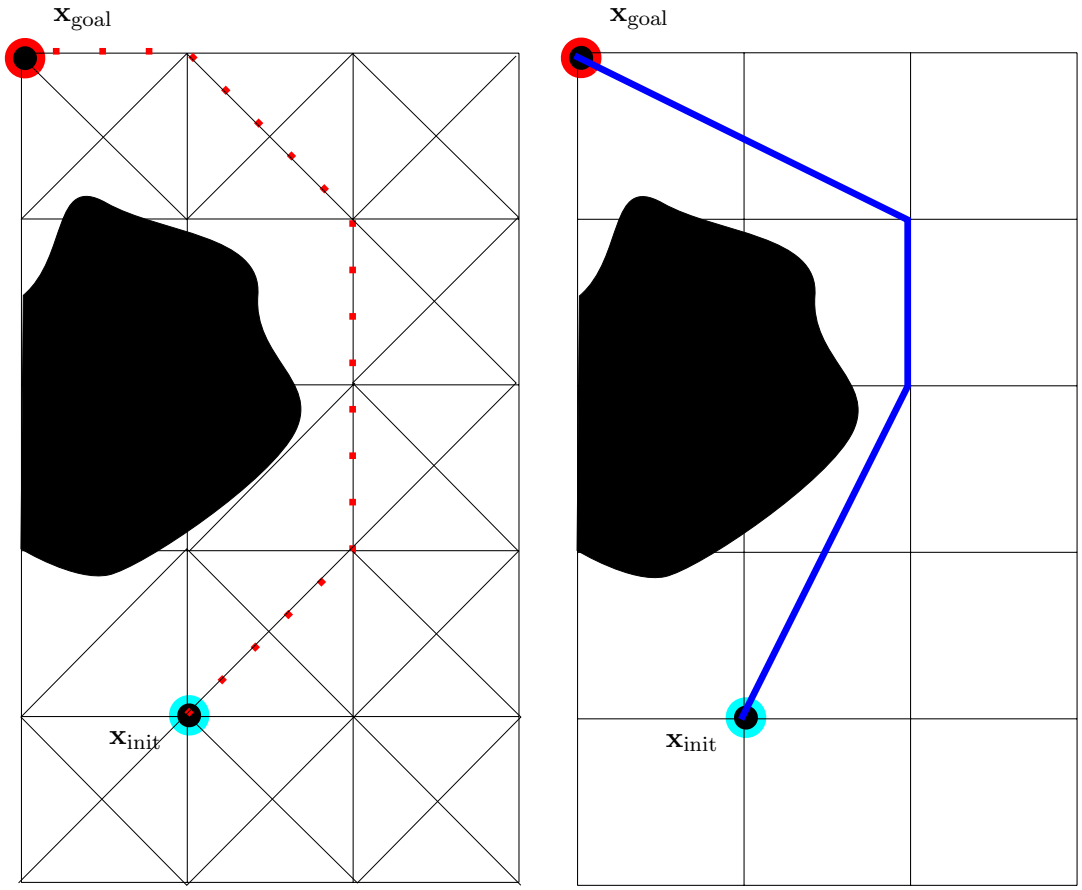
The cost-to-go to each state from the initial state is verified each time. These cost values are used as a sorting function in queue in order to determine in which order states are to expand. Dijkstra's algorithm is guaranteed to find a shortest path from the initial state to the goal state as long as none of the edges have a negative cost. A Dijkstra algorithm for fixed-wing UAV motion planning is presented in [MS10].

- **Best-First-Search:** Best-First-Search algorithm is very similar to Dijkstra's algorithm in the way that it uses the estimated cost function, called a heuristic. This heuristic is used as a sorting function in queue and is determined by how far from the goal state to any state. The heuristic function represents additional knowledge of how to solve the problem, *i.e.* how to reach the goal. One and well-known best-first search algorithm is Greedy Best-First-Search algorithm [RN09]. Instead of selecting the closest state to the initial state, it tries to expand the state that is the closest to the goal by assuming that it is likely to lead to a solution quickly. It runs much faster than Dijkstra's algorithm because it uses the heuristic function to guide its way toward the goal very quickly. However, it is not guaranteed to find the shortest path.

- **A*:** A* [HNR68, RN09] is an extension of Dijkstra's algorithm using heuristics from Greedy Best-First-Search algorithm. It tries to reduce the total number of states explored by incorporating a heuristic estimate of the cost-to-go to the goal from a given state. This algorithm is as fast as best-first-search in the simple problem and can also find the shortest path. The UAV applications using A*-based algorithm are shown in [SJFD08, uVP09, Dic12].

- **D* Lite:** D* [Ste93] or Dynamic A* is an A* algorithm whose cost parameters can change during the problem-solving process. However, the D* algorithm is complex and hard to understand. D* Lite [KL02] which acts the same way as D* but easier to understand is developed. D* and D* Lite have an advantage over the A* algorithm because of their propriety, *i.e.* "dynamic" cost function. Thus, they are more suitable than A* algorithm to be implemented in the real system with conditions that the environment is known in advance.

- **Theta*:** Theta* [NDKF07, DNKF10] is another variant of A*. Theta* is identical to A* except that Theta* and its variants [NKT10] allows the parent of a vertex to be any vertices, unlike A* where the parent must be a successor. Therefore, Theta* and its variants propagate information along grid edges without constraining the paths to grid edges as shown in figure 4.1. Theta* is simple and fast. It can find short and realistic looking paths. However, it is not guaranteed to find the shortest paths. It is applied for a 3D path planning for UAVs in [FGQ12].



(a) A* path (b) Theta* path

Figure 4.1: A* path versus Theta* path

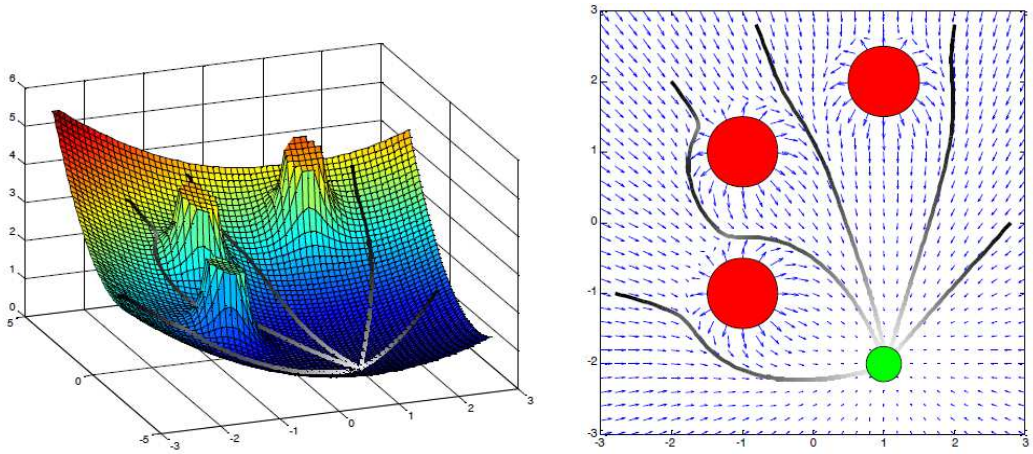
- **Other search methods:** There are some other search methods that use the same search algorithms previously described with the different schemes such as backward search which starts the search from the goal state $\mathbf{x}_{\text{goal}}$ supposing that there is a single goal state or bidirectional search [Poh69] which is a combination of both forward and backward search. A tree grows from the initial state $\mathbf{x}_{\text{init}}$ and another tree grows from the goal state $\mathbf{x}_{\text{goal}}$. Then, the search terminates with success when both trees connect.

The roadmap methods try to fit graphs in the state space while the cell decomposition methods divide the work space into smaller subspace . Then, they search their graphs for solutions. They are very simple and fast to be implemented in low dimensional systems. They are used widely in the robot motion planning and graph searching application. However, the UAV path planning with the constraints of kinematics and dynamics can hardly be solved by these methods effectively. Moreover, implementation of these algorithms for high dimensional problems can be expensive in terms of computational time.

4.2.2 Artificial Potential Field

Artificial Potential Field or APF [AH83, Kha85] uses an idea from nature. By assuming that the vehicle is a charged particle inside the electric or magnetic field, the vehicle is moved by the induced forced guided by the vector field. In robotics, the same effect is simulated by creating an artificial potential field that moves the robot to the desired destination.

There are several types of artificial potential fields; for example, the attractive field and the repulsive field. Attractive field is usually used to guide the vehicle to the desired destination $\mathbf{x}_{\text{goal}}$ while repulsive field is created by the obstacles in order for the robot to avoid them or by the initial state $\mathbf{x}_{\text{init}}$ to make the robot leave its starting point. By combining these two types of artificial potential field, the robot can follow the force induced by the potential field to reach $\mathbf{x}_{\text{goal}}$ while avoiding the obstacles.



(a) Illustration of potential field (b) Illustration of command field

Figure 4.2: Example of potential fields path planning with obstacles [Hel11]

Figure 4.2 shows an example of an APF planning with 3 obstacles and one goal. The black lines represent the path of the robot. The grey scale represents different speed of the robot. In figure 4.2(a), the color gradient represents different magnitude of the APF. In figure 4.2(b), red circles represent repulsive obstacles and the green circle represents the attractive goal.

Despite of how well this method works, the vehicle has some oscillatory movement around the obstacle and is sometimes trapped in the local minima (see figure 4.3). The problems of oscillatory movement can be solved by assigning some distance criteria from the obstacles which may indicate attraction distance. Moreover, several methods have been suggested to deal with the local minima phenomenon in APF. For example, some random movements are used while hoping that these movements will help escape the local minima

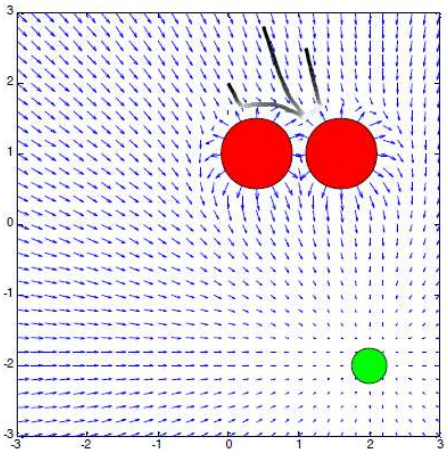


Figure 4.3: The robot is trapped in a local minimum from several different initial positions [Hel11]

in [BL90, BL91]. Or, in [AKH90], potential fields that are solutions to the Laplace equation (harmonic function), which do not have local minima, are used. The APF using harmonic function generates smoother paths, which makes it more suitable to aircraft-like vehicles, than the previous methods. Moreover, it cannot be applied only to a fixed obstacle but also to a moving obstacle. More literature reviews and the application of the APF in robotics can be found in [RK91, RK92, SHSS07]. Navigation functions, a improved version of the APF, that overcome the local minima problem can be found in [FK11, FKA12, FK12].

4.2.3 Sampling-based methods

Sampling-based path planning tries to connect each state in the state space $\mathbb{X}$ by sampling it. For difficult problems, the randomized approach is applied to provide fast solutions. Since the state space is usually formed from Cartesian products, random sampling is the easiest method to generate samples. The sampling based planning is proposed to overcome the complexity of planning algorithms for vehicles of high degree of freedom. However, the solutions are widely regarded as no optimal, or in the better case, suboptimal solution. Some sampling-based algorithms for the honolomic system are presented here.

- **Random walk planner:** Random walk planning builds a search graph by generating a new sample at each iteration. According to a multivariate Gaussian distribution, the new sample is created in the neighborhood of the last successful state whose connecting edge lies in $\mathbb{X}_{\text{free}}$. The covariance parameters of the Gaussian distribution are adaptively tuned according to the success of the last iteration. It is combined

with the APF planner in [CP05]. The drawback of this method is that it is difficult to find a path across the long and narrow corridors.

- **Ariadne's Clew algorithm:** Ariadne's Clew algorithm [MATB96, MAB98] is developed based on the genetic algorithm [Whi]. This algorithm attempts to explore the state space where it has never been before while searching for a solution to $\mathbf{x}_{\text{goal}}$ at the same time. It is unlike the APF that tries to approach the destination at each iteration. Thus, there is no local minima problem. The genetic algorithms are used to solve the optimization for where to place a new state at each iteration. This algorithm is designed to solve a path planning problem in static and dynamic environment. However, these methods require a lot of tuning parameters which is not suitable for path planning.

- **Expansive-space planner:** Expansive-space planner [HLM97, SL03] attempts to explore the state space in the same way as the Ariadne's Clew algorithm with an additional idea borrowing from the bidirectional search scheme. This algorithm samples only the state space that is relevant to the current search graph to avoid the computational cost for the entire state space. This method can solve a lot of problems by using a simple criterion for the placement of states. However, it requires substantial parameter tuning which is specified for each different domain of problems. Moreover, the performance of this method tends to degrade in the long and winding labyrinth.

There are a lot of sampling-based path planning methods that can solve a honolomic problem with a static or dynamic environment but not all of them are capable to solve the problem in general without any parameter tuning.

In the following section, path planning methods for nonholonomic system are presented.

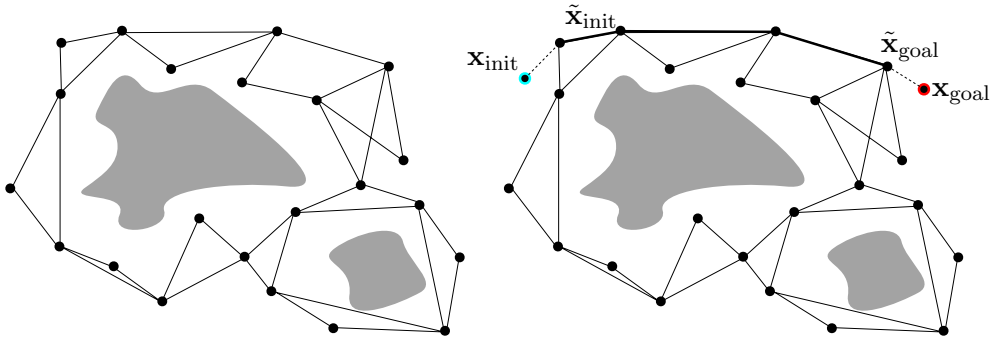
4.3 Path planning methods for nonholonomic system

4.3.1 Probabilistic Roadmap Planner Method (PRM)

Probabilistic Roadmap Planner Method or PRM [KSLO96] is one of the most famous incremental sampling-based path planning methods to plan a collision-free path. The PRM is a two-query path planning methods consisting of a learning/construction phase and a query phase.

An indirect graph or a roadmap is built during a construction phase (see figure 4.4(a)). Then, the query phase attempts to connect the initial state $\mathbf{x}_{\text{init}}$ and the goal state $\mathbf{x}_{\text{goal}}$ to some two states, respectively $\tilde{\mathbf{x}}_{\text{init}}$ and $\tilde{\mathbf{x}}_{\text{goal}}$, in the roadmap, with feasible paths. If it fails

to connect them to the roadmap, the query fails. Otherwise, a feasible path from $\mathbf{x}_{\text{init}}$ to $\mathbf{x}_{\text{goal}}$ is found. The feasible path is illustrated in figure 4.4(b).



(a) Construction phase of PRM (b) Query phase of PRM

Figure 4.4: Example of the PRM roadmap

In PRM, multiple queries can be answered at the same time and the learning and the query phases do not have to be executed sequentially. If there are some difficulties during the query phase, the PRM can switch to the learning phase to adapt the size of the roadmap instead. Thus, the learning flavor of the PRM increases. This makes the PRM suitable for trajectory planning for the multi-agent because it can look for solutions for every agent at the same time. However, the main shortcoming of this method is its poor performance on problems requiring paths that pass through narrow passages in the free space. The Medial Axis Probabilistic Roadmap Planner or MAPRM [WAS99] is developed to overcome this weakness. This method combines the approach of PRM and the Medial axis together by generating random networks whose states lie on the medial axis of the free state space.

4.3.2 Rapidly-exploring Random Trees (RRT)

RRT [LK99] is another famous sampling-based planning algorithm. While the PRM is a two-query path planning methods, the RRT is a single-query planning methods, *i.e.* building a search graph and finding a solution at the same time. The RRT is an incremental sampling-based planning algorithm that rapidly searches high-dimensional space with algebraic and differential constraints. Its principle is to bias the exploration toward unexplored areas by sampling states in the exploration space, *i.e.* breaking its large Voronoi diagrams [Vor07], and then extending the exploration tree toward them.

Definition 1 *Voronoi diagrams*
The Voronoi diagram partitions the state space into regions based on the samples. Each samples $\mathbf{x}$ *has an associated Voronoi region* $Vor(\mathbf{x})$. *For any state* $\mathbf{x}_i \in Vor(\mathbf{x})$, $\mathbf{x}$ *is the closest sample to* $\mathbf{x}_i$ *using euclidean distance (see figure 4.5).*

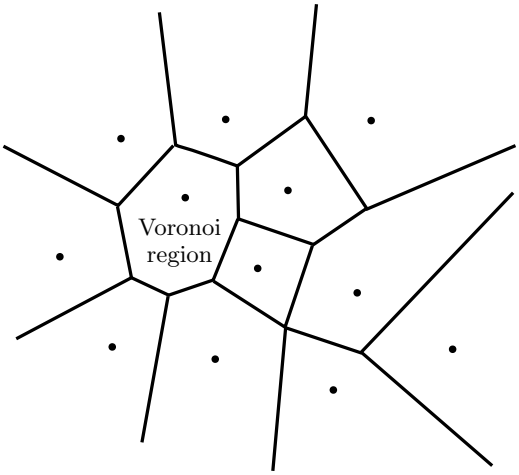


Figure 4.5: Example of Voronoi diagram

The algorithm proceeds by growing a single tree G from the initial state $\mathbf{x}_{\text{init}}$ until one of its branches reaches the destination $\mathbf{x}_{\text{goal}}$. For each iteration, the RRT algorithm generates a random state $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$. Then, it tries to expand the closest state in the tree $\mathbf{x}_{\text{near}} \in G$ in the tree to $\mathbf{x}_{\text{rand}}$ according to a metric d by applying a control input $\mathbf{u}$ to the system. A new state $\mathbf{x}_{\text{new}}$ is obtained after an integration step Δt. A newly obtained state and its connection edge are tested if they are collision-free. If they are collision-free, they are added to the tree. This is called the `rrt_extend` operation illustrated in figure 4.6. This procedure repeats until K iterations or a solution is found. The exploration tree and its Voronoi regions are illustrated in figure 4.7. These figures are taken from Lavalle's RRT webpage (http://msl.cs.uiuc.edu/rrt/gallery_2drrt.html).

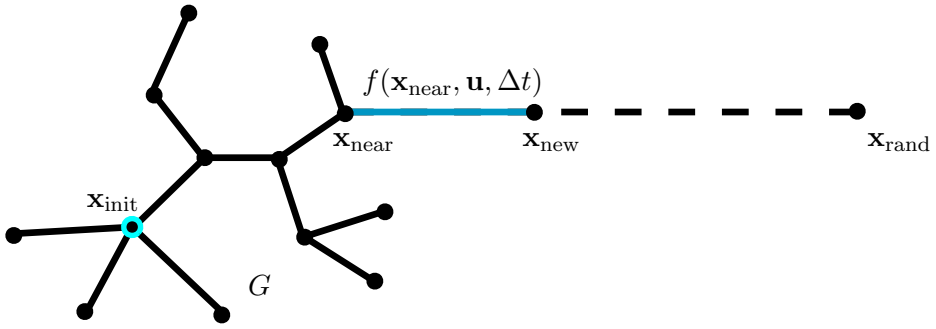


Figure 4.6: The `rrt_extend` operation

This makes RRT more suitable than other methods for real-time implementation. The generation of random state serves only to expand the search tree in the state space. The RRT algorithm is suitable for many practical planning problems due to its properties:

- The expansion of the RRT is principally biased toward large voronoi areas, *i.e.* the

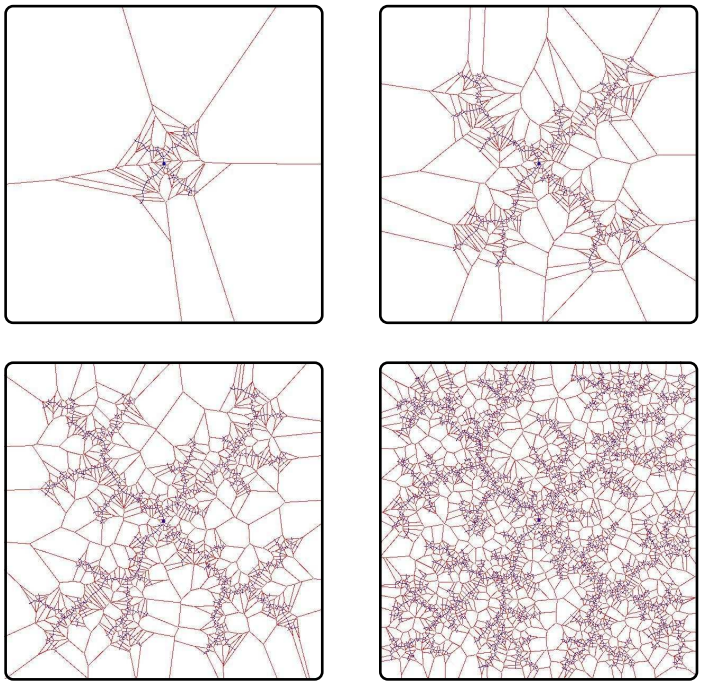


Figure 4.7: The RRT is biased by large Voronoi regions to rapidly explore, before uniformly convering the space

unexplored areas of the exploration space. This makes the RRT explores the entire space rapidly.

- The RRT is probabilistically complete [KL00]. It means that the probability of finding a solution approaches one as time approaches infinity. However, no solution can be found if no solution exists.
- There is no need for any local planners because the global planner is capable of finding a feasible path between two states by itself.
- The RRT always remains connected with the fewest edges. Due to this property, the RRT might be faster than a basic PRM because the PRM often has many extra edges generated trying to form a connected roadmap.
- The RRT uses a single nearest neighbor query while the PRM uses a more expensive k-nearest queries search.
- The RRT is a collision-free path planner algorithm.

According to these properties, it seems that the RRT algorithm may have better performance than the PRM algorithm. However, it is difficult to prove with the experimental comparison [LaV98].

The RRT has been a point of interest in robotics for the past 10 years. There are a lot of extensions of RRT. In this section, some interesting extensions are explained.

4.3.2.1 RRT-goalbias

This extension [LK01] is developed to ameliorate the performance of the RRT algorithm by directing the branches of the search tree to the desired destination. Instead of uniformly generating random states $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$, $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$ is generated with a probability p and $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$ is generated with a probability $(1 - p)$. This makes the algorithm converge to $\mathbf{x}_{\text{goal}}$ much faster than the classic RRT. However, if too much bias p is introduced, the search tree tends to expand its branches to $\mathbf{x}_{\text{goal}}$ even if the cost is too expensive. This will lead to the local minima problem, like potential field planner.

4.3.2.2 RRT-goalzoom

The improvement of this extension [LK01] is based on a biased coin toss, that chooses a random state $\mathbf{x}_{\text{rand}}$ from either a region around $\mathbf{x}_{\text{goal}}$ or $\mathbb{X}_{\text{free}}$. The size of the region around the goal is defined by the closest RRT-node to $\mathbf{x}_{\text{goal}}$ at any iterations (see figure 4.8). The consequence of this method is that the focus of the random states gradually increases around the destination as the RRT draws closer. This works very well in practice. However, its performance can also be degraded due to local minima.

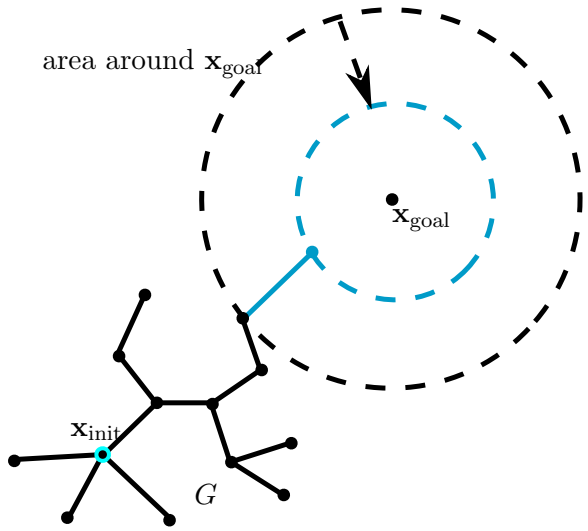


Figure 4.8: Illustration of area around the goal for RRT-goalzoom algorithm

4.3.2.3 RRT-connect

This extension [LK01] is designed, after considering the size of iterations used to build the search tree, specifically for path planning problems that involve no differential constraints. In this case, the need for incremental motions is less important. The method is based on two ideas: the *connect* heuristic that attempts to move over a longer distance if an obtained state is far from colliding, and the growth of RRTs from both $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$, *i.e.* Dual-RRT [LK01] in the next section.

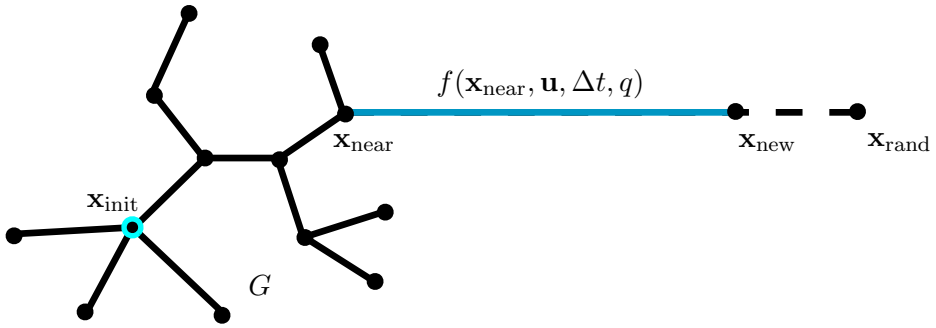


Figure 4.9: The `rrt_connect` operation

The *connect* heuristic can be considered as an alternative to the *extend* function. Instead of attempting to extend the RRT by a single integration step Δt, the *connect* heuristic keeps repeating that step q times or until an obstacle is reached (see figure 4.9). It is very similar to the artificial potential function in a randomized potential field approach. In both approaches, the search tree converges rapidly to a solution. However, with this extension, the heuristic is combined well with the properties of the RRT. The algorithm seems to avoid the well-known local minima problem and can find a solution rapidly. In one sense, with the *connect* heuristic, the center of attraction continues to move around as the RRT grows, while the center of attraction remains fixed at the destination in the APF.

After experiments on a variety of problems [LK01], it is concluded that the primitive *connect* yields the best performance for holonomic planning problems while the primitive *extend* seems to be the best for nonholonomic problems. One reason for this difference is that *connect* places more faith in the metric, and it becomes more challenging to design good metric for nonholonomic problems.

4.3.2.4 Dual-RRT

The Dual-RRT [LK01] uses a bidirectional scheme [Poh69], *i.e.* two search trees, to accelerate the exploration. One tree grows from the initial state and another grows from the goal state. Both trees are connectable in the realistic way, verified by the *connect* heuristic,

if their states are very close or the same state. This method is not suitable for problems whose decided destination is not a single state $\mathbf{x}_{\text{goal}}$ but a set of states $\mathbb{X}_{\text{goal}}$.

4.3.2.5 RRT* (optimal RRT)

Optimal RRT or RRT* [KF10, KF11] is a RRT algorithm with the asymptotic optimality property, *i.e.* almost-sure convergence to an optimal solution, along with probabilistic completeness guarantees. The RRT* is like the RRT that it can find a feasible path quickly. Moreover, the quality of the path is improved toward the optimal solution when more time passes before the path execution is complete. Since most robotic systems usually spend more time to execute trajectories than to plan them, this property is very advantageous.

The RRT* algorithm is the same as the RRT algorithm with some additional functions. Once $\mathbf{x}_{\text{new}}$ is obtained, the algorithm search the tree G around $\mathbf{x}_{\text{new}}$ in near vertex zone for a better path, *i.e.* less cost-to-go to $\mathbf{x}_{\text{new}}$. If there is a better path, the algorithm disconnects a path form $\mathbf{x}_{\text{nearest}}$ and establish a new path from $\mathbf{x}_{\text{min}}$, equivalent to $\mathbf{x}_{\text{near}}$ from the RRT algorithm, to $\mathbf{x}_{\text{new}}$ as shown in figure 4.10. Then, the algorithm searches among the states

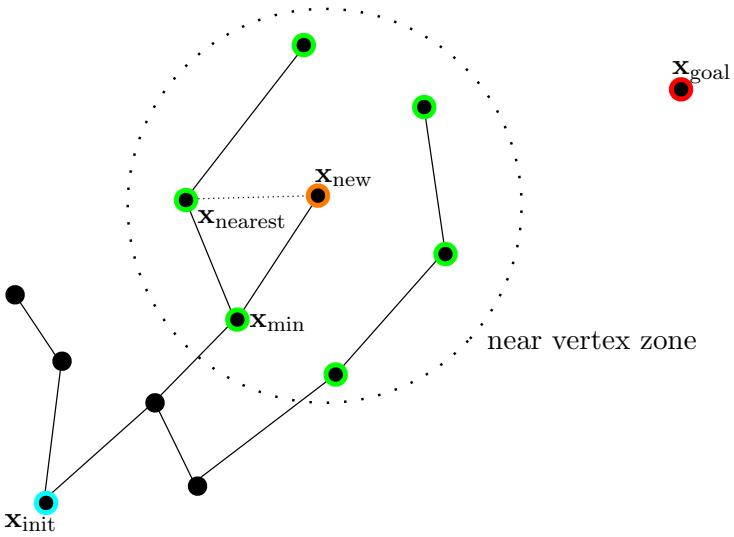


Figure 4.10: Process in finding the nearest path to $\mathbf{x}_{\text{new}}$

in the near vertex zone again. This time, it is looking for a better path from $\mathbf{x}_{\text{new}}$ to any states of the search tree in the near vertex zone. If there are better paths, the old paths are replaced by the new paths from $\mathbf{x}_{\text{new}}$. This process is called rewire process. The illustration of this process is shown in figure 4.11.

The RRT* algorithm is more time consuming than the RRT algorithm because of the two newly added functions. However, it can find a solution closer to the optimal solution than the RRT algorithm while using the remaining time after the first solution is found and before the path is executed by the vehicle.

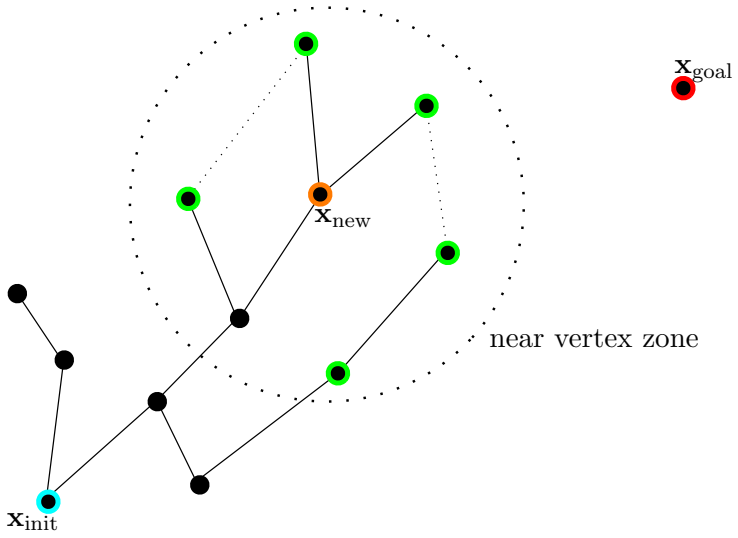


Figure 4.11: Rewire process of the tree around $\mathbf{x}_{\text{new}}$

4.4 Conclusion

All the planning algorithms presented here are very interesting with their own advantages. The search algorithms such as A*, D*, and Theta* are very simple and used widely. However, their performance depends mostly on the grid resolution. If the resolution is too low, *i.e.* large grid size, the feasible path can be missed by the search algorithm. If the resolution is too high, large computational effort is required. Moreover, for a complex system where the control capability varies in the state space. It is very hard to find a suitable resolution of the grid.

The APF is very interesting if the system can be interpreted into potential fields. For the complex system such as UAVs, it is quite difficult to find potential field functions that can represent the system.

While the PRM and RRT are suitable for high dimensional problems as their proprieties say. The principle of PRM and RRT is the same. It is to construct a roadmap or a tree in order to explore the space for a solution. The difference is that the PRM is a two-query algorithm while the RRT is a single-query. The PRM builds a roadmap in one query and tries to find a solution in another query. It is suitable for problems whose calculation time is not very important and for multiple robot path planning. While, the RRT builds its tree and tries to find a solution at the same time. It is more suitable for systems whose calculation time is important such as hypersonic aerial vehicles. Moreover, the RRT is more likely implementable in a real-time system. Thus, the RRT is the subject of interest in this thesis.

In the following chapter, a trajectory planning framework based on RRT algorithm for

an aerial vehicle will be studied. In chapter 5, the RRT will be tested along with some classical guidance laws and a specific metric. The performances are shown and analyzed.

Part III

Trajectory planning using a realistic model

5 RRT path planning for an aerial vehicle

The main contribution of this chapter is to introduce a new and novel method of trajectory planning for UAVS such as interceptor missiles using a probabilistic method especially the RRT algorithm.

In this chapter, the combination of path planning method and optimal control theory is proposed and is used to generate a feasible trajectory while considering complex system and environment. The RRT algorithm is used as a basic path planner together with the shortest Dubins' path as a metric function and a classical closed-loop control law as a node expansion method to find a feasible solution.

The objective of this chapter is to show the capability of finding a feasible trajectory for rendez-vous problems for UAVs. First, the RRT algorithm is explained in detail. Then, the application and modification of the RRT algorithm for an intercept mission of an interceptor missile are explained attentively. Next, the simulation results are presented and analyzed. Finally, the concluding remarks are made at the end of this chapter.

5.1 RRT algorithm

5.1.1 An overview

Rapidly-exploring Random Trees (RRT) [LK01] is an incremental method designed to efficiently explore non-convex high-dimensional space. The key idea is to visit the unexplored part of the state space by breaking its large Voronoi areas [Vor07]. The principle of the RRT as a path planner is described in Algorithm 1.

Algorithm 1 RRT path planner

Function : build_rrt(in : $K \in \mathbb{N}$, $\mathbf{x}_{\text{init}} \in \mathbb{X}_{\text{free}}$, $\mathbb{X}_{\text{goal}} \subset \mathbb{X}_{\text{free}}$, out : G)

1: $G \leftarrow \mathbf{x}_{\text{init}}$
2: $i = 0$
3: **repeat**
4: $\quad \mathbf{x}_{\text{rand}} \leftarrow \text{random\_state}(\mathbb{X}_{\text{free}})$
5: $\quad \mathbf{x}_{\text{new}} \leftarrow \text{rrt\_extend}(G, \mathbf{x}_{\text{rand}})$
6: **until** $i++ > K$ or ($\mathbf{x}_{\text{new}} \neq$ null and $\mathbf{x}_{\text{new}} \in \mathbb{X}_{\text{goal}}$)
7: **return** G

Function : rrt_extend(in : G, $\mathbf{x}_{\text{rand}}$, out : $\mathbf{x}_{\text{new}}$)

8: $\mathbf{x}_{\text{near}} \leftarrow \text{nearest\_neighbour}(G, \mathbf{x}_{\text{rand}})$
9: $(\mathbf{x}_{\text{new}}, \mathbf{u}) \leftarrow \text{steer}(\mathbf{x}_{\text{rand}}, \mathbf{x}_{\text{near}}, \Delta t)$
10: **if** collision_tests($\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}}$) **then**
11: $\quad G$.AddNode($\mathbf{x}_{\text{new}}$)
12: $\quad G$.AddEdge($\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}}$)
13: $\quad$ **return** $\mathbf{x}_{\text{new}}$
14: **else**
15: $\quad$ **return** null
16: **end if**

The search starts from the initial state $\mathbf{x}_{\text{init}}$ as the root of the tree G. Then, a state $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$ is randomly generated in `random_state` function. In `rrt_extend` function, the algorithm searches the tree G for the nearest vertex to $\mathbf{x}_{\text{rand}}$ according to a user-defined metric d in `nearest_neighbor` function. This state is called $\mathbf{x}_{\text{near}}$. In `steer` function, a control input $\mathbf{u}$ is selected according to a specified criterion in order to move the vehicle from $\mathbf{x}_{\text{near}}$ to $\mathbf{x}_{\text{rand}}$. The system model, environment model, and other constraints are considered and integrated for an integration step Δt. A newly obtained state is called $\mathbf{x}_{\text{new}}$. Finally, a collision test is performed in `collision_test` function: if $\mathbf{x}_{\text{new}}$ and the path between $\mathbf{x}_{\text{near}}$ and $\mathbf{x}_{\text{new}}$ lie in $\mathbb{X}_{\text{free}}$ then they are added to the tree.
These steps are repeated until K iterations are reached or when a solution is found, *i.e.* $\mathbf{x}_{\text{new}} \in \mathbb{X}_{\text{goal}}$.

5.1.2 Important components

According to LaValle and Kuffner, in order for the RRT algorithm to work effectively, the following elements must be well-defined.

5.1.2.1 Exploration space $\mathbb{X}$

The exploration space $\mathbb{X}$ determines the space where the RRT algorithm should explore for solutions. It can intuitively define how long the algorithm will take to find a solution.

Thus, it is very important to the RRT algorithm.

- If the exploration space $\mathbb{X}$ is too large, the algorithm will take time to find a solution.
- If the exploration space $\mathbb{X}$ is too small, the algorithm cannot guarantee to find a solution because the small exploration space $\mathbb{X}$ can possibly contain no solution.

The exploration space can be defined by considering the maximum capability and range of the vehicle. A exploration space $\mathbb{X}$ must be large enough in order to ensure that $\mathbb{X}$ contains all possible solutions. Thus, in a 2-dimensional plane, $\mathbb{X}$ is defined as

$$\begin{aligned} \mathbb{X} &= \mathbb{P} \times \mathbb{V}, \\ \mathbb{P} &= \{\boldsymbol{\xi} = (x, z)^\top \in \mathbb{R}^2\}, \\ \mathbb{V} &= \{(v, \gamma) \in \mathbb{R}^2\}. \end{aligned} \tag{5.1}$$

A state $\mathbf{x} \in \mathbb{X}$ is represented by a position $\boldsymbol{\xi}$, a vehicle speed v and a vehicle orientation γ, where x is a horizontal distance, and z is an altitude,

The exploration space contains two main subsets: collision-free space $\mathbb{X}_{\text{free}}$ and collision space $\mathbb{X}_{\text{obs}}$, *i.e.* $\mathbb{X}_{\text{free}} = \mathbb{X}/\mathbb{X}_{\text{obs}}$. These two subsets are defined according to each problem.

5.1.2.2 Random state generation

The random state generation is one of the key elements of the RRT algorithm. It determines where the algorithm tries to explore at each iteration. The principle of the RRT algorithm is to explore the space by expanding the exploration tree toward large Voronoi areas [Vor07]. In other words, the random states are normally generated in order to break large Voronoi areas into smaller ones.

Some criteria are applied in the `random_state` function to generate a random state $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$. The `random_state` function usually generate $\mathbf{x}_{\text{rand}}$ randomly and uniformly in the exploration space, *i.e.* the uniform distribution is normally used.

The rate of convergence to a solution of the algorithm can be increased by adding some heuristics such as RRT-*goalbias* or RRT-*goalzoom* [LK01].

5.1.2.3 Metric

Metric is used to determine a distance between two vehicle states in the RRT algorithm. The importance of the precision of the metric depends on each problem which we will see later in this thesis. Generally, the metric $d(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{rand}})$ determines a distance between two vehicle states, $\mathbf{x}_{\text{near}}$ and and $\mathbf{x}_{\text{rand}}$, in `nearest_neighbor` function. The definition of metric is defined in definition 2.

Definition 2 ***Metric***
A metric is a positive-definite function which determines a distance between elements of a set $\mathbb{X}$, i.e. $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$. *For all* $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3$ *in* $\mathbb{X}$, *the following conditions must be satisfied:*

1. $d(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2) \geqslant 0$ *(non-negativity);*
2. $d(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2) = 0$ *if and only if* $\boldsymbol{\xi}_1 = \boldsymbol{\xi}_2$ *(identity of indiscerniblility);*
3. $d(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2) = d(\boldsymbol{\xi}_2, \boldsymbol{\xi}_1)$ *(symmetry);*
4. $d(\boldsymbol{\xi}_1, \boldsymbol{\xi}_3) \leqslant d(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2) + d(\boldsymbol{\xi}_2, \boldsymbol{\xi}_3)$ *(subadditivity/triangle inequality).*

Usually, an **Euclidean metric** is used to determine a distance between two states. The euclidean metric gives the shortest distance of a line-of-sight between two states. In Cartesian coordinates, euclidean metric is defined as

$$d(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2) = d(\boldsymbol{\xi}_2, \boldsymbol{\xi}_1) = ||\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2||$$

.

- **Advantages:** the Euclidean metric is very simple and very easy-to-implement in a real system. This metric is usually used and probably is the best choice in most of holonomic systems and also some very simple nonholonomic systems since for simple systems, the euclidean distance can be considered the shortest path.
- **Drawbacks:** it does not take the orientation of the vehicle into account. Thus, it is not as interesting for nonholonomic vehicles as for holonomic vehicles.

Since the Euclidean metric is not always suitable to solve the nonholonomic problem, a user-defined metric d can be used instead.

Remark 1 *Determining an ideal metric for a problem can be quite difficult and challenging as solving the problem itself.*

5.1.2.4 Node expansion methods

Node expansion methods are used to move the vehicle from one state to another state. In order to expand the tree, a control input $\mathbf{u}$ used to expand the tree from $\mathbf{x}_{\text{near}}$ to $\mathbf{x}_{\text{rand}}$ is determined. The control input can be computed randomly or using a specific criterion such as some guidance control laws. The system model, environment model, and other constraints are integrated for an integration step Δt to obtain $\mathbf{x}_{\text{new}}$ in `steer` function.

5.1.2.5 Collision tests

Collision tests are methods used to verify a newly obtained state $\mathbf{x}_{\text{new}}$ and a trajectory between $\mathbf{x}_{\text{near}}$ and $\mathbf{x}_{\text{new}}$ if they are collision-free, *i.e.* $\mathbf{x}_{\text{new}} \in \mathbb{X}_{\text{free}}$. The collision-free states and trajectories are added to the tree G.

5.2 Application for an interceptor missile

A hypersonic aerial vehicle such as an interceptor missile is chosen to be a case study here. The reason is that the missile flies in a large range of altitude which results in an exponential decrease of maneuverability at high altitude compared to the one at low altitude. This is due to the exponential decrease of the air density with respect to altitude. Moreover, the surface of reference of the missile is much smaller than the other aerial vehicles. As a consequence, the air resistance caused by wind has less effect on the missile than any other aerial vehicles. Thus, it can be ignored.

5.2.1 System modeling

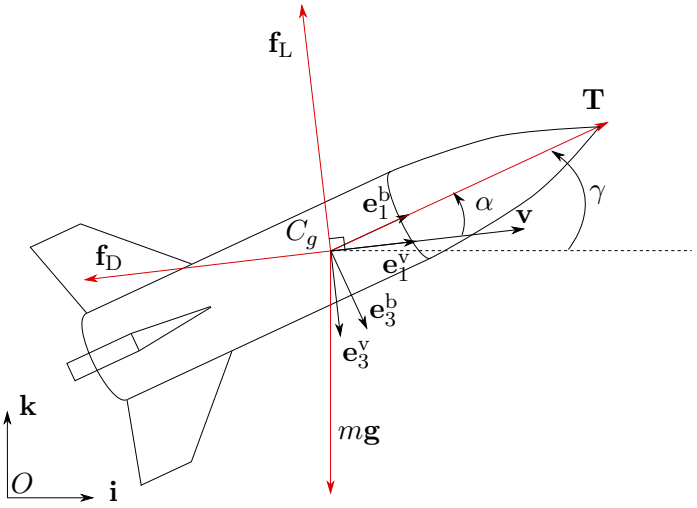


Figure 5.1: Definition of reference frames in vertical plane

The missile is modeled as a rigid body of mass m and inertia I maneuvering in a vertical 2-dimensional plane. A round Earth model is used. Due to small flight times (less than one minute), the Earth rotation has very few effects on the missile so it is neglected. Three frames (see figure 5.1) are introduced to describe the motion of the vehicle: an Earth-Centred Earth-Fixed (ECEF) reference frame $\mathcal{I}$ centered at point O and associated with the basis vectors $(\mathbf{i}, \mathbf{k})$; a body-fixed frame $\mathcal{B}$ attached to the vehicle at its center of mass C_g with the vector basis $(\mathbf{e}_1^{\text{b}}, \mathbf{e}_3^{\text{b}})$; and a velocity frame $\mathcal{V}$ attached to the vehicle at C_g with

the vector basis $(\mathbf{e}_1^{\mathrm{v}}, \mathbf{e}_3^{\mathrm{v}})$ where $\mathbf{e}_1^{\mathrm{v}} \stackrel{\text{def}}{=} \frac{\mathbf{v}}{\|\mathbf{v}\|}$ and $\mathbf{v}$ is the translational velocity of the vehicle in $\mathcal{I}$. Position and velocity defined in $\mathcal{I}$ are denoted $\boldsymbol{\xi} = (x, z)^\top$ and $\mathbf{v} = (\dot{x}, \dot{z})^\top$. Since the velocity of the missile is much greater than the velocity of the wind and the flight time is less than a minute, it is reasonable to assume that wind has no effect on the vehicle. Thus, the translational velocity $\mathbf{v}$ is assumed to coincide with the apparent velocity. The orientation of the missile is represented by the pitch angle γ from horizontal axis to $\mathbf{e}_1^{\mathrm{b}}$. The angular velocity is defined in $\mathcal{B}$ as $q \stackrel{\text{def}}{=} \dot{\gamma}$.

The model of interceptors with a boost phase and a midcourse phase is studied in this chapter. Thus, translational forces include lift $\mathbf{f}_{\mathrm{L}}$, drag $\mathbf{f}_{\mathrm{D}}$, thrust $\mathbf{T}$, and weight $m\mathbf{g}$ (see figure 5.1). Aerodynamic and perturbation torque are denoted τ_{aero} and τ_{pert}, respectively. Using these notations, the vehicle dynamics can be written as

$$\dot{\boldsymbol{\xi}} = \mathbf{v}, \tag{5.2}$$

$$m\dot{\mathbf{v}} = \mathbf{f}_{\mathrm{D}} + \mathbf{f}_{\mathrm{L}} + \mathbf{T} + m\mathbf{g}, \tag{5.3}$$

$$\dot{\gamma} = q, \tag{5.4}$$

$$I\dot{q} = \tau_{\mathrm{aero}} + \tau_{\mathrm{pert}}. \tag{5.5}$$

The aerodynamic forces are

$$\begin{aligned} \mathbf{f}_{\mathrm{D}} &= -\frac{1}{2}\rho v^2 S C_{\mathrm{D}} \mathbf{e}_1^{\mathrm{v}}, \\ \mathbf{f}_{\mathrm{L}} &= -\frac{1}{2}\rho v^2 S C_{\mathrm{L}} \mathbf{e}_3^{\mathrm{v}}, \end{aligned} \tag{5.6}$$

where ρ is the air density, S is the missile reference area, C_{D} is the drag coefficient, C_{L} is the lift coefficient, and $v \stackrel{\text{def}}{=} \|\mathbf{v}\|$. C_{L} and C_{D} both depend on the angle of attack α [Sio04].

A hierarchical controller is used to control the lateral acceleration $\mathbf{u} = u\mathbf{e}_3^{\mathrm{v}}$ perpendicular to $\mathbf{v}$: an inner loop stabilizes the rotational velocity q of the vehicle and an outer loop controls the angle of attack α [DSF00]. In the following, $\mathbf{u}_{\mathrm{c}}$ denotes the acceleration set point and α_{d} denotes the desired angle of attack related to $\mathbf{u}_{\mathrm{c}}$. The second order dynamic response of α is modeled as

$$\frac{\alpha}{\alpha_{\mathrm{d}}} = \frac{1}{\left(\frac{p}{\omega}\right)^2 + \frac{2\zeta}{\omega_0}p + 1}, \tag{5.7}$$

where ζ is a damping ratio and ω_0 is an angular frequency.

For the environment modeling, the US Standard Atmosphere 1976 (US-76) is used. In the lower earth atmosphere (altitude < 35 km), density of air, and atmospheric pressure decrease exponentially with altitude and approach zero at approximately 35 km. As we consider a missile with only aerodynamic flight controls, the maneuvering capabilities are linked to the density of air (cf. equation (5.6)) and approach zero at 35 km.

5.2.2 The Predicted Intercept Point (PIP)

In interceptor missile guidance, the mission is defined by the destination $\mathbf{x}_{\text{goal}}$ called a predicted intercept point or PIP denoted $\mathbf{x}_{\text{pip}}$. The predicted intercept point is defined by a specific algorithm after receiving information from a radar station where target trajectory is predicted and estimated. Then, the missile trajectory to a specific point on the target trajectory is computed. Thus, the trajectory generation to the PIP is studied.

5.2.3 Problem formulation

Let $\mathbf{x}(t) = (\boldsymbol{\xi}(t)^\top, v, \gamma)^\top \in \mathbb{X} = \mathbb{R}^4$ be the state of the system, $\mathbf{u}_{\text{c}} \in \mathbb{U}(\mathbf{x}) \subset \mathbb{R}^2$ be an admissible control input and consider the differential system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}_{\text{c}}), \tag{5.8}$$

where $\mathbf{f}(\cdot)$ is defined in section 5.2.1. Note that the state is hypothetically measurable for the trajectory planning method.

$\mathbb{X} = \mathbb{R}^4$ is the state space. It is divided into two subsets. Let $\mathbb{X}_{\text{free}}$ be the set of admissible states. $\mathbb{X}_{\text{obs}} = \mathbb{X} \setminus \mathbb{X}_{\text{free}}$ is the obstacle region *i.e.* the set of non-admissible states.

The initial state of the system is $\mathbf{x}_{\text{init}} \in \mathbb{X}_{\text{free}}$. The path planning algorithm is given a predicted intercept point $\mathbf{x}_{\text{pip}} = (\boldsymbol{\xi}_{\text{pip}}, \mathbf{v}_{\text{pip}})$. In order to achieve its mission, the interceptor has to reach a goal set $\mathbb{X}_{\text{goal}} \subset \mathbb{X}_{\text{free}}$.

The motion planning problem is to find a collision-free trajectory $\mathcal{X}(t) : [0, t_f] \to \mathbb{X}_{\text{free}}$ with $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}_{\text{c}})$, that starts at $\mathbf{x}_{\text{init}}$ and reaches the goal region, *i.e.* $\mathbf{x}(0) = \mathbf{x}_{\text{init}}$ and $\mathbf{x}(t_f) \in \mathbb{X}_{\text{goal}}$. The secondary objective is to obtain a trajectory that maximizes the final speed $v(t_f)$.

5.2.4 Dubins' paths

5.2.4.1 An overview

Dubins' paths [Dub57, BCL91] (cf. section 3.3.2.3) are calculated using the optimal control method while fixing the maneuverability of the car to the maximum, *i.e.* minimal turning radius. For nonholonomic systems, the minimal length path is studied by Dubins [Dub57]. Dubins succeeds in finding the shortest path between two vehicle states considering their departure and arrival orientations. The vehicle used in his study is known as Dubins' car.

The Dubins' car is modeled as follows:

$$\begin{cases} \dot{x} = \cos\gamma, \\ \dot{y} = \sin\gamma, \\ \dot{\gamma} = c_{\max} u, \end{cases} \tag{5.9}$$

where $(x, y) \in \mathbb{R}^2$ is the vehicle position, $\gamma \in \mathbb{R}$ is the vehicle orientation, $c_{\max}$ is the maximum path curvature, and $u \in [-1, 1]$ is the control input.

As state in section 3.3.2.3, according to Dubins, the optimal path must be one of these two types: CSC (arc-segment-arc) and CCC (arc-arc-arc) or their degeneration forms (C, S, CS, SC). The CSC and CCC paths are illustrated in figure 3.5 in chapter 3.

5.2.4.2 Dubins' paths for aerial vehicles

In order for our system to be considered as a Dubins' car, the gravity g is neglected. Thus the system in two-dimensional plane is simplified and can be expressed as

$$\begin{cases} \dot{x} = v\cos\gamma, \\ \dot{z} = v\sin\gamma, \\ \dot{\gamma} = vc(z)u, \end{cases} \tag{5.10}$$

where v is the vehicle speed and $c(z) \in \mathbb{R}_+$ is the maximum path curvature that the vehicle can perform at the altitude z. Here, $c(z)$ is considered constant at the initial altitude z_0. Thus, the path curvature at z_0 is written as $c(z_0) = \frac{1}{2m}\rho S C_L$ where $\rho = \rho(z_0)$.

The optimization problem is to minimize the total length of the trajectory. It means minimizing the function

$$s_f = \int_0^{t_f} v dt,$$

where t_f is the final time assumed to be free (free interval optimal control problem) and s_f is the total length of the path.

Since the speed v varies along the flight, the system is very difficult to solve. Changing variable from time t to curvilinear abscissa $s(t) = \int_0^{t_f} v(u)du$ is necessary to solve this problem since the dynamics of the speed v is not specified. Thus, the model of the vehicle can be represented by:

$$\begin{cases} x' = \cos\gamma, \\ z' = \sin\gamma, \\ \gamma' = c(z)u, \quad |u| \leqslant 1. \end{cases} \tag{5.11}$$

Therefore, the problem consists in minimizing s_f, the path length which is equivalent to the system used by Dubins and Boissonnat's team. Notice that in case of constant velocity v, minimizing the path length s_f is equivalent to minimizing the final time t_f.

This simplified system (5.11) can represent our problem. However, the structural limitation and the control saturation due to the diminution of air density are not considered in the calculation. Thus, the Dubins' paths cannot represent the actual trajectory of the missile directly. However, they can still be used to estimate the distance between two missile states. The detailed calculation of how to find Dubins' paths and their length are shown in appendix B.

5.2.5 RRT configurations

5.2.5.1 Exploration space

1. Obstacle space $\mathbb{X}_{\text{obs}}$: this space contains obstacles, *i.e.* a set of inadmissible states, such as mountains, radar detection zones, no-fly zones, etc. It can also contain conditions of the vehicle state that are considered impossible to complete the mission; for example, low vehicle speed. In this chapter, $\mathbb{X}_{\text{obs}}$ is defined as

$$\begin{aligned} \mathbb{X}_{\text{obs}} &= \mathbb{P}_{\text{obs}} \times \mathbb{V}_{\text{obs}}, \\ \mathbb{P}_{\text{obs}} &= \{\boldsymbol{\xi} = (x, z)^\top \in \mathbb{R}^2 : z < 0\}, \\ \mathbb{V}_{\text{obs}} &= \{(v, \gamma) \in \mathbb{R}^2 : v(t > t_{\text{boost}}) < v_{\min}\}, \end{aligned} \tag{5.12}$$

   where t_{boost} is the duration of the boost phase starting from launch, and $v_{\min}$ is the acceptable minimal speed of the missile defined by a lethal system to be able to destroy the target.

2. Collision-free exploration space $\mathbb{X}_{\text{free}} = \mathbb{X}/\mathbb{X}_{\text{obs}}$: this is the space where the RRT algorithm will try to explore to find a solution.

3. Destination set $\mathbb{X}_{\text{goal}} \in \mathbb{X}_{\text{free}}$: this space, illustrated in figure 5.2, is defined by considering the capability of lethal system of the missile, mostly the capability to detect, hit, and destroy the target. Thus,

$$\begin{aligned} \mathbb{X}_{\text{goal}} &= \mathbb{P}_{\text{goal}} \times \mathbb{V}_{\text{goal}}, \\ \mathbb{P}_{\text{goal}} &= \{\boldsymbol{\xi} \in \mathbb{P}_{\text{free}} : \|\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{pip}}\| < R_{\min}\}, \\ \mathbb{V}_{\text{goal}} &= \{(v, \gamma) \in \mathbb{V}_{\text{free}} : \gamma \in \mathbb{C}(\boldsymbol{\xi}, \gamma_{\text{pip}}, \phi_f)\}, \end{aligned} \tag{5.13}$$

   where $R_{\min}$ is a radius of a sphere centered at $\boldsymbol{\xi}_{\text{pip}}$ and ϕ_f is an angle related to the maximum capability of the terminal guidance system. Values of $R_{\min}$ and ϕ_f are defined by the lethal system of the missile.

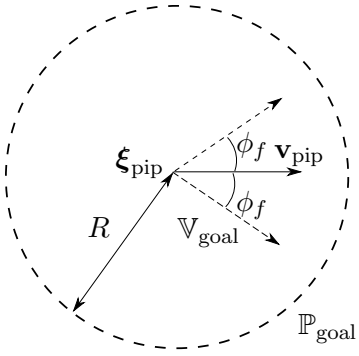


Figure 5.2: The goal set $\mathbb{X}_{\text{goal}}$ for path planning using RRT

5.2.5.2 Random state generation

The random state $\mathbf{x}_{\text{rand}}$ is generated such that $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$. A uniform distribution is used. Moreover, a bias toward the goal can be introduced to reduce the number of generated states in order to reach $\mathbb{X}_{\text{goal}}$. The RRT-*goalbias* [LK01] is used. Instead of randomly generating random states, `random_state` function returns $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$ with a probability p and returns randomly generated state using uniform distribution with a probability $(1-p)$. According to LaValle and Kuffner the bias p must not be too large in order for the RRT algorithm to explore the state space and not be trapped in a local minimum.

5.2.5.3 Metric

A metric, called Dubins' metric in this thesis, is developed based on Dubins' paths described in section 5.2.4. The shortest Dubins' path is chosen as the distance function. There are two cases to consider in order to use this metric to solve the problem:

1. $\mathbf{x}_{\text{rand}} \notin \mathbb{X}_{\text{goal}}$: the shortest CSC path is used to calculate the nearest neighbor.

2. $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$: according to $\mathbb{X}_{\text{goal}}$ stated in section 5.2.5.1, it means that the arrival condition is no longer a state (position and orientation). It is more interesting to consider the shortest path between each $\mathbf{x} \in G$ and a set of goal states $\mathbb{X}_{\text{goal}}$ than considering a single state $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$. Indeed, there can exist the shortest path to another element of $\mathbb{X}_{\text{goal}}$ than the shortest path to $\mathbf{x}_{\text{rand}}$. To this manner, the degenerated form, CS path, of CSC path needs to be considered first. Indeed, if the shortest path to the set $\mathbb{X}_{\text{goal}}$ is a CS path, then the arrival orientation is within the arrival cone, *i.e.* $\gamma_f \leqslant \phi_{\text{pip}} \pm \phi_f$, the shortest path is the CS path (blue curve in figure 5.3). If no CS path arrives in $\mathbb{X}_{\text{goal}}$, the shortest path is necessarily a CSC path whose arrival orientation is one of the extremities of the arrival cone (black solid curve in figure 5.4), *i.e.* $\gamma_f \in \{\phi_{\text{pip}} - \phi_f, \phi_{\text{pip}} + \phi_f\}$.

Thus, for each $\mathbf{x} \in \mathbb{G}$, the approach first consists in finding the shortest CS path to the desired final position, *i.e.* $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$. If the final orientation is in the arrival cone, it is the expected solution. If not, the solution is the shortest CSC path to one of the extremities of $\mathbb{X}_{\text{goal}}$.

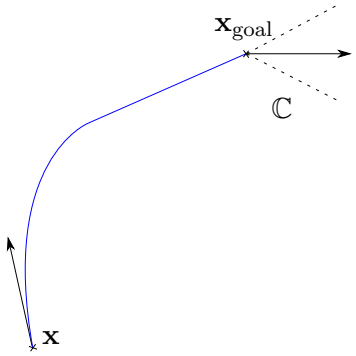


Figure 5.3: CS type path arrives in the arrival cone

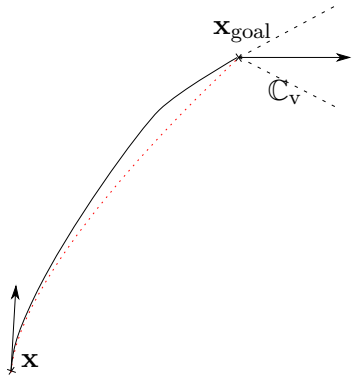


Figure 5.4: CS type path does not arrive in the arrival cone

- **Advantages:** the Dubins' metric considers the orientation of both states. It is an optimal length solution in two-dimensional plane in case of constant curvature.
- **Drawbacks:** it takes more computational time than Euclidean metric. There are some discontinuities of curvature at the intersection between different type paths. This leads to the discontinuities of control input.

Remark 2 *There are some drawbacks to the Dubins' metric such as the computational time and discontinuities along the path. However, these discontinuities are not important in our case since Dubins' metric is only used to estimate the shortest distance between two states while considering their orientation. Moreover, the objective of this section is to show*

that the RRT algorithm can find a solution while the classical guidance laws cannot. Thus, the computational time is not considered important at this moment.

Therefore, the Dubins' metric is used in `nearest_neighbor` function instead of Euclidean metric.

5.2.5.4 Node expansion methods

To rapidly explore the exploration space $\mathbb{X}_{\text{free}}$, the control input $\mathbf{u}_{\text{com}}$ is chosen in order to create a new state $\mathbf{x}_{\text{new}}$ as close as possible to $\mathbf{x}_{\text{rand}}$. Two guidance laws are used and tested in `control_input` function.

1. Proportional Navigation (PN) guidance law [Sio04] :

$$\mathbf{u}_{\text{com}} = \left(\frac{N}{t_{\text{go}}^2} (\boldsymbol{\xi}_{\text{rand}} - \boldsymbol{\xi}_{\text{near}} - \mathbf{v}_{\text{near}} t_{\text{go}}) \cdot \mathbf{e}_3^{\text{v}} \right) \mathbf{e}_3^{\text{v}} \tag{5.14}$$

   where N is a constant gain and t_{go} is the estimated time-to-go, *i.e.* $t_{\text{go}} = t_f - t$.

2. Kappa guidance law [Lin91] :

$$\mathbf{u}_{\text{com}} = ((\mathbf{a}_1 + \mathbf{a}_2) \cdot \mathbf{e}_3^{\text{v}}) \mathbf{e}_3^{\text{v}}, \tag{5.15}$$

   with

$$\begin{aligned} \mathbf{a}_1 &= \frac{K_1}{t_{\text{go}}} (\mathbf{v}_{\text{rand}} - \mathbf{v}_{\text{near}}), \\ \mathbf{a}_2 &= \frac{K_2}{t_{\text{go}}^2} (\boldsymbol{\xi}_{\text{rand}} - \boldsymbol{\xi}_{\text{near}} - \mathbf{v}_{\text{near}} t_{\text{go}}), \end{aligned} \tag{5.16}$$

   choosing $||\mathbf{v}_{\text{rand}}|| = ||\mathbf{v}_{\text{near}}||$, K_1 and K_2 are the gains.

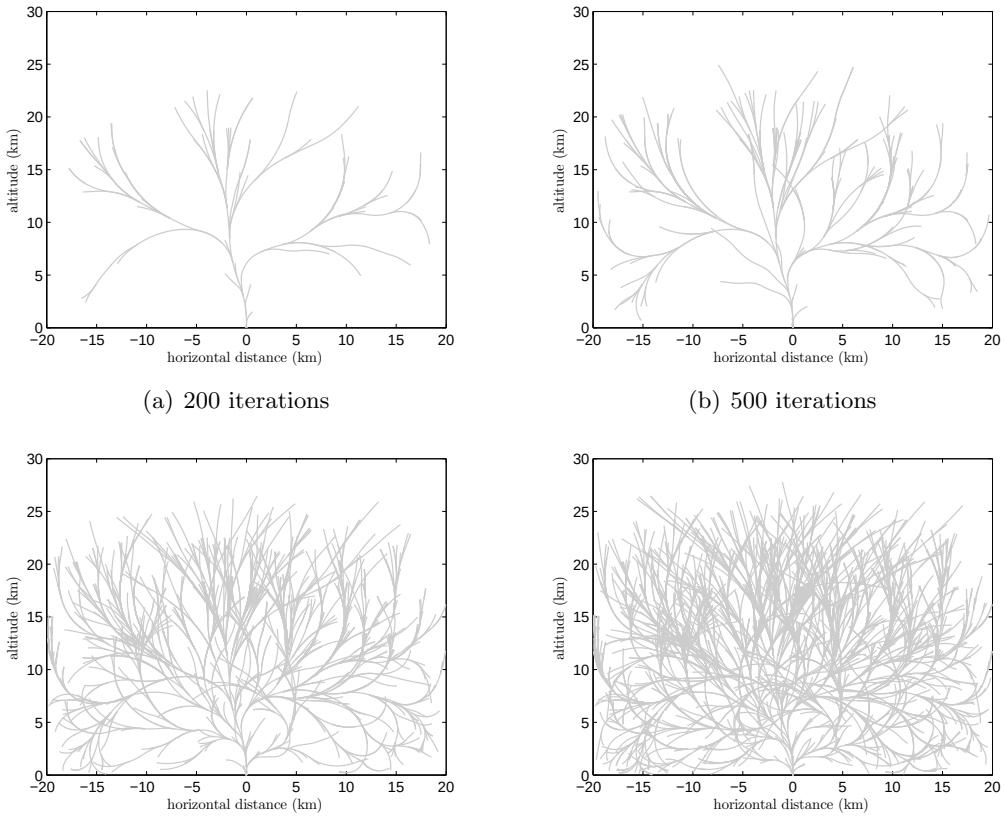
   The second term $\mathbf{a}_2$ is the proportional navigation term which nullifies the heading error to the PIP while the first term $\mathbf{a}_1$ is the shaping term which rotates the orientation of the missile so that γ converges to γ_f. The optimal gains K_1 and K_2 are approximated, shown in [Lin91] and are adapted to our system in appendix A.

The estimation of t_{go} is a difficult problem as it theoretically necessitates to estimate the entire flight time t_f to the target. There exists many techniques to provide an approximate of t_{go} [Lin91].

To satisfy the missile constraints, the desired control input $\mathbf{u}_{\text{c}}$ is obtained after saturating $\mathbf{u}_{\text{com}}$ so that $\mathbf{u}_{\text{c}} \in \mathbb{U}(t, \mathbf{x})$. Then, the second order dynamic response of α is considered to obtain the real control input applied to the system $\mathbf{u} \in \mathbb{U}(t, \mathbf{x})$.

5.3 Simulation results

The RRT algorithms using the previously mentioned configurations are simulated in certain scenarios. The one using NP as the control input selection will be referred as RRT_{PN} published in [PHPB13] and the one using kappa guidance will be referred as $\text{RRT}_{\text{kappa}}$ published in [PPHB13]. In this section, the results obtained by the RRT algorithms are compared with the ones obtained by the classical missile guidance law, the kappa guidance with optimal gains. The results are visualized by MATLAB simulations. Figure 5.5 illustrates an example of the tree expansion of the RRT algorithm using this framework. It shows that the exploration tree covers the entire state space $\mathbb{X}$.



(a) 200 iterations

(b) 500 iterations

(c) 2000 iterations

(d) 4000 iterations

Figure 5.5: RRT expansion with proportional navigation guidance law

Both algorithms are simulated and analyzed with two scenarios. For both scenarios, the integration step $\Delta t = 2\text{s}$, the initial position $\boldsymbol{\xi}_{\text{init}} = (0\text{km}, 0\text{km})^\top$, the missile is launched vertically, $\gamma_{\text{pip}} = 0$ ($\mathbf{v}_{\text{pip}}$ is parallel to the ground), and the second order dynamic response of α, $\zeta = 0.8$ and $\omega_0 = 16\text{rad/s}$. The difference lies in the PIP:

1. scenario 1: $\boldsymbol{\xi}_{\text{pip}} = (15\text{km}, 15\text{km})^\top$
2. scenario 2: $\boldsymbol{\xi}_{\text{pip}} = (10\text{km}, 25\text{km})^\top$

In the following figures, the dashed curve is the trajectory obtained using only the kappa guidance, the tree G is represented in grey, $\mathbb{X}_{\text{goal}}$ is represented by a red circle with two dashed line segments. The boost phase of the generated trajectory between $\mathbf{x}_{\text{init}}$ and $\mathbb{X}_{\text{goal}}$ is in green, the second phase in pink.

The simulation results for scenario 1 are shown in figures 5.6 and 5.7. The maximum control input tolerated by the missile is denoted $\mathbf{u}(\alpha_{\max})$. The control input given by the kappa guidance and the RRT-based algorithm are denoted $\mathbf{u}_{\text{kappa}}$ and $\mathbf{u}_{\text{RRT}}$, respectively. Figures 5.6(a) and 5.7(a) illustrate the exploration tree and the obtained trajectories. The number of generated nodes to find the result of scenario 1 shown in figures 5.6(a) and 5.7(a) are 1380 and 473 for the RRT_{PN} and $\text{RRT}_{\text{kappa}}$, respectively (cf. table 5.1). The final speed for the kappa guidance is 1635m/s. Whereas, the final speed of the trajectory found by RRT_{PN} and the one found by $\text{RRT}_{\text{kappa}}$ are 1360m/s and 1290m/s, respectively (cf. table 5.1). It is reasonable that the final speed of the solutions found by the RRT-based algorithm is lower than the one found by the kappa guidance. This is due to the randomness of the algorithm.
Figures 5.6(b) and 5.7(b) illustrate the normalized control inputs $||\mathbf{u}_{\text{kappa}}||/||\mathbf{u}(\alpha_{\max})||$ and $||\mathbf{u}_{\text{RRT}}||/||\mathbf{u}(\alpha_{\max})||$ returned by system using kappa guidance and RRT-based algorithm, respectively. Although the control input of the kappa guidance method is saturated at the end of the trajectory ($t/t_f > 0.9$), *i.e.* the desired control input is larger than the maximum control input that the missile can tolerate; and the control input of the RRT-based algorithm is saturated at some points during the trajectory, both trajectories reach $\mathbb{X}_{\text{goal}}$ successfully. Note that the envelope of the maximum control input is calculated using the solution trajectory of the RRT-based algorithm as the reference. Thus, it may not always be compatible with the trajectory obtained by the kappa guidance as shown later in the results of the scenario 2.

Scenario 1 illustrates a case where a solution can be found using a classical guidance law. It is a representative of general cases for an interceptor missile. In this case, the RRT-based algorithm is also able to provide a trajectory with similar performance even though the final speed is smaller than the one given by the classical guidance law.

The simulation results for the scenario 2 are presented in figures 5.8 and 5.9. Figures 5.8(a) and 5.9(a) illustrate the exploration trees and the obtained trajectories simulated in scenario 2. This case differs from scenario 1 since the target is higher in altitude and closer in terms of horizontal distance. It is a representative of difficult cases for an interceptor missile since the difficulty for aerodynamically controlled missiles lies in the low maneuver-

ability at high altitude due to the low density of air. Therefore, it is hard to satisfy the constraint $\angle(\mathbf{v}(t_f), \mathbf{v}_{\text{pip}}) = (\gamma_f - \gamma_{\text{pip}}) < \phi_f$ where $\mathbf{v}_{\text{pip}}$ is parallel to the ground ($\gamma_{\text{pip}} = 0$)). Since the trajectory found by kappa guidance (dashed curve) has $\angle(\mathbf{v}(t_f), \mathbf{v}_{\text{pip}}) = (\gamma_f - \gamma_{\text{pip}}) > \pi/8$ rad, it cannot satisfy this arrival constraint. It does not anticipate the future lack of maneuverability and sends a sequence of low control inputs until $t/t_f = 0.6$ (see figures 5.8(b) and 5.9(b)). Thus, at the end of the trajectory ($t/t_f > 0.9$), the kappa guidance tries to respect the aspect angle by sending large control input sequence to the controller. As the maneuvering capabilities are low, the missile cannot perform such demanded control input. Thus, the control input is saturated. As a consequence, the missile fails to reach $\mathbb{X}_{\text{goal}}$.

On the contrary, since the RRT-based algorithm anticipates the loss of maneuverability at high altitude near $\mathbb{X}_{\text{goal}}$, the generated trajectory performs a back-turn. Indeed, it moves away from the line-of-sight at the beginning in order to reduce the curvature of the trajectory when approaching $\mathbb{X}_{\text{goal}}$. Thus, the needed control input at the end of the trajectory remains lower than the maneuvering capabilities at these altitudes (see figures 5.8(b) and 5.9(b)). Since this problem is harder to solve than the previous one, the number of generated nodes increases and reaches 18149 for RRT_{PN} and 11261 for $\text{RRT}_{\text{kappa}}$ (cf. Table 5.1). Furthermore, $\|\mathbf{v}(t_f)\| = v_f = 1129\text{m/s} > v_{\min}$ for RRT_{PN} and $\|\mathbf{v}(t_f)\| = v_f = 953\text{m/s} > v_{\min}$ for $\text{RRT}_{\text{kappa}}$ are respected (cf. table 5.1).

Method	Scenario 1		Scenario 2	
	Number of iterations	Final speed (m/s)	Number of iterations	Final speed (m/s)
RRT_{PN}	1380	1360	18149	1129
$\text{RRT}_{\text{kappa}}$	473	1290	11261	953
Kappa	-	1635	-	×

Table 5.1: Results of simulations using RRT algorithm

This scenario illustrates the effectiveness of the RRT-based guidance law compared to classical midcourse guidance laws, based on its capability to anticipate future flight conditions.

Note that it is normal that the control input of the RRT algorithm is continuous but not smooth. This is due to the randomly generated state $\mathbf{x}_{\text{rand}}$. The desired control input is calculated each time with the different $\mathbf{x}_{\text{rand}}$. In figures 5.6(b), 5.7(b), 5.8(b), and 5.9(b), we can see the points where the control input increases or decreases dramatically. That is where $\mathbf{x}_{\text{rand}}$ is changed.

5.4 Conclusions

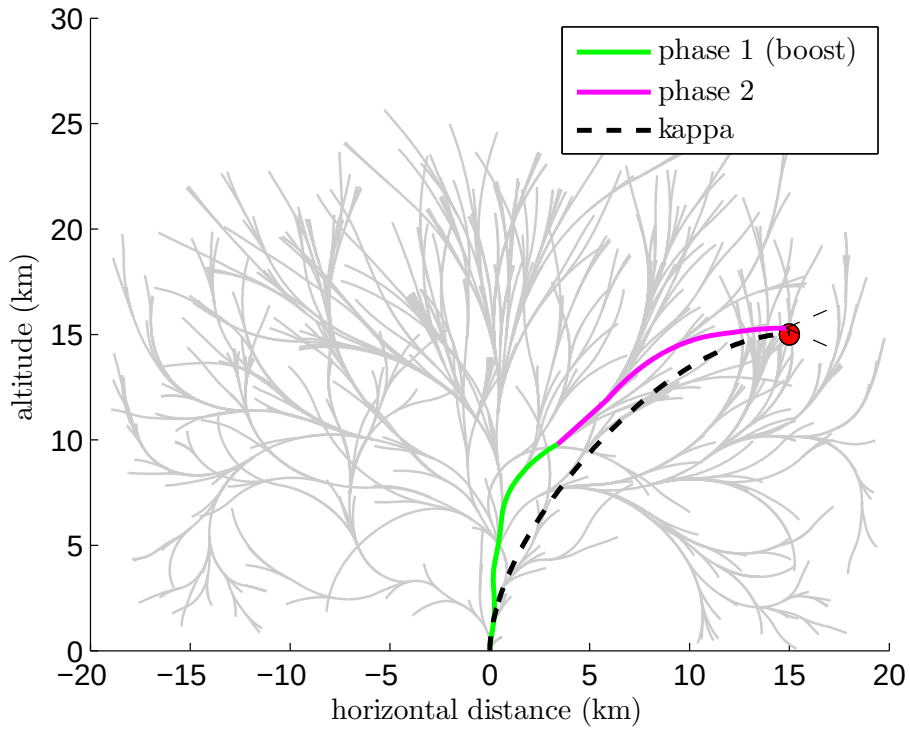
This trajectory planning framework is a combination of a sampling-based RRT path planner, Dubins' paths, and classical guidance laws. The RRT path planner is used as a basis. The shortest Dubins' path is used as a metric to determine a distance between two vehicle states. Classical guidance laws such as PN and kappa guidance law determine sequences of control inputs to move from one state to another state. The critical midcourse guidance problems that cannot be easily solved using classical guidance laws can be solved by this method as it anticipates the future flight conditions. The difference between $\mathrm{RRT}_{\mathrm{PN}}$ and $\mathrm{RRT}_{\mathrm{kappa}}$ is that $\mathrm{RRT}_{\mathrm{kappa}}$ uses the guidance law that considers both orientations of the starting and ending states unlike the $\mathrm{RRT}_{\mathrm{PN}}$ which considers only the orientation of the starting state. This makes the $\mathrm{RRT}_{\mathrm{kappa}}$ can find a solution with less number of nodes than the $\mathrm{RRT}_{\mathrm{PN}}$ while the final speed of the solution found by $\mathrm{RRT}_{\mathrm{kappa}}$ is a bit less than the one found by $\mathrm{RRT}_{\mathrm{PN}}$ as shown in Table 5.1.

Even though this method can find a solution for problems which are difficult to solve using the classical guidance laws. There are still some problems:

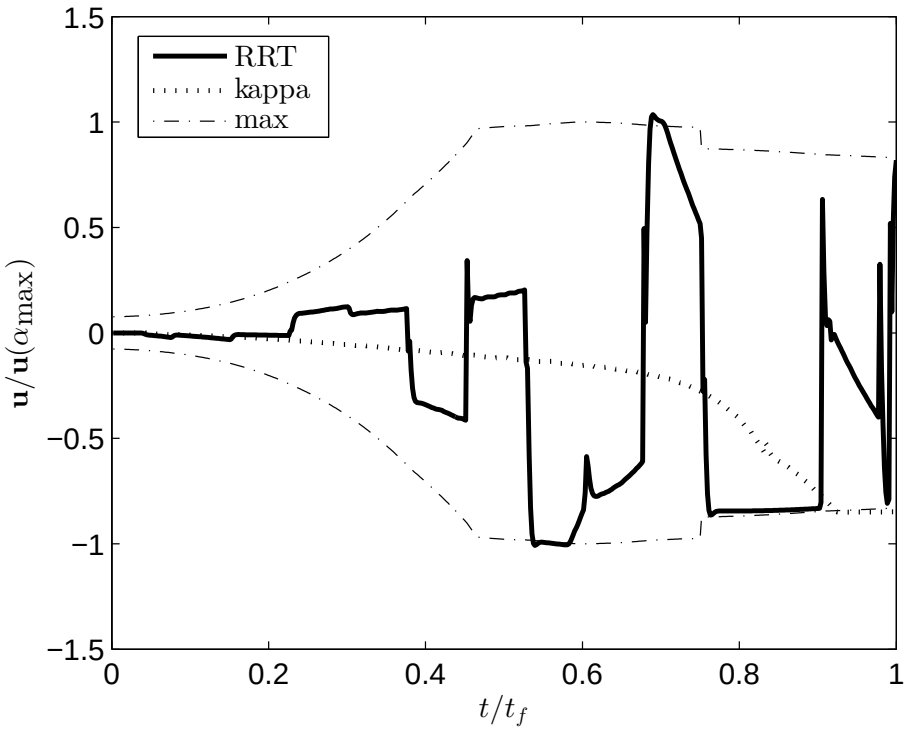
- The optimality neither in path length nor in final speed of the obtained solutions is not guaranteed
- The acquisition of solutions is not guaranteed for the fixed time since a large number of iterations are required in order to solve this problem

In order to improve the performance of this algorithm, a solution to these problems is proposed in the next chapter.

Note that the researches described in this chapter are published in a paper presented in "The 19th IFAC Symposium on Automatic Control in Aerospace" under the title "Sampling-based path planning: a new tool for missile guidance" and in a paper presented in "The IEEE/RSJ International Conference on Intelligent Robots and Systems" under the title "Missile trajectory shaping using sampling-based path planning" in 2013.

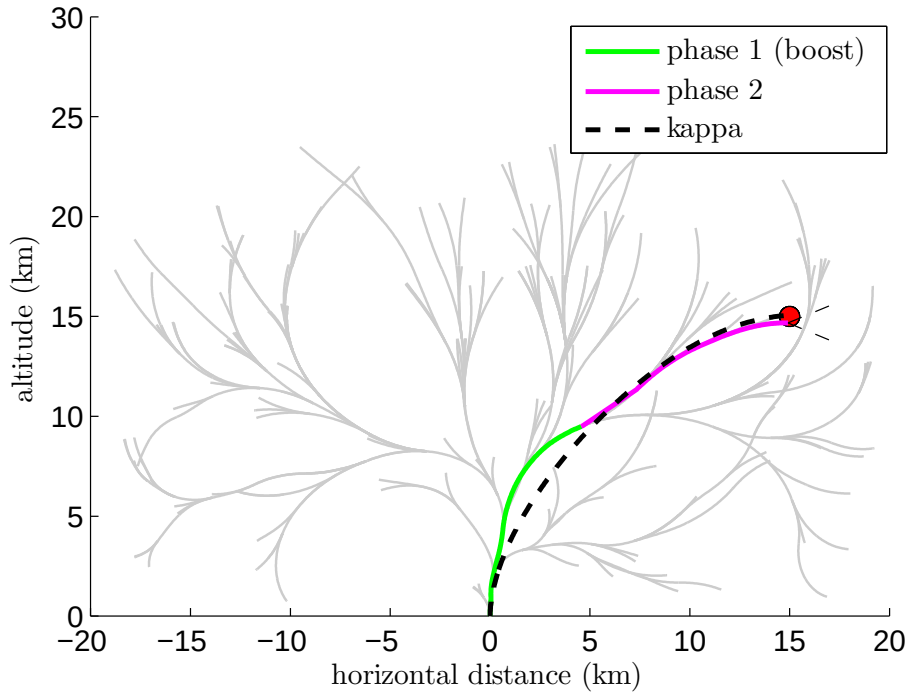


(a) RRT_{PN}: Trajectories

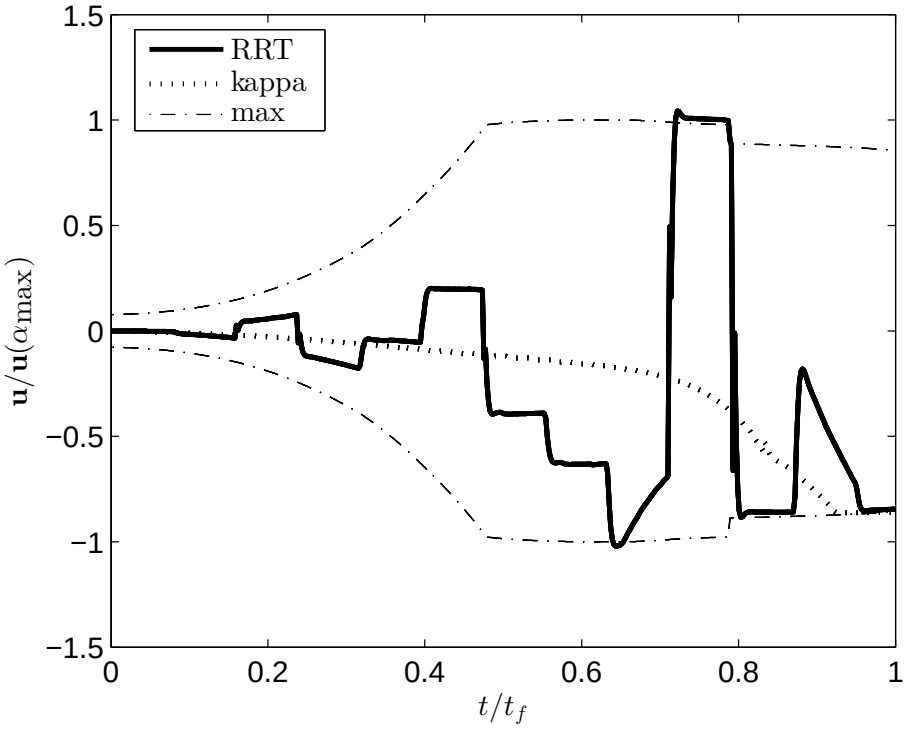


(b) RRT_{PN}: Lateral accelerations along the solution trajectories

Figure 5.6: Simulation results using RRT_{PN} for scenario 1

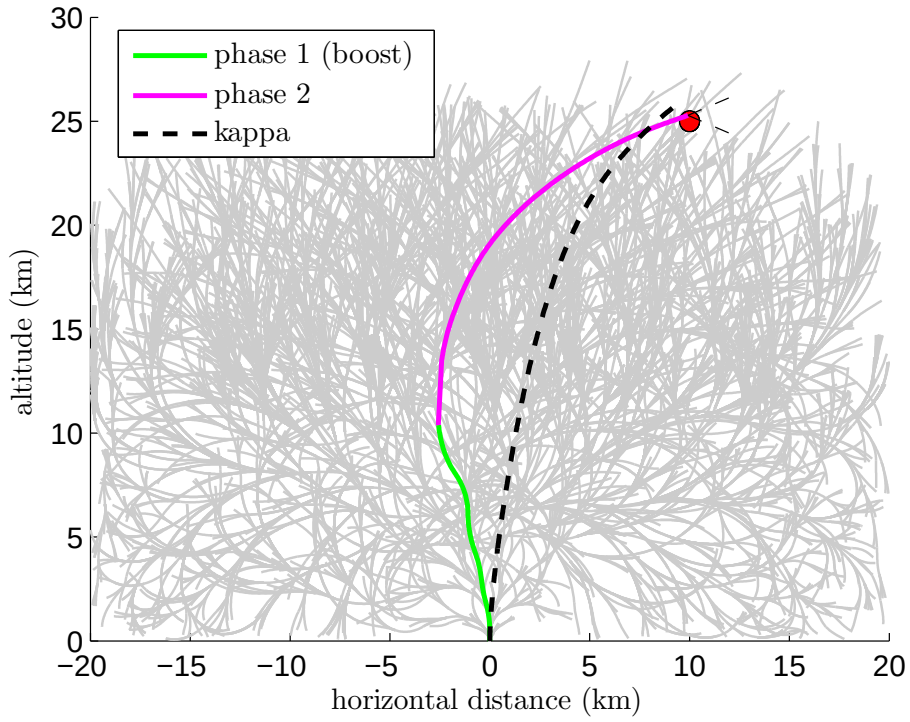


(a) $\text{RRT}_{\text{kappa}}$: Trajectories

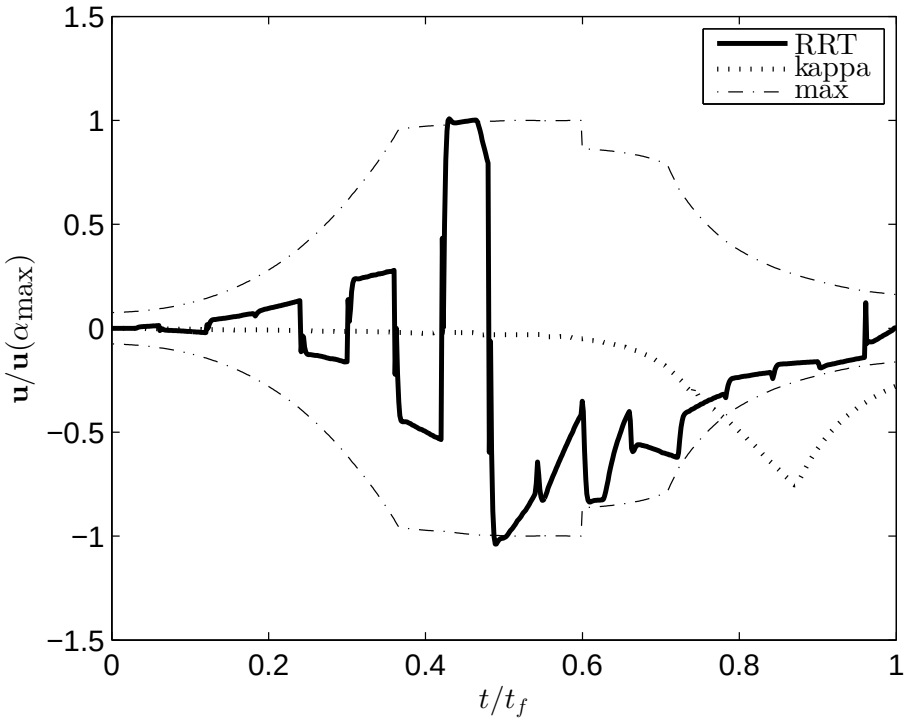


(b) $\text{RRT}_{\text{kappa}}$: Lateral accelerations along the solution trajectories

Figure 5.7: Simulation results using $\text{RRT}_{\text{kappa}}$ for scenario 1

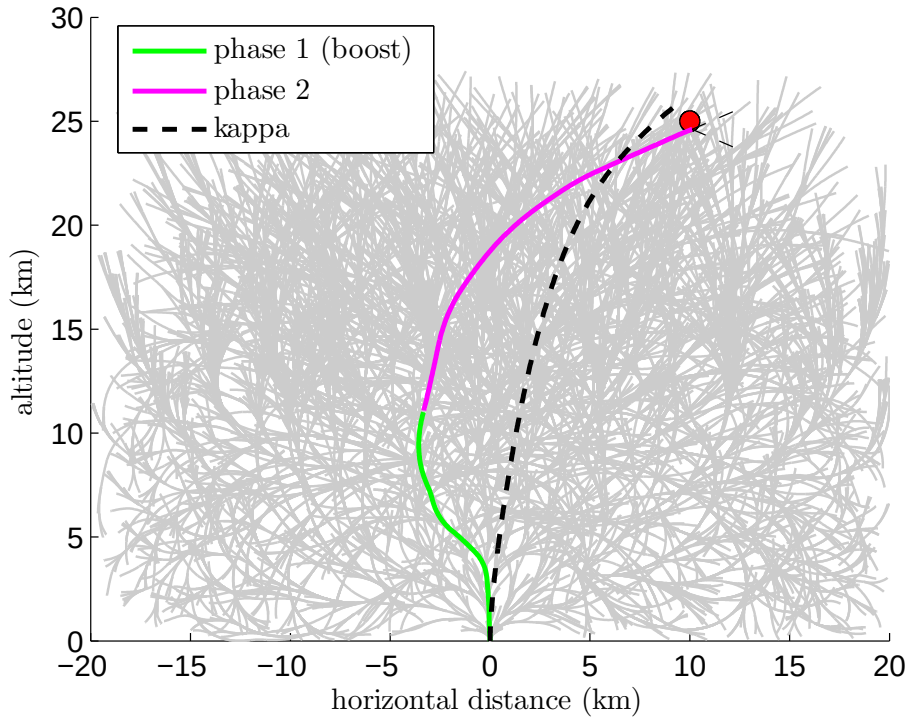


(a) RRT_{PN}: Trajectories

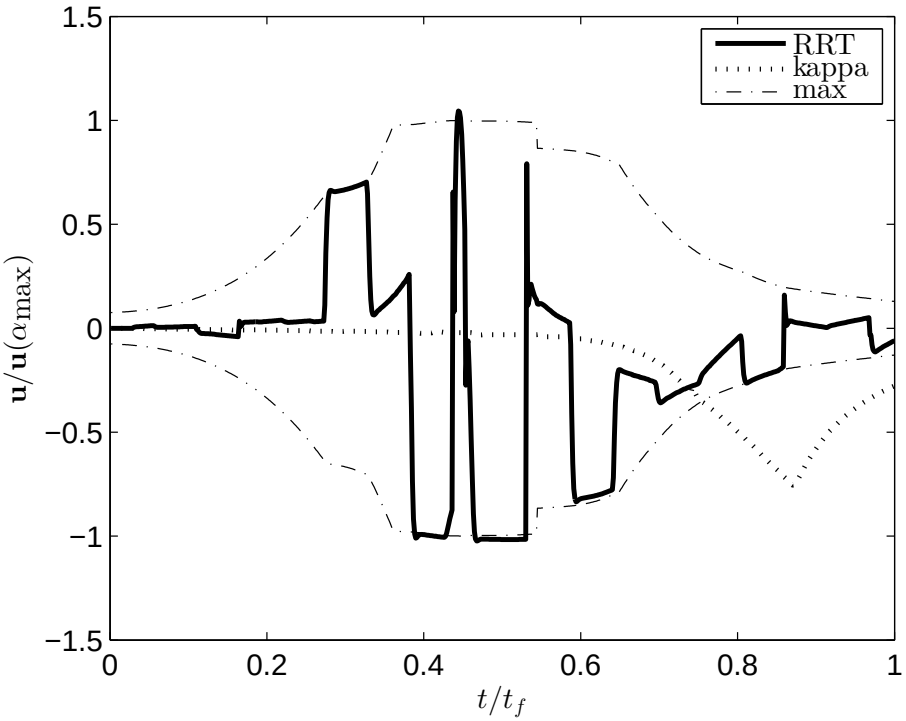


(b) RRT_{PN}: Lateral accelerations along the solution trajectories

Figure 5.8: Simulation results using RRT_{PN} for scenario 2



(a) $\text{RRT}_{\text{kappa}}$: Trajectories



(b) $\text{RRT}_{\text{kappa}}$: Lateral accelerations along the solution trajectories

Figure 5.9: Simulation results using $\text{RRT}_{\text{kappa}}$ for scenario 2

6 RRT path planning with preprocessed exploration space

As a consequence of the previous chapter, the main contribution of this chapter is focused on the methods used to find the preprocessed exploration space. A new pseudometric function related to the preprocessed exploration space is also developed for an interceptor missile application.

The RRT algorithm using the original Dubins' paths [PHPB13, PPHB13] as the metric to determine the distance between two vehicle states in the exploration space is studied in the previous chapter. The results show that an admissible trajectory between $\mathbf{x}_{\text{init}}$ and $\mathbb{X}_{\text{goal}}$ can be found. However, the approach has some drawbacks:

- Since the vehicle model is more complex than the Dubins' car, the metric based on Dubins' car is not perfectly suitable for the trajectory planning of an aerial vehicle
- The optimality of obtained trajectories is not guaranteed neither in path length or in final speed; No optimal criterion is considered
- As the exploration space is large, a large number of iterations are often required to obtain a result

To remediate to these problems, a reduction of the exploration space $\mathbb{X}_{\text{free}}$ is proposed in such a way that it includes the optimal or near-optimal solutions.

In this chapter, the preprocessing methods of the exploration space $\mathbb{X}$ are presented. Then, it is introduced to the RRT framework presented in previous chapter. An interceptor missile is also used as a case study to demonstrate the performance of the proposed algorithm. The demonstration of how to find and use the preprocessed exploration space

for an interceptor missile is explained. Then, some modifications of the RRT algorithm are explained. A pseudometric related to the preprocessing exploration space is also developed. Next, the simulation results are shown and analyzed. Finally, the concluding remarks are made at the end of this chapter.

6.1 Preprocessing of the exploration space $\mathbb{X}$

The idea of using the preprocessed exploration space $\mathbb{X}$ is that, by reducing the exploration space, a solution can be obtained faster. However, please note that the probabilistic methods like RRT algorithm can find a solution if and only if the solution exists. Thus, one of these hypothesis must be hold:

Hypothesis 1 *In order to find a feasible trajectory using a probabilistic method, the preprocessed exploration space $\mathbb{X}$ must contain at least a feasible trajectory.*

Hypothesis 2 *In order for find the optimal trajectory, the preprocessed exploration space $\mathbb{X}$ must contain the optimal trajectory so that obtained trajectories can be improved and approach the optimal solution. Therefore, it can be considered a suboptimal solution.*

Hypothesis 1 is automatically required in order to use this framework to find a solution. In addition, if the optimal solution is requested, the hypothesis 2 must be hold.

There are several ways to preprocess the exploration space. Here, two methods are presented.

6.1.1 Artificial potential field

Artificial Potential Field or APF [AH83, Kha85] is inspired by nature. By assuming that the vehicle is a charged particle inside the electric or magnetic field, the vehicle is moved by the induced forced guided by the vector field. In robotics, the same effect is simulated by creating an artificial potential field that moves the robot to the desired destination.

With help of a good combination of APF, an estimated trajectory from one point to another can be found. Given position and orientation of two vehicle states, several trajectories can be found and used as a preprocessing exploration space (see Appendix C). This method can justify hypothesis 1. However, it is very difficult to find a suitable APF function for the complex system in our case. Moreover, several tuning parameters are required. Thus, it is not suitable for our problem (see Appendix C for more details).

6.1.2 Trajectory generation methods

The objective of preprocessed exploration space to to limit the exploration zone of the algorithm. Thus, any methods that can generate trajectories can be applied for the preprocessing methods. Thus, trajectory generation methods can be used to define the preprocessed exploration space. For example, Bézier curve, Pythagorean Hodographs, polynomial curves, cubic spline, etc. can be used to find the trajectories surrounding the interesting exploration space while justifying one of the mentioned hypothesis.

6.2 Application for an interceptor missile

The algorithm is studied in the same application as in chapter 5, *i.e.* an interceptor missile. The algorithm is looking for a feasible trajectory of an interceptor missile between two vehicle states in 2-dimensional plane. The same system model and problem formulation are used (see section 5.2 for details) and will not be mathematically recalled here. In addition to the framework presented in section 5.2, only the midcourse phase without propulsion of an interceptor missile is studied and the exploration space is preprocessed to obtain smaller but efficient exploration space. For an interceptor missile, the Dubins' paths in a heterogeneous environment is used.

6.2.1 Dubins' paths in a heterogeneous environment

The environment is heterogeneous in the sense that the maximum path curvature of the vehicle is not constant and varies with the position of the vehicle. In [HP13], it is shown that, analogously to Dubins' paths [Dub57][BCL91], shortest paths are a combination of curves of maximum curvature C and straight lines S. Several examples of these Dubins' paths are shown in [HP13]. For the problem considered in this paper, only CSC paths are considered since the distance between $\mathbf{x}_{\text{init}}$ and $\mathbb{X}_{\text{goal}}$ is sufficiently large. The Dubins' paths are proven to be the optimal solution between two vehicles states. Thus, they can be used to find the preprocessed exploration space using two shortest Dubins' paths with the extremity conditions. This method can be easily used to determine the preprocessed exploration space under hypothesis 1. Moreover, the extremity of the preprocessed exploration space is composed by the trajectories proven to be optimal. Then, the hypothesis 2 can be justified by supposing that a space constructed by two optimal solutions (with different condition initial/condition final) also contains optimal solutions.

6.2.2 Preprocessed exploration space using Dubins' paths in a heterogeneous environment

The new admissible exploration space consists of position space $\mathbb{P}$ and velocity space $\mathbb{V}$, *i.e.* $\mathbb{X} = \mathbb{P} \times \mathbb{V}$. $\mathbb{P}$ and $\mathbb{V}$ are defined in the following sections such that $\mathbb{X}_{\text{goal}} \subset \mathbb{X}$ (see definition of $\mathbb{X}_{\text{goal}}$ in section 5.2.5.1).

6.2.2.1 The position space $\mathbb{P}$

An envelope $\mathbb{P}$ that is a subset of $\mathbb{R}^2$, including $\boldsymbol{\xi}_{\text{init}}$ and $\mathbb{P}_{\text{goal}}$ is defined here.

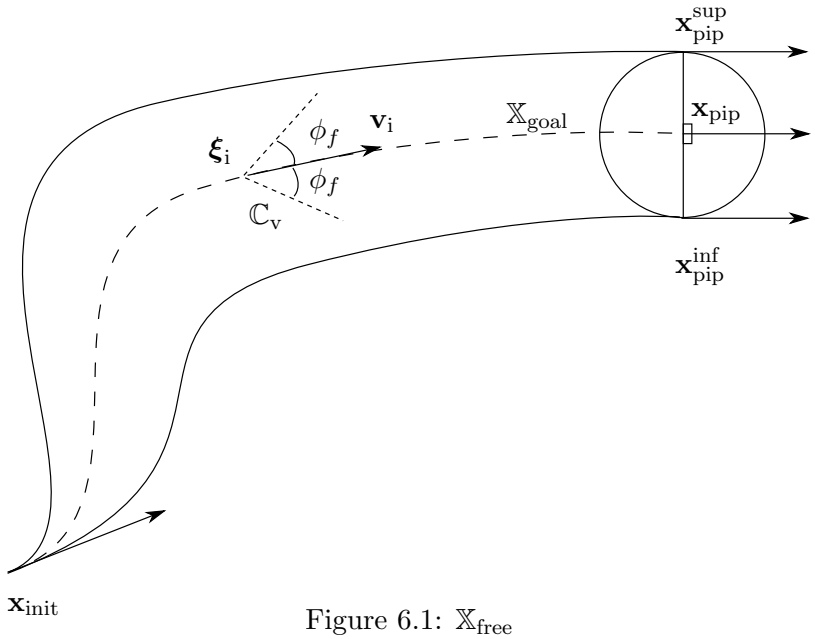


Figure 6.1: $\mathbb{X}_{\text{free}}$

The envelope $\mathbb{P}$ is built from two extremity Dubins' CSC paths in a heterogeneous environment starting at $\mathbf{x}_{\text{init}}$ and ending in $\mathbb{X}_{\text{goal}}$. For this propose, $\mathbf{x}_{\text{pip}}^{\text{sup}}, \mathbf{x}_{\text{pip}}^{\text{inf}} \in \mathbb{X}_{\text{goal}}$, are defined as follows:

- The positions $\boldsymbol{\xi}_{\text{pip}}^{\text{sup}}$ and $\boldsymbol{\xi}_{\text{pip}}^{\text{inf}}$ are both on the sphere of radius R around $\boldsymbol{\xi}_{\text{pip}}$ in the direction perpendicular to $\mathbf{v}_{\text{pip}}$ as shown in figure 6.1;
- The orientations $\gamma_{\text{pip}}^{\text{sup}}$ and $\gamma_{\text{pip}}^{\text{inf}}$ are equal to γ_{pip};
- If two trajectories have their starting path that coincides with each other, $\mathbf{x}_{\text{init}}$ is redefined to be the state $\mathbf{x}_{\text{init}}^{\text{new}}$ where both trajectories separate from each other as shown in figure 6.2. This is done with the assumption that the UAVs can perfectly follow the calculated trajectory.

Once the envelope $\mathbb{P}$ is obtained, it is assumed that an optimal trajectory is contained inside the envelope since the envelope is constructed from two extremity optimal Dubins-like trajectories.

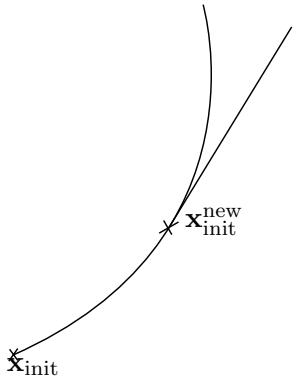


Figure 6.2: The starting trajectories coincide with each other

The admissible exploration space $\mathbb{P}_{\text{free}} = \mathbb{P} \setminus \mathbb{P}_{\text{obs}}$ is the collision-free envelope where $\mathbb{P}_{\text{obs}} \in \mathbb{R}^2$ is a set of positions of the obstacles in the state space (see red dashed curves in figure 6.3).

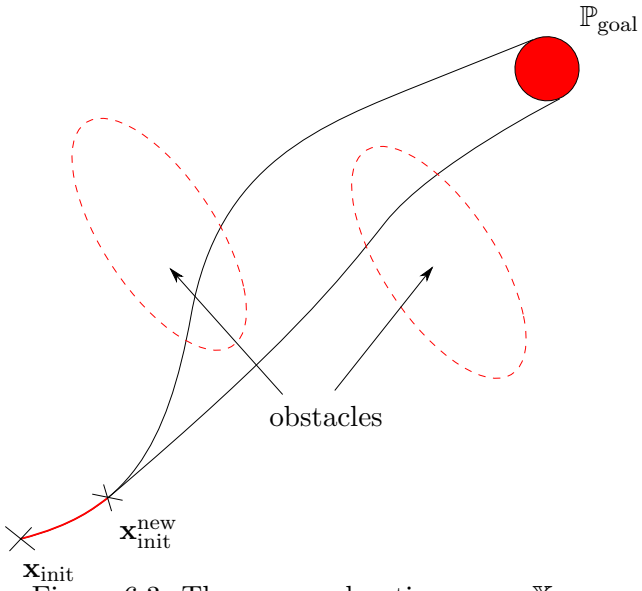


Figure 6.3: The new exploration space $\mathbb{X}$

6.2.2.2 The velocity space $\mathbb{V}$

The velocity space $\mathbb{V}$ is defined as

$$\mathbb{V} = \{(v, \gamma) \in \mathbb{R}^2 : v \geqslant v_{\min}\}, \tag{6.1}$$

where $v_{\min}$ is the acceptable minimal speed of the missile defined by a lethal system to be able to destroy the target.

The admissible velocity space $\mathbb{V}_{\text{free}}$ is defined such that the exploration tree does not waste time searching non-optimal space. Thus, given a position $\boldsymbol{\xi}_{\text{i}} \in \mathbb{P}_{\text{free}}$, the shortest CSC path from $\mathbf{x}_{\text{init}}$ to $\mathbb{X}_{\text{goal}}$ passing by $\boldsymbol{\xi}_{\text{i}}$ is considered (see dashed curve in figure 6.1). Then, by considering the unit vector $\mathbf{e}_1^{\text{v}}$ of the velocity frame $\mathcal{V}$ tangent to this path at $\boldsymbol{\xi}_{\text{i}}$, $\mathbb{V}_{\text{free}}$ is defined as follows:

- $\boldsymbol{\xi}_{\text{i}} \notin \mathbb{P}_{\text{goal}}$,
$$\mathbb{V}_{\text{free}} = \{(v, \gamma) \in \mathbb{R}^2 : v \geqslant v_{\min}, \gamma \in \mathbb{C}(\mathbf{x}_{\text{i}}, \phi_f)\}, \tag{6.2}$$
where $\mathbb{C}(\mathbf{x}_{\text{i}}, \phi_f)$ is the convex cone pointing toward $\mathbf{v}_{\text{i}}$ with apex $\boldsymbol{\xi}_{\text{i}}$ and apex angle $2\phi_f$;
- $\boldsymbol{\xi}_{\text{i}} \in \mathbb{P}_{\text{goal}}$,
$$\mathbb{V}_{\text{free}} = \mathbb{V}_{\text{goal}}^{\text{new}}, \tag{6.3}$$
$\mathbb{V}_{\text{goal}}^{\text{new}}$ is defined in the next section.

6.2.2.3 Definition of $\mathbb{X}_{\text{goal}}$

Once $\mathbb{X}_{\text{free}}$ is defined, $\mathbb{X}_{\text{goal}}$ is refined as $\mathbb{X}_{\text{goal}}^{\text{new}}$ with condition $\mathbb{X}_{\text{goal}} \subset \mathbb{X}_{\text{free}}$. Thus, $\mathbb{X}_{\text{goal}}^{\text{new}} = \mathbb{X}_{\text{goal}} \cap \mathbb{X}_{\text{free}}$:

$$\begin{aligned}
&\mathbb{X}_{\text{goal}}^{\text{new}} = \mathbb{P}_{\text{goal}} \times \mathbb{V}_{\text{goal}}^{\text{new}}, \\
&\mathbb{P}_{\text{goal}} = \{\boldsymbol{\xi} = (x, z)^\top \in \mathbb{R}^2 : \|\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{pip}}\| < R_{\min}\}, \\
&\mathbb{V}_{\text{goal}}^{\text{new}} = \{(v, \gamma) \in \mathbb{R}^2 : v \geqslant v_{\min}, \gamma \in \mathbb{C}(\boldsymbol{\xi}, \gamma_{\text{pip}}, \phi_f^{\text{pip}}) \cap \mathbb{C}(\boldsymbol{\xi}, \gamma, \phi_f^{\text{pip}})\},
\end{aligned} \tag{6.4}$$

where $R_{\min}$ is a radius of a sphere centered at $\boldsymbol{\xi}_{\text{pip}}$ and ϕ_f is an angle related to the maximum capability of the terminal guidance system. Values of $R_{\min}$ and ϕ_f are defined by the lethal system of the missile.

6.2.3 RRT reconfigurations

The RRT algorithm in this chapter uses the RRT framework explained in chapter 5 as a basis. Thus, see section 5.2.5 for detailed configurations. Only the configurations that are modified or different from the ones from section 5.2.5 are presented in this section.

6.2.3.1 Random state generation

The random state $\mathbf{x}_{\text{rand}}$ is generated such that $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$. A bias called RRT-*g*oalbias [LK01] is also used here. It consists in choosing $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$ with a probability p. As

stated in section 6.2.2 that the velocity is generated based on the shortest path from $\mathbf{x}_{\text{init}}$ to $\mathbb{X}_{\text{goal}}$. Here, the methodology of finding the suboptimal trajectories lying inside the envelope for implementation is explained.

Since the envelope $\mathbb{P}$ gives the idea about what the optimal trajectory should be, the ideal way to generate $\mathbf{x}_{\text{rand}} = (\boldsymbol{\xi}_{\text{rand}}^{\top}, v_{\text{rand}}, \gamma_{\text{rand}})^{\top} \in \mathbb{X}_{\text{free}}$ is to find the optimal trajectory, illustrated by a dashed curve in figure 6.1, containing the position $\boldsymbol{\xi}_{\text{rand}}$. Then, the orientation γ_{rand} is generated while respecting the orientation of the suboptimal trajectory with a random marge ϕ_f as shown in figure 6.1. However, it requires a large computational effort to be implemented in the algorithm. Thus, an approximated interpolation is proposed under the following assumptions.

Assumption 1 *Both extremity trajectories must have about the same length. If not, the envelope should be divided into i small envelopes* $\mathbb{P}_i \in \mathbb{P} \;\; i = 1, 2, 3, ..., n.$

Assumption 2 *There always exists a suboptimal trajectory from* $\mathbf{x}_{init}$ *to* $\mathbb{X}_{goal}$ *containing* $\boldsymbol{\xi}_i \in \mathbb{P}$ *(see figure 6.1).*

Assumption 1 and 2 ensure the precise approximation of suboptimal trajectories inside $\mathbb{P}$. The larger the difference of their length, the less precise the approximation is. In order to generate the orientation γ_{rand} of the vehicle, the suboptimal trajectory (see figure 6.4) containing $\boldsymbol{\xi}_{\text{rand}}$ must be determined using the following definitions:

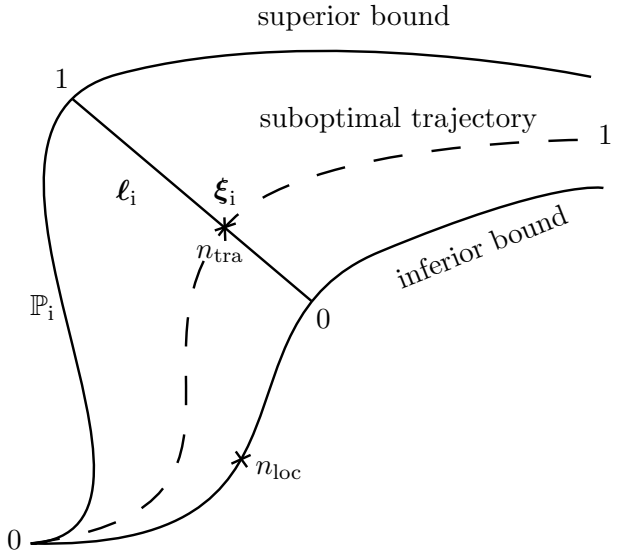


Figure 6.4: References of a point in $\mathbb{X}_{\text{free}}$

Definition 3 ***Location number*** *(n_{loc})*

On each normalized trajectory, the location of a point $\boldsymbol{\xi}$ *is indicated by a variable called*

"location number", i.e. $n_{loc} \in [0,1]$. *The beginning of the trajectory* ($\mathbf{x}_{init}$) *is indicated by* $n_{loc} = 0$ *and the end of the trajectory* ($\mathbf{x}_{goal} \in \mathbb{X}_{goal}$) *is indicated by* $n_{loc} = 1$.

Definition 4 ***Location line***
A "location line" is formed by connecting all points with the same value of n_{loc} *with a curve. It is denoted* $\boldsymbol{\ell}$.

Definition 5 ***Trajectory location number*** (n_{tra})
On each normalized location line between two extremities trajectories of $\mathbb{P}$, *a location of a point* $\boldsymbol{\xi}_i$ *is indicated by a variable called "trajectory location number",* i.e. $n_{tra} \in [0,1]$. *The inferior extremity trajectory is indicated by* $n_{tra} = 0$ *and the superior one is indicated by* $n_{tra} = 1$.

The illustration of these definitions is shown in figure 6.4.

In the following paragraphs, the calculations of n_{loc} and n_{tra} are demonstrated. Then, the position and orientation generation of the random state are explained.

Calculation of n_{loc_i}

According to definitions 3 and 4, if all normalized trajectories are sampling into N points, then there are $N-1$ location lines known along with a point, $\mathbf{x}_{\text{init}}$ (the starting point is the same for both extremity trajectories). The location number of each location line is defined as $n^m_{\text{loc}_1} = \frac{m-1}{N-1}$ where $m = 0, 1/(N-1), 2/(N-1), ..., (N-2)/(N-1), 1$ is a number of sampling points.

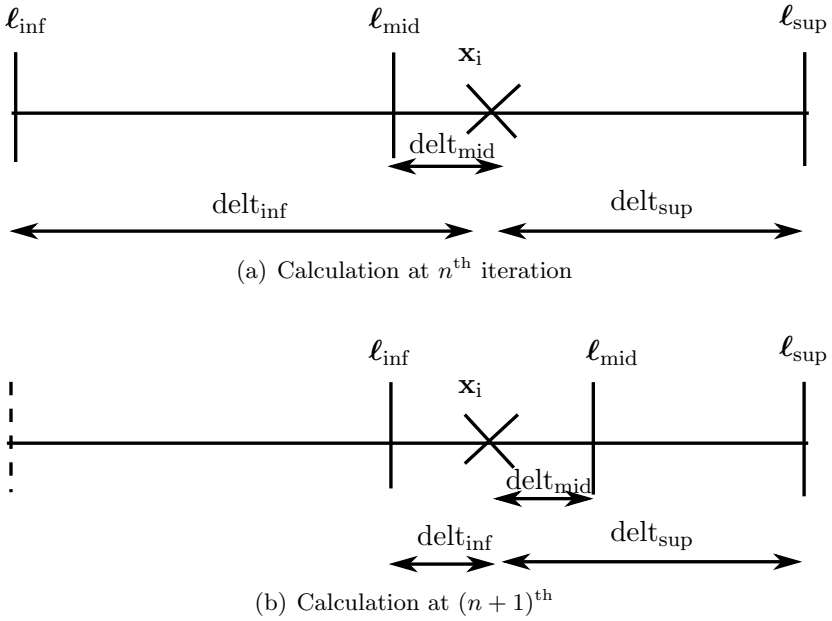
For the sake of locating $\boldsymbol{\xi}_{\text{i}}$, the two closest location lines must be found. The simplest method is to calculate a distance between $\boldsymbol{\xi}_{\text{i}}$ and every location line. The shortest distance $d(\boldsymbol{\xi}_{\text{i}}, \boldsymbol{\ell}_{\text{j}})$ between a point $\boldsymbol{\xi}_{\text{i}} = (x_{\text{i}}, z_{\text{i}})$ and a line $\boldsymbol{\ell}_{\text{j}}$: $a_{\text{j}}x + b_{\text{j}}z + c_{\text{j}} = 0$ can be calculated by

$$d(\boldsymbol{\xi}_{\text{i}}, \boldsymbol{\ell}_{\text{j}}) = \frac{|a_{\text{j}}x_{\text{i}} + b_{\text{j}}z_{\text{i}} + c_{\text{j}}|}{\sqrt{a_{\text{j}}^2 + b_{\text{j}}^2}}. \tag{6.5}$$

Then, the inverse distance weighting (explained later in this section) is used to estimate the location line $\boldsymbol{\ell}_{\text{i}}$ passing by this point $\boldsymbol{\xi}_{\text{i}}$. However, this method takes an enormous computational effort, $8Nn$, where N is a number of sampling points and n is a number of iterations. Thus, the dichotomy paradox illustrated in figure 6.5 is used to find the location line in order to accelerate the calculation.

Suppose that the number of sampling points on each extremity trajectory is $N = 2^k$ (highly recommended) where $k \in \mathbb{N}^+$ is a number of iterations to find a line closest to a given point.

Two distances (see figure 6.5) are required to use the dichotomy paradox:



(a) Calculation at n^{th} iteration

(b) Calculation at $(n+1)^{\text{th}}$

Figure 6.5: Location line calculation

- Distance between the first location line, called $\boldsymbol{\ell}_{\text{inf}}$, here is a point $\mathbf{x}_{\text{init}}$, and a given point $\mathbf{x}_{\text{i}}$, called delt_{inf}
- Distance between the last location line, $\boldsymbol{\ell}_{\text{sup}}$, a location line passing by $\mathbf{x}_{\text{goal}}$, and a given point $\mathbf{x}_{\text{i}}$, called delt_{sup}

Then, the location number $n_{\text{loc}_{\text{i}}}$ of $\boldsymbol{\xi}_{\text{i}}$ can be calculated by following the instruction below:

1. Let $\boldsymbol{\ell}_{\text{mid}}$ denote a location line in the middle of $\boldsymbol{\ell}_{\text{inf}}$ and $\boldsymbol{\ell}_{\text{sup}}$ in figure 6.5(a). Then, a distance between $\boldsymbol{\ell}_{\text{mid}}$ and $\boldsymbol{\xi}_i$ is denoted delt_{mid}. This line $\boldsymbol{\ell}_{\text{mid}}$ becomes line_{inf} if $\text{delt}_{\text{inf}} > \text{delt}_{\text{sup}}$ and vice-versa (see figure 6.5(b)). This step is repeated until the two closest location lines to $\boldsymbol{\xi}_{\text{i}}$ are obtained, *i.e.* k iterations.

2. The inverse distance weighting is used to calculated the location line passing by $\mathbf{x}_{\text{i}}$. Then, the location number $n_{\text{loc}_{\text{i}}}$ of $\boldsymbol{\xi}_{\text{i}}$ can be expressed as:

$$n_{\text{loc}_{\text{i}}} = \frac{n_{\text{loc}}^{\text{inf}} \text{delt}_{\text{sup}} + n_{\text{loc}}^{\text{sup}} \text{delt}_{\text{inf}}}{\text{delt}_{\text{sup}} + \text{delt}_{\text{inf}}}, \tag{6.6}$$

   where $n_{\text{loc}}^{\text{inf}}$ is the location number of $\boldsymbol{\ell}_{\text{inf}}$ and $n_{\text{loc}}^{\text{sup}}$ be the location number of $\boldsymbol{\ell}_{\text{sup}}$.

The computational effort of this method is only $(16 + 8N^{1/k})n$ which is much faster than $8Nn$ for $N >> 0$.

Calculation of n_{tra_1}

The value of n_{tra_1} can be calculated by using the formulation below:

$$n_{\text{traj}_\text{i}} = \frac{d_\text{i}^{\text{inf}}}{d_\text{i}^{\text{inf}} + d_\text{i}^{\text{sup}}},$$

where d_i^{inf} is a distance between $\boldsymbol{\xi}_\text{i}$ and an intersection between the inferior bound and the location line containing $\boldsymbol{\xi}_\text{i}$ and d_i^{sup} is a distance between $\boldsymbol{\xi}_\text{i}$ and an intersection between the superior bound and the location line containing $\boldsymbol{\xi}_\text{i}$.

With n_{loc} and n_{tra}, the random state $\mathbf{x}_{\text{rand}}$ can be generated.

- **Position generation:** Position $\boldsymbol{\xi}_{\text{rand}}$ of a random state $\mathbf{x}_{\text{rand}}$ is generated within the envelope (see figure 6.1). Since we do not have an analytic solution for the envelope, it is very hard to verify if the generated position is within the envelope. Thus, we propose the following approximated method.

  Once the location line containing $\boldsymbol{\xi}_{\text{rand}}$ is identified using the method from figure 6.5, five more information can be found:

  - The length $L_{\text{rand}}^{\text{loc}}$ of this location line
  - The intersection point $\boldsymbol{\xi}_{\text{sup}}$ between this location line and the superior extremity trajectory of the envelope $\mathbb{P}$
  - The intersection point $\boldsymbol{\xi}_{\text{inf}}$ between this location line and the inferior extremity trajectory of the envelope $\mathbb{P}$
  - A distance $d_{\text{rand}}^{\text{sup}}$ between $\boldsymbol{\xi}_{\text{rand}}$ and $\boldsymbol{\xi}_{\text{sup}}$
  - A distance $d_{\text{rand}}^{\text{inf}}$ between $\boldsymbol{\xi}_{\text{rand}}$ and $\boldsymbol{\xi}_{\text{inf}}$

  By using these variables, the randomly generated point $\boldsymbol{\xi}_{\text{rand}} \in \mathbb{P}$ if $d_{\text{rand}}^{\text{sup}}$ and $d_{\text{rand}}^{\text{inf}}$ are inferior or equal to $L_{\text{rand}}^{\text{loc}}$.

- **Orientation generation:** Once $\boldsymbol{\xi}_{\text{rand}} \in \mathbb{P}$ is obtained, the reference orientation γ of the suboptimal trajectory containing $\boldsymbol{\xi}_{\text{rand}}$ at $\boldsymbol{\xi}_{\text{rand}}$ can be estimated by using the inverse distance weighting as follows (see figure 6.6):

$$\mathbf{e}_\text{i} = \frac{\mathbf{e}_\text{i}^{\text{sup}} d_{\text{rand}}^{\text{inf}} + \mathbf{e}_\text{i}^{\text{inf}} d_{\text{rand}}^{\text{sup}}}{d_{\text{rand}}^{\text{inf}} + d_{\text{rand}}^{\text{sup}}} \tag{6.7}$$

  where $\mathbf{e}_\text{i}$ is the orientation at the decided position, $\mathbf{e}_\text{i}^{\text{sup}}$ is the orientation at $\boldsymbol{\xi}_{\text{sup}}$ and $\mathbf{e}_\text{i}^{\text{inf}}$ is the orientation at $\boldsymbol{\xi}_{\text{inf}}$.

  Then, γ_{rand} is generated using $\mathbf{e}_{\text{rand}}$ within a given margin of error ϕ_f.

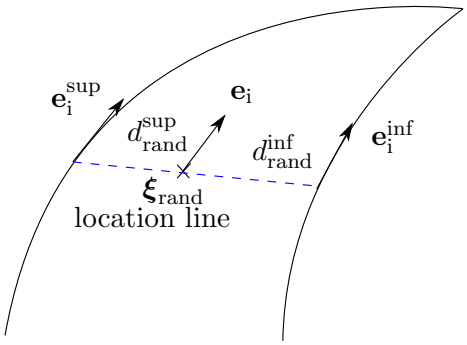


Figure 6.6: Approximated orientation calculation

Finally, the random state $\mathbf{x}_{rand}$ is obtained in such a way that $\mathbf{x}_{rand} \in \mathbb{X}_{free}$. A bias consisting in choosing $\mathbf{x}_{rand} \in \mathbb{X}_{goal}$ with a probability p is also chosen (RRT-*GoalBias* [LK01]) to accelerate the convergence to the solution.

6.2.3.2 Pseudometric

The Euclidean metric is not suitable for this problem since it does not consider any properties of each state other than its position. Moreover, Dubins' metric (see section 5.2.4) is calculated using a simplified system. Therefore, it is not the best metric to be used. Thus, a new pseudometric is developed, while respecting definition 6, to define the nearest neighbor to $\mathbf{x}_{rand}$.

Definition 6 ***Pseudometric***

These three properties must be considered while respecting the pseudometric conditions:

1. $d(\mathbf{x}_1, \mathbf{x}_2) \geqslant 0$ *(non-negativity);*
2. $d(\mathbf{x}_1, \mathbf{x}_1) = 0$*;*
3. $d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_2, \mathbf{x}_1)$ *(symmetry);*
4. $d(\mathbf{x}_1, \mathbf{x}_3) \leqslant d(\mathbf{x}_1, \mathbf{x}_2) + d(\mathbf{x}_2, \mathbf{x}_3)$ *(subadditivity/triangle inequality).*

Note that, one property of metric, which is $d(\mathbf{x}_1, \mathbf{x}_2) = 0$ if and only if $\mathbf{x}_1 = \mathbf{x}_2$, is omitted. It implies that a state in a pseudometric does not need to be distinguishable.

The objective of the wished pseudometric is to define the nearest state while considering three conditions:

- The distance between two states must be considered as all other metrics

- The dynamics of the missile must be considered, *i.e.* less maneuver are preferred over more maneuver
- The velocity of the missile must be considered, *i.e.* high speed missile is preferred over slow speed missile

According to assumption 1 and 2, there always exists an optimal trajectory inside the envelope $\mathbb{P}$ connecting $\mathbf{x}_{\text{init}}$ to $\mathbb{X}_{\text{goal}}$ including $\boldsymbol{\xi}$. Thus, for two position $\boldsymbol{\xi}_1$ with a velocity vector $\mathbf{v}_1$ and $\boldsymbol{\xi}_2$ with a velocity vector $\mathbf{v}_2$, there exists two dotted trajectories as shown in figure 6.7.

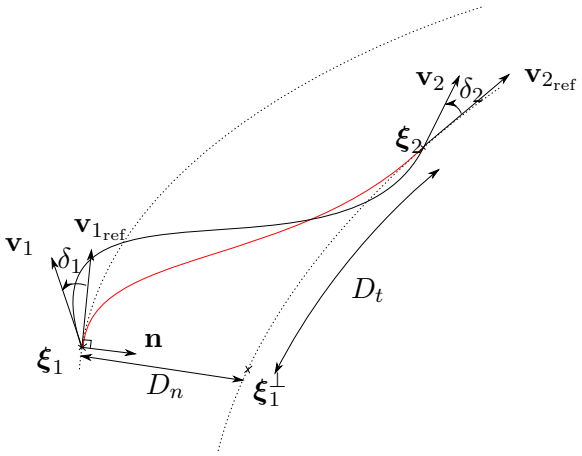


Figure 6.7: Illustration of a connecting trajectory between two state

In figure 6.7, the orientation of the velocity is represented by an arrow. The velocities $\mathbf{v}_{1_{\text{ref}}}$ and $\mathbf{v}_{2_{\text{ref}}}$ are the velocities of reference trajectories including $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$, respectively. The position $\boldsymbol{\xi}_1^{\perp}$ is a projection of $\boldsymbol{\xi}_1$ on another trajectory. D_n is a distance between two trajectories from $\boldsymbol{\xi}_1$ to $\boldsymbol{\xi}_1^{\perp}$ in $\mathbf{n}$ direction. D_t is a length of trajectory from $\boldsymbol{\xi}_1^{\perp}$ to $\boldsymbol{\xi}_2$. δ_1 is an angle from $\mathbf{v}_1$ to $\mathbf{v}_{1_{\text{ref}}}$, the same goes for δ_2.

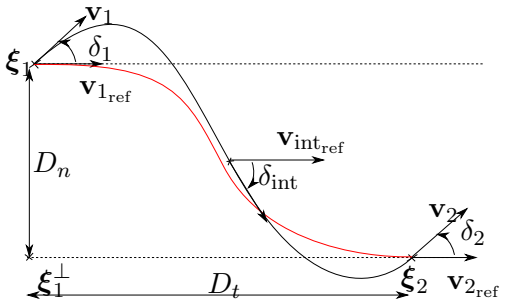


Figure 6.8: Illustration of a connecting trajectory between two state on different parallel linear trajectories

Supposing that these two states are sufficiently far from each other, these two trajectories can be considered parallel linear trajectories. Let's study a particular case shown in figure 6.8 where $\mathbf{v}_{\text{int}_{\text{ref}}}$ is an orientation tangent to the optimal trajectory at the intermediate position and δ_{int} is an angle from the corresponding trajectory to $\mathbf{v}_{\text{int}_{\text{ref}}}$. In order to calculate a distance between two vehicle states, a simplified system of a missile is considered and shown below:

$$\begin{aligned} \dot{y} &= v \sin\delta \approx v\delta \\ \dot{\delta} &= ucv \end{aligned} \tag{6.8}$$

where y is the position on $\mathbf{n}$-axis, $u \in \{-1, 1\}$ is the control input, c is the curvature of the missile, v is considered constant speed, and δ is considered a small angle. Since c varies during the flight, an average curvature is used in this case, *i.e.* $c = c_{\text{av}} = \frac{1}{z_2 - z_1}\int_{z_1}^{z_2} c(z)dz$.

After integrating the system , we have

$$\begin{aligned} \delta - \delta_1 &= ucvt \\ \Delta y &= \frac{ucv^2t^2}{2} + \delta_1 vt \end{aligned} \tag{6.9}$$

The essential total time T to move from $\mathbf{x}_1$ to $\mathbf{x}_2$ is given by considering $D_t = vT$. As v is a constant, we have $T = D_t/v$. The control strategy for pseudometric calculation is shown in figure 6.9. The control $u = u$ is applied during the first part of the trajectory and then $u = -u$ is applied for the second part.

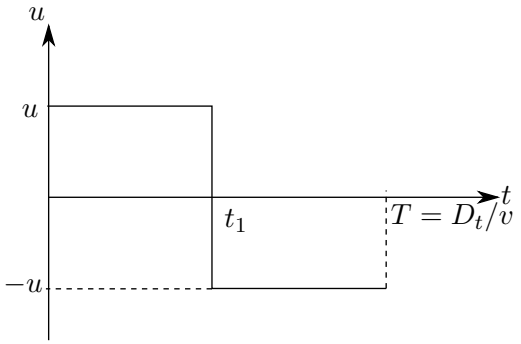


Figure 6.9: Control strategy for pseudometric calculation

If $\delta_0 = \delta_1 \neq 0$ and $\delta_0 = \delta_2 \neq 0$ as shown in the black line in figure 6.8, the trajectory is divided into two arcs of circle at δ_{int} at time t_1 with control input $u = u$ for the first arc and $u = -u$ for the second.

- $u = u$:

$$\delta_{\text{int}} = \delta_1 + uvct_1 \tag{6.10}$$

$$\Delta y(t_1) = v\delta_1 t_1 + u\frac{v^2 c}{2}t_1^2 \tag{6.11}$$

- $u = -u$:

$$\delta_2 = \delta_{\text{int}} - uvc(T - t_1) \tag{6.12}$$

$$\Delta y(T - t_1) = v\delta_{\text{int}}(T - t_1) - u\frac{v^2 c}{2}(T - t_1)^2 \tag{6.13}$$

By substituting δ_{int} from equation (6.10) in equations (6.12) and (6.13), we have

$$t_1 = \frac{T + \frac{\delta_2 - \delta_1}{uvc}}{2} \tag{6.14}$$

Then, by substituting δ_{int} again in the summation of equations (6.11) and (6.13), we have

$$\Delta y(T) = \Delta y(t_1) + \Delta y(T - t_1) = -\frac{ucv^2}{2}T^2 + v\delta_1 T + 2ucv^2 t_1 T - ucv^2 t_i^2 \tag{6.15}$$

With equations (6.14) and (6.15), we have

$$\Delta y_{\max} = \frac{1}{4}ucD_t^2 + \frac{D_t(\delta_2 + \delta_1)}{2} - \frac{(\delta_2 - \delta_1)^2}{4uc}$$

According to the fact that $\delta_1 = \gamma_1 - \gamma_{1_{\text{ref}}}$ and $\delta_2 = \gamma_2 - \gamma_{2_{\text{ref}}}$, we finally have

$$\Delta y_{\max} = \frac{1}{4}ucD_t^2 + \frac{D_t(\gamma_2 + \gamma_1 - \gamma_{1_{\text{ref}}} - \gamma_{2_{\text{ref}}})}{2} - \frac{(\gamma_2 - \gamma_1 + \gamma_{1_{\text{ref}}} - \gamma_{2_{\text{ref}}})^2}{4uc} \tag{6.16}$$

where $D_t = vT$. This result is also applicable for the general case shown in figure 6.7. Once $\Delta y_{\max}$ is obtained, the pseudometric function $d(\mathbf{x}_1, \mathbf{x}_2)$ is processed as follows:

1. If $\text{sign}(\Delta y_{\max}) \neq \text{sign}(D_n)$, it means that it is not possible to move from one state to another. Thus,

$$d(\mathbf{x}_1, \mathbf{x}_2) = \infty \tag{6.17}$$

2. If $\text{sign}(\Delta y_{\max}) = \text{sign}(D_n)$ and $||D_n|| \leqslant ||\Delta y_{\max}||$, it means that it is possible to move from one state to another. Thus,

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\alpha\left(\frac{D_t}{D_{t_{\max}}}\right)^2 + \left(\frac{D_n}{\Delta y_{\max}}\right)^2} / \frac{v_{\text{av}}}{v_{\max}} \tag{6.18}$$

where $D_{t_{\max}}$ is the maximal total distance of the optimal trajectory connecting $\mathbf{x}_{\text{init}}$ and $\mathbb{X}_{\text{goal}}$, v_{av} is the estimated average velocity of the missile which is presented in Appendix E, $v_{\max}$ is the maximal velocity of the missile and α is the tuning parameter.

Justification of pseudometric calculation

There are several possible choices to define a pseudometric. Here, equation (6.18) is chosen while considering the three primary conditions previously mentioned:

- The first condition, *i.e.* distance between two states, for defining the pseudometric is respected by the first term of equation (6.18)
- The second condition, *i.e.* dynamics of the missile, is respected by the second term of equation (6.18). If D_n is larger while $\Delta y_{\max}$ is the same, it means that the state with the larger D_n requires less maneuver. Thus, that state is preferred over another
- The third condition, *i.e.* velocity of the missile, is considered by the third term of equation (6.18)

All terms are normalized by their maximal values in order to be compatible to one another. Since the first term stands for the distance in an axis and the second term stands for the perpendicular distance to that axis, it is logical to use Pythagorean distance to calculate the total distance between these two states. By dividing this Pythagorean distance by the normalized speed, the pseudometric is equivalent to the time to go from one state to another.

6.2.3.3 Node expansion method

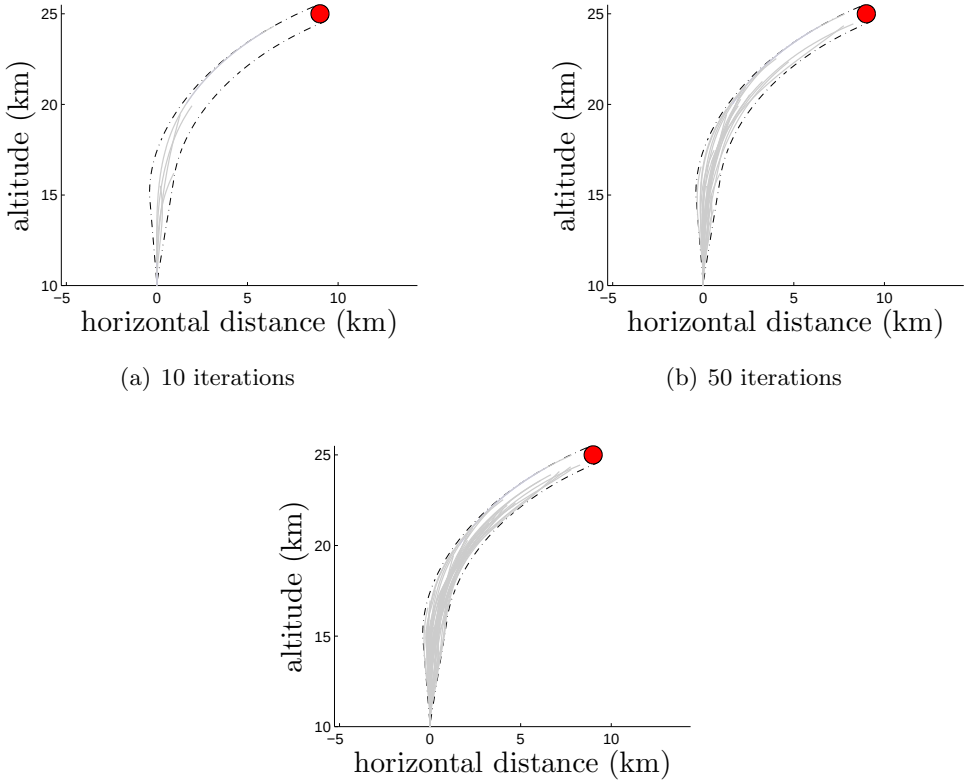
Kappa guidance law, cf. equation (5.15), is used to select the control input $\mathbf{u}_{\text{com}}$ to move the vehicle from $\mathbf{x}_{\text{near}}$ to $\mathbf{x}_{\text{rand}}$. The desired control input $\mathbf{u}_{\text{c}} \in \mathbb{U}(t, \mathbf{x})$ is obtained after saturating $\mathbf{u}_{\text{com}}$ by the control loop that satisfies the system constraints. A new state $\mathbf{x}_{\text{new}}$ is obtained after integrating the system model while considering the dynamics of the angle of attack α until the closest state to $\mathbf{x}_{\text{rand}}$ is found.

6.2.3.4 Collision tests

The collision tests are used to verify that the path between $\mathbf{x}_{\text{near}}$ and $\mathbf{x}_{\text{new}}$ lie in $\mathbb{X}_{\text{free}}$. If it is collision-free, the path between $\mathbf{x}_{\text{near}}$ and $\mathbf{x}_{\text{new}}$ is added to the tree G.

6.3 Simulation results

The framework presented in this chapter is simulated in several scenarios. The unpowered interceptor missile during the midcourse phase is considered. The missile speed at the end



(a) 10 iterations

(b) 50 iterations

(c) 100 iterations

Figure 6.10: Preprocessed RRT expansion

of the boost phase is approximately 2000m/s. Every scenario is analyzed with the initial state $\mathbf{x}_{\text{init}} = (0, 10\text{km}, 2000\text{km/s}, \pi/2)^\top$,

$$\begin{aligned} \mathbb{P}_{\text{goal}} &= \{\boldsymbol{\xi} = (x, z)^\top \in \mathbb{R}^2 : \|\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{pip}}\| < 500\text{m}\}, \\ \mathbb{V}_{\text{goal}} &= \{(v, \gamma) \in \mathbb{R}^2 : v \geqslant 500\text{m/s}, \gamma \in \mathbb{C}(\mathbf{x}_{\text{pip}}, \pi/8)\}, \end{aligned} \tag{6.19}$$

The configurations of each scenario are:

- Scenario 1: $\boldsymbol{\xi}_{\text{pip}} = (9\text{km}, 25\text{km})^\top$, and $\mathbf{v}_{\text{pip}}$ is parallel to the ground ($\gamma_{\text{pip}} = 0$) without obstacle
- Scenario 2: $\boldsymbol{\xi}_{\text{pip}} = (15\text{km}, 20\text{km})^\top$, and $\gamma_{\text{pip}} = 0$ with a presence of a stationary radar detection zone as an obstacle
- Scenario 3: $\boldsymbol{\xi}_{\text{pip}} = (20\text{km}, 17.5\text{km})^\top$, and $\gamma_{\text{pip}} = -\pi/12$ with a presence of several stationary obstacles such as radar detection zone, forbidden area, and altitude restriction zone

The difficulty to solve the problem increases from scenario 1 to 3. The starting orientation at $\pi/2$ is selected in order not to biased toward neither forward nor backward direction.

Moreover, the dynamics response of the angle of attack α is parameterized with a damping ratio $\zeta = 0.8$ and an angular frequency $\omega_0 = 16$rad/s corresponding to a delay of 0.2s. The probability to generate $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$ is set to $p = 0.01$ (this is usually a default value used by the RRT algorithm).

Two algorithms are considered and visualized by MATLAB simulations:

- $\text{RRT}_{\text{kappa}}$ (cf. Chapter 5), *i.e.* an RRT algorithm using the shortest Dubins' path as a metric and kappa guidance law as a control selection
- $\text{RRT}_{\text{prepro}}$ framework from this chapter

Figure 6.10 illustrates RRT expansion using this framework.

In the following figures, the preprocessed exploration space $\mathbb{X}_{\text{free}}$ is enveloped by dashdotted curves while the obstacles are presented by the red dashed curves. The red circle with two dashed lines represent $\mathbb{X}_{\text{goal}}$. The exploration tree is represented in grey and the magenta curve represents a path between the original $\mathbf{x}_{\text{init}}$ and the separation point of the two extremities of the envelope of $\mathbb{X}_{\text{free}}$ which becomes the new starting point $\mathbf{x}_{\text{init}}^{\text{new}}$ of the algorithm.

Figures 6.11, 6.12, and 6.13 show the simulation results obtained by using $\text{RRT}_{\text{kappa}}$ framework on each scenario. They are the results obtained after 5000 iterations and the feasible trajectories are hardly found. The reason is that the missile only uses aerodynamic forces to control its path, the loss of velocity along the path is very critical. The random sampling in large exploration space makes the missile unnecessarily lose its velocity which can result in not finding a feasible solution. On the other hand, figures 6.14, 6.15, and 6.16 show the simulation results obtained by using the $\text{RRT}_{\text{prepro}}$. Only in the first scenario that the result is compared to the one obtained using kappa guidance with the desired flight path angle $\gamma = \phi_f = \pi/8$ which is the easiest arrival condition at $\mathbf{x}_{\text{goal}}$.

In figures 6.14(a), 6.15(a), and 6.16(a), $\mathbb{X}_{\text{free}}$ is illustrated in dashdotted curve, the dotted curve represents the result of the kappa guidance, the solid curve represents the result of the $\text{RRT}_{\text{prepro}}$. Figures 6.14(b), 6.15(b), and 6.16(b) present their lateral accelerations along the computed trajectory along with the maximal control inputs along the trajectories represented by the dashdotted curves.

In scenario 1 (see figure 6.14), the trajectory generated by the kappa guidance fails to reach $\mathbb{X}_{\text{goal}}$ because it does not anticipate the lack of maneuverability at the end of the trajectory (see figure 6.14(b)). On the contrary, as the $\text{RRT}_{\text{prepro}}$ anticipates the loss

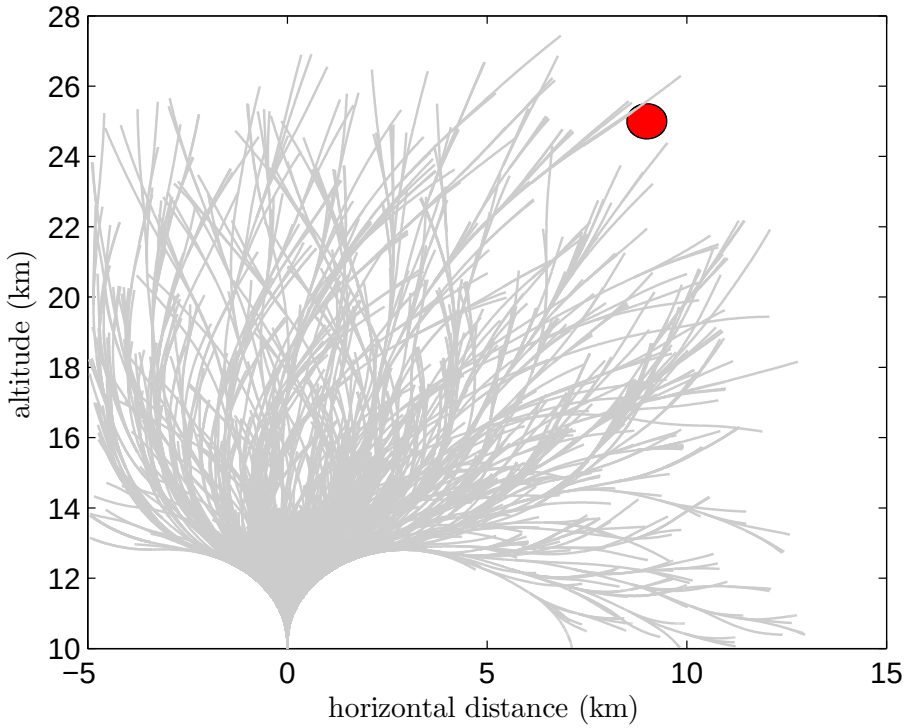


Figure 6.11: Scenario 1: Simulation result using $\text{RRT}_{\text{kappa}}$

of maneuverability at high altitude near $\mathbb{X}_{\text{goal}}$, the interceptor is maneuvered before this loss happens to reduce the curvature of the trajectory when approaching $\mathbb{X}_{\text{goal}}$. As a consequence, an admissible trajectory that satisfies the final constraints is found.

For in-depth analysis, the numerical results are obtained by simulating 100 Monte-Carlo simulations. Each simulation is launched for 2000 iterations. After 100 Monte-Carlo simulations for scenario 1, the solutions are found around the 46^{th} iteration with the average final speed of 907m/s. The optimal solution obtained by using GPOPS [RBD+10] has the final velocity equal to 1117.5m/s. It implies that the $\text{RRT}_{\text{prepro}}$ is capable of finding a solution closer to the optimal one with less number of iterations than the $\text{RRT}_{\text{kappa}}$.

An obstacle is added in scenario 2. Figure 6.15 shows one of its simulation results. According to 100 Monte-Carlo simulations, the $\text{RRT}_{\text{prepro}}$ is capable of finding a solution within the average of 114 iterations. The average final velocity is 762m/s that respects the final constraints represented by a dashed cone at the end of the trajectory.

Scenario 3 is more difficult to solve than the first one because there are more obstacles. One of the simulation results for this scenario is presented in figure 6.16. A solution is found within the average of 469 iterations with the average final velocity of 596m/s as a result of 100 Monte-Carlo simulations.

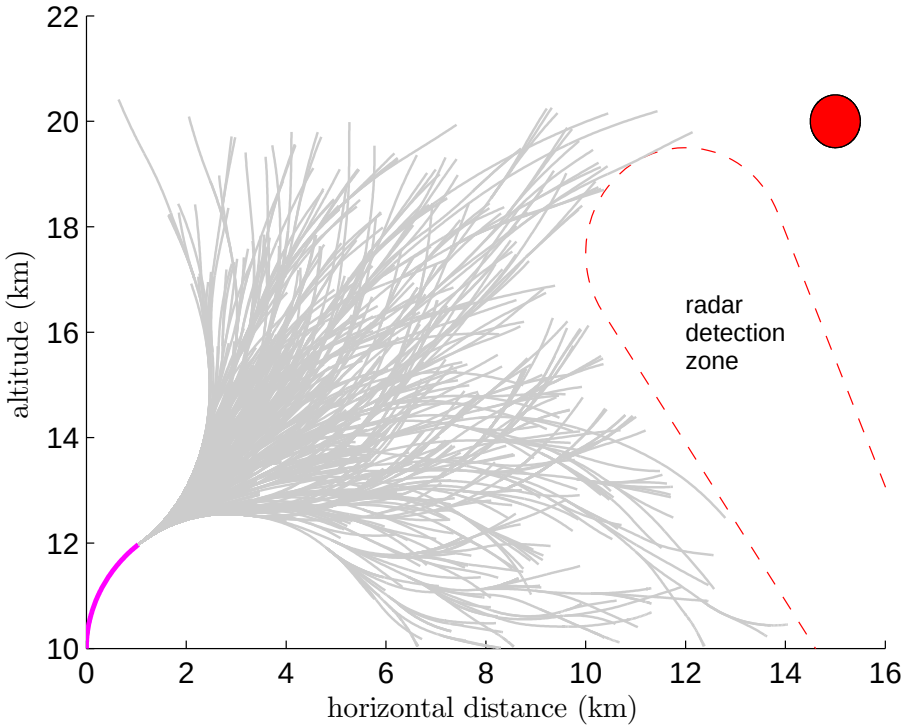


Figure 6.12: Scenario 2: Simulation result using $\text{RRT}_{\text{kappa}}$

These results show that the RRT framework together with the preprocessed exploration space is capable of finding a feasible solution for an aerial vehicle flying in an environment cluttered with obstacles that the classical methods cannot.

6.4 Conclusions

The simulation results show that the preprocessed exploration space ameliorates the computing efficiency and the optimality of the RRT algorithm. However, there are some critical drawbacks and problems:

- There is no theoretical proof that the envelope of preprocessed exploration space contains an optimal or near-optimal solutions
- The preprocessed exploration space does not take the obstacle zones into consideration; This can lead to two or more completely separated exploration space that contains no feasible solution
- The preprocessed exploration space is some sort of an estimated initial configuration for the optimal control problem; Moreover, the exploration space becomes smaller;

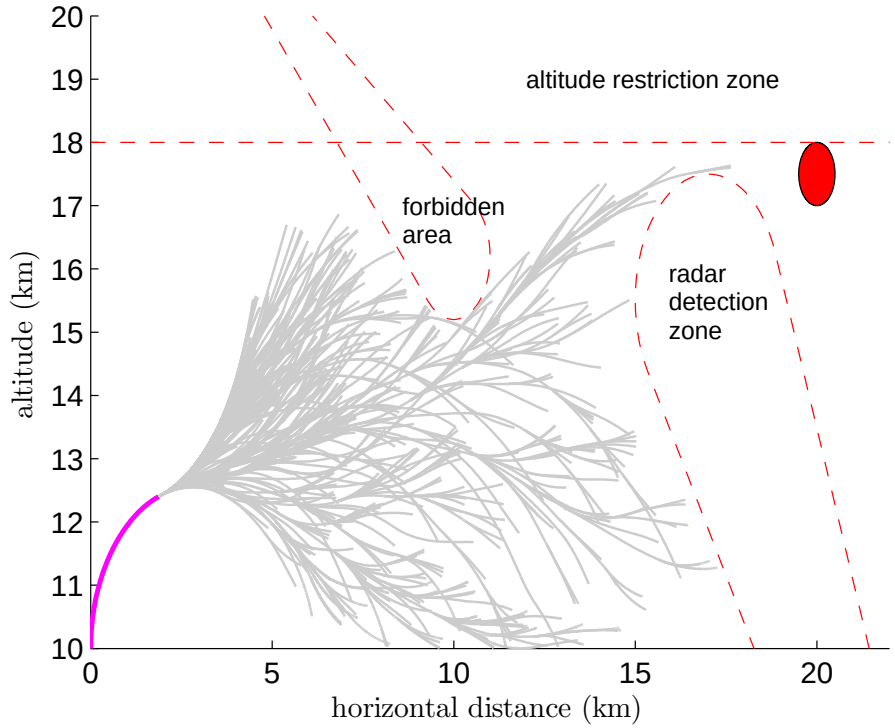
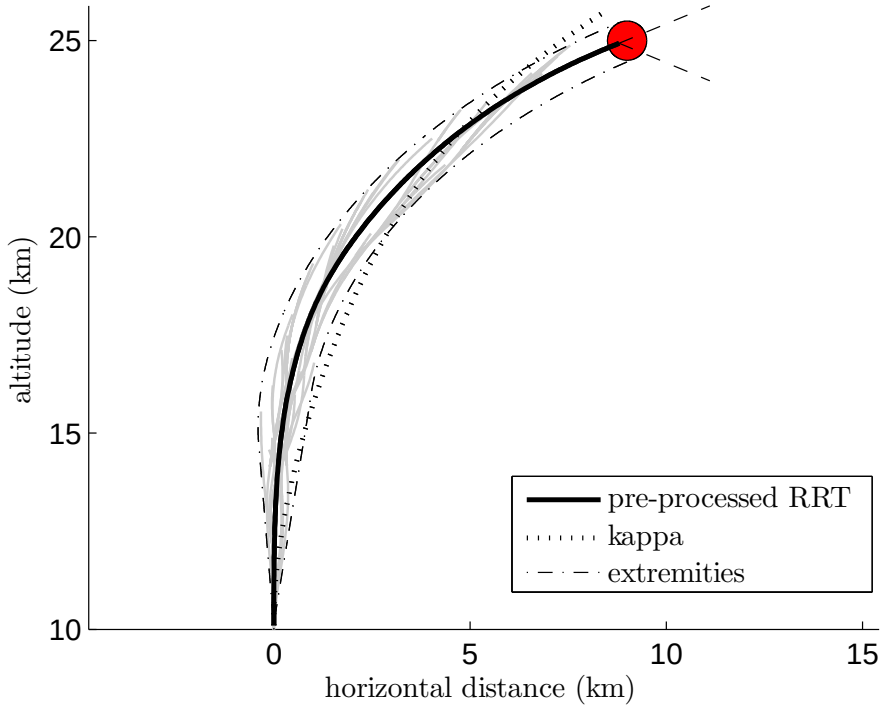


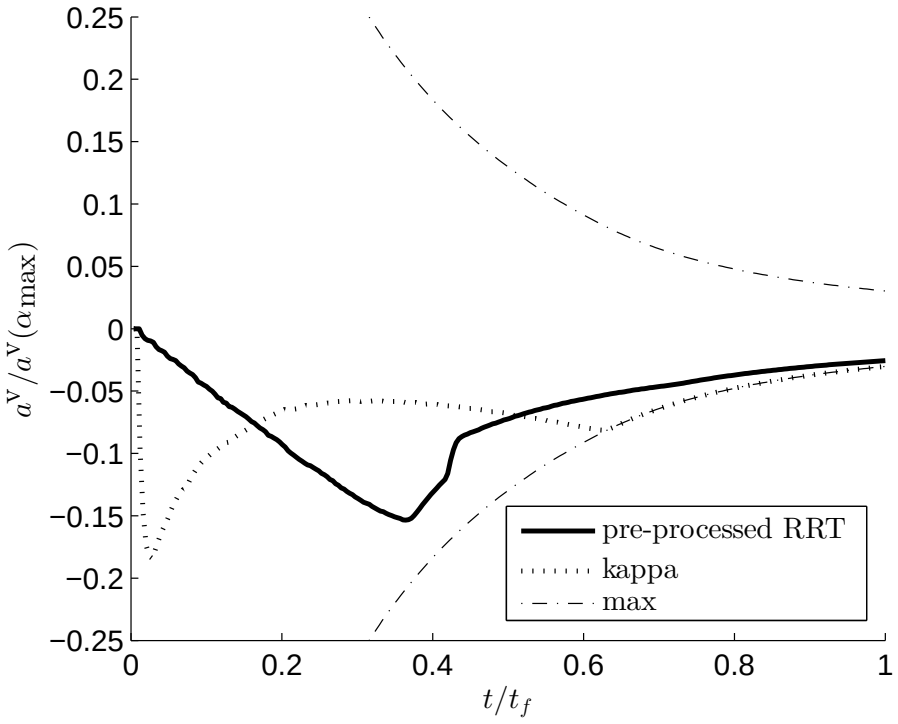
Figure 6.13: Scenario 3: Simulation result using $\text{RRT}_{\text{kappa}}$

Thus, the non-linear optimization approach, such as GPOPS, can be used to find an optimal solution instead of the RRT

According to these drawbacks, it seems that the preprocessed exploration space can be used as an estimated initial condition for other non-linear optimization approaches rather than the RRT. The reduced exploration space also makes the RRT lose its main advantage, the exploration of high-dimensional problem without a need of approximations [PLM06]. Moreover, with the non-linear optimization approaches, the solution is ensured to be an optimal solution. Thus, in the next part of the thesis, the problem is reformulated. Moreover, the optimal RRT or RRT* is used to find an optimal trajectory of aerial vehicles in the slightly different approach of using sampling-based path planning algorithms.

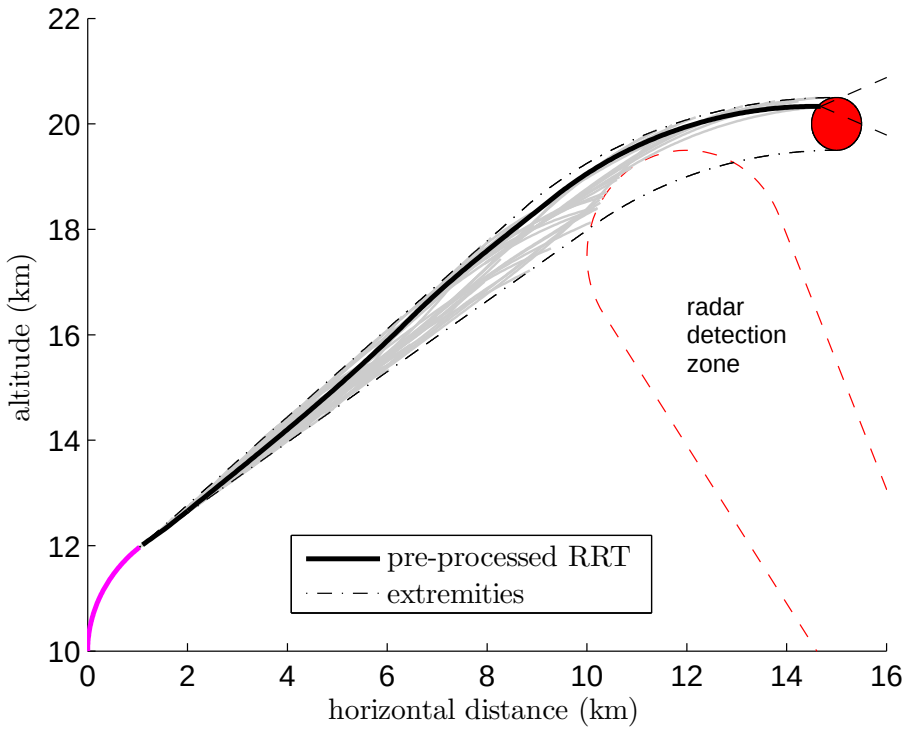


(a) Trajectories

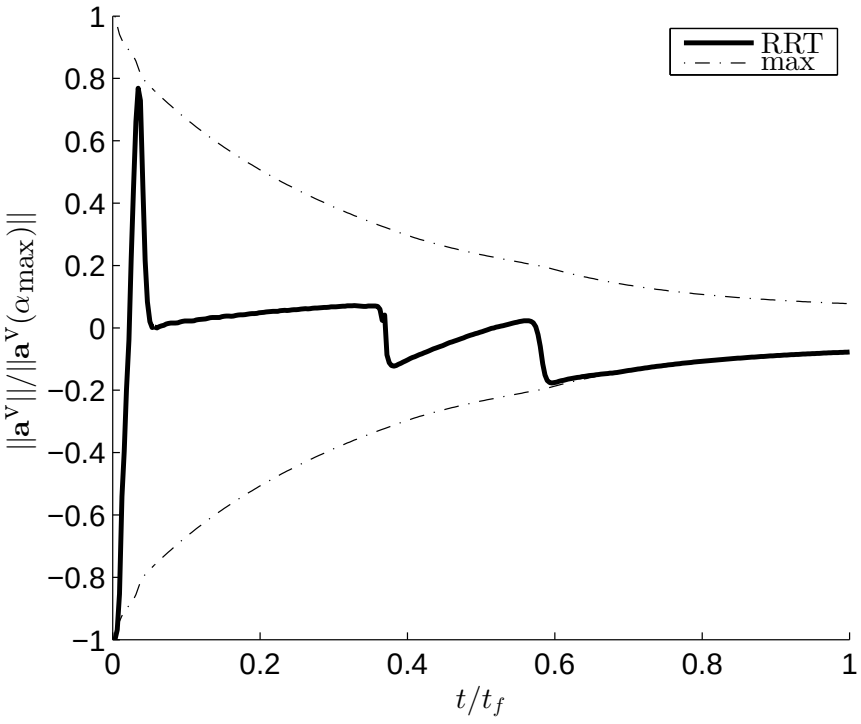


(b) Control inputs along the computed trajectories

Figure 6.14: Simulation result for scenario 1

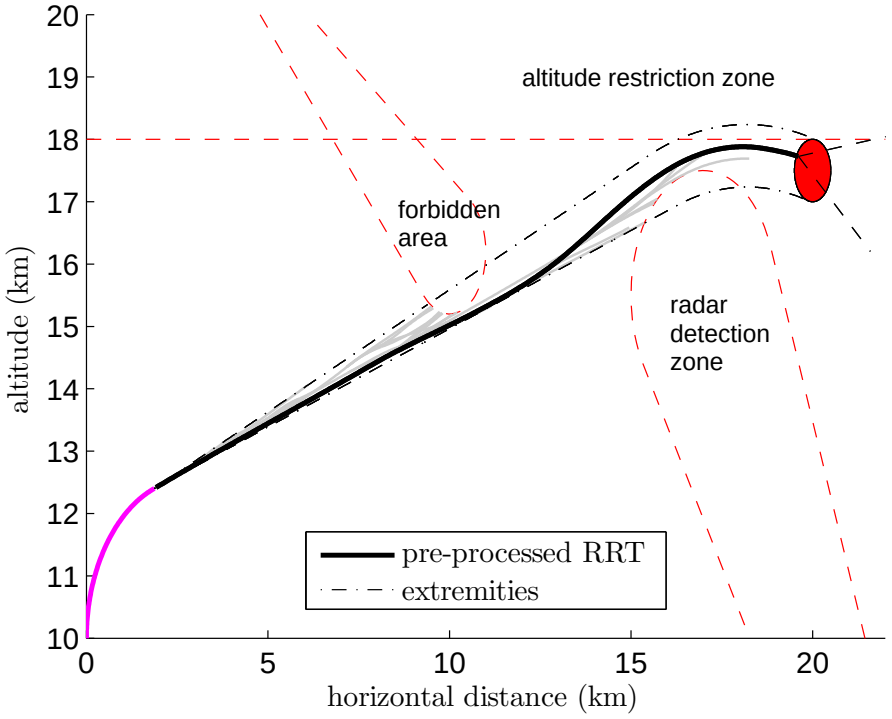


(a) Trajectories

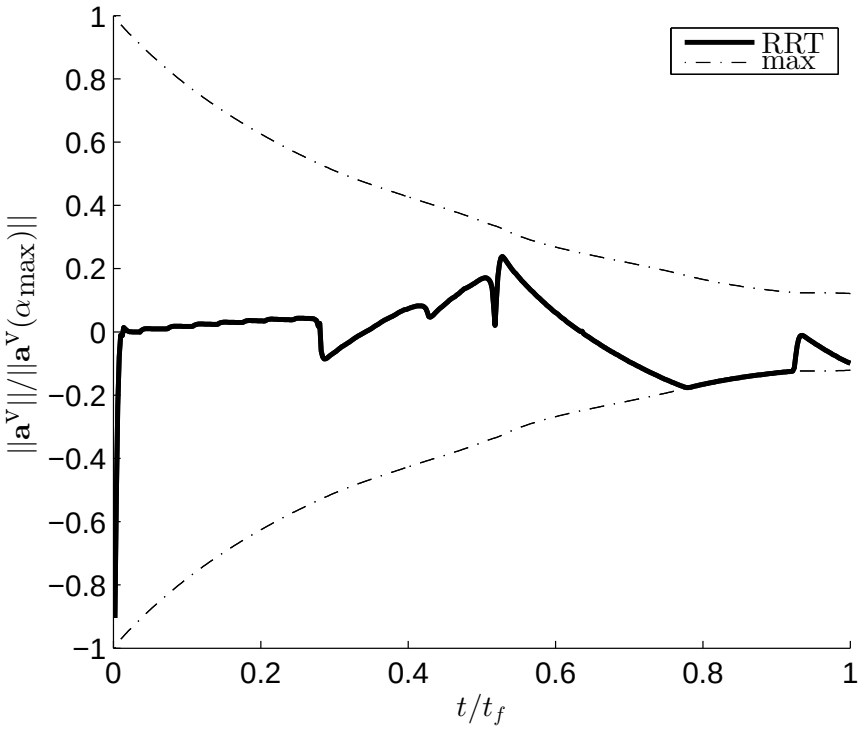


(b) Control inputs along the computed trajectories

Figure 6.15: Simulation result for scenario 2



(a) Trajectories



(b) Control inputs along the computed trajectories

Figure 6.16: Simulation result for scenario 3

Part IV

Path planning using a simplified model

7 Path planning of aerial vehicles based on RRT* algorithm

From the results of the previous chapters, the complete system model makes the problem more difficult to prove the optimality of the solution. Moreover, the preprocessing of the exploration space makes the RRT lose its charm as a method of exploration for high-dimensional problems without a need of approximation. Thus, the system model is simplified in order to prove the optimality of the obtained solution. The objective of the trajectory planning is slightly changed since the previous approach encounters some difficulties. Instead of finding a trajectory corresponding to the complete system model, the trajectory planning algorithm focuses on finding a reference trajectory using a simplified system model that is easy to follow by using path following algorithms such as MPC. Moreover, the optimality of the solution is easier to prove than the one using a complete model.

Therefore, the main contribution of this chapter is to the expansion of the framework from 2D plane to 3D plane with help of the shortest 3D Dubins' path in heterogeneous environment developed in this chapter for missile application. The computational effort is also improved by integrating the Artificial Potential Fields in a random state generation of the algorithm. Moreover, the optimal RRT or RRT* is used as a main path planner in this chapter.

In this chapter, the RRT*, the RRT algorithm with asymptotic optimality property, is used as a basis path planner to find an optimal reference trajectory between two vehicle states. The classical guidance laws are no longer used as a node expansion method. The reason is that they cannot ensure the arrival at the desired destination. Not using a complete model is also supported by this reason. This issue is a serious problem when using the

RRT* algorithm since there are some deconnections and reconnections of the exploration tree, which are explained in the description of the RRT* algorithm. Then, the application of the RRT* algorithm for aerial vehicles is explained attentively. Later, the simulation results are presented and analyzed. Finally, some conclusions are made at the end of this chapter.

7.1 Motion planning framework

The motion planning framework is developed based on the RRT* algorithm as a basis path planner. The details of the RRT* algorithm can be found in Algorithm 2 in section 7.1.1. A heuristic is added to the RRT* algorithm in order to increase the convergence rate to the optimal solution of the algorithm. The APF is used as a heuristic in random state generation of the RRT* algorithm [PHB15b]. The integration of the APF in the RRT* algorithm is explained in section 7.1.3.

7.1.1 Optimal RRT known as RRT* algorithm: An overview

Optimal RRT known as RRT* [KF10, KF11] is a RRT algorithm with the asymptotic optimality property, *i.e.* almost-sure convergence to an optimal solution, along with probabilistic completeness guarantees. The RRT* algorithm achieves the asymptotic optimality absent from the RRT without incurring substantial computational overhead. Thus, the RRT* provides substantial benefits, especially for real-time applications.

The RRT* is like the RRT that it can find a feasible path quickly. Moreover, the quality of the path is improved toward the optimal solution over the remaining time before the path execution is complete. Since most robotic systems usually spend more time to execute trajectories than to plan them, this property is very advantageous. The principle of the RRT* as a path planner is described in Algorithm 2.

Let G be the exploration tree, V be the set of vertices of the tree, E be the set of connecting edges of the tree, $\text{cost}(\{(\mathbf{x}_1, \mathbf{x}_2)\})$ be the minimal cost from $\mathbf{x}_1$ to $\mathbf{x}_2$ according to the specified criterion J. Let $\text{cost}(\mathbf{x})$ also be the total cost-to-go to $\mathbf{x}$, that is $\text{cost}(\mathbf{x})$=$\text{cost}(\{(\mathbf{x}_{\text{init}}, \mathbf{x})\})$.

First, the initial state $\mathbf{x}_{\text{init}}$ is added to the tree G. Then, a state $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$ is generated randomly using the uniform distribution. The `nearest_neighbor` function searches the tree G for the nearest vertex to $\mathbf{x}_{\text{rand}}$ according to a user-defined metric d. This state is called $\mathbf{x}_{\text{nearest}}$. In `steer` function, a control input $\mathbf{u}^*$ is selected according to the specified criterion, *i.e.* such that $J\left(t_{\text{nearest}}, t_{\text{rand}}, \mathbf{u}^*\right)$ is minimized.

Algorithm 2 RRT* path planner

Function : build_rrt*(in : $K \in \mathbb{N}$, $\mathbf{x}_{\text{init}} \in \mathbb{X}_{\text{free}}$, $\mathbb{X}_{\text{goal}} \subset \mathbb{X}_{\text{free}}$, $\Delta t \in \mathbb{R}^+$, out : G)

$G \leftarrow \mathbf{x}_{\text{init}}$
$\text{cost}(\mathbf{x}_{\text{init}}) \leftarrow 0$
$i = 0$
repeat
    $\mathbf{x}_{\text{rand}} \leftarrow \text{random\_state}(\mathbb{X}_{\text{free}})$
    $\mathbf{x}_{\text{new}} \leftarrow \text{rrt*\_extend}(G, \mathbf{x}_{\text{rand}})$
until $i++ > K$
return G

Function : rrt*_extend(in : G, $\mathbf{x}_{\text{rand}}$, out : $\mathbf{x}_{\text{new}}$)

$V \leftarrow G.\text{Node}$
$E \leftarrow G.\text{Edge}$
$\mathbf{x}_{\text{nearest}} \leftarrow \text{nearest\_neighbor}(G, \mathbf{x}_{\text{rand}})$
$(\mathbf{x}_{\text{new}}, \mathbf{u}^*) \leftarrow \text{steer}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{rand}})$
if $\text{collision\_free\_path}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$ **then**
    $V \leftarrow V \cup \{\mathbf{x}_{\text{new}}\}$
    $\text{cost}(\mathbf{x}_{\text{new}}) \leftarrow \text{cost}(\mathbf{x}_{\text{nearest}}) + \text{cost}(\{(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})\})$
    $\mathbb{X}_{\text{near}} \leftarrow \text{near\_vertices\_parents}(G, \mathbf{x}_{\text{new}})$
    $E \leftarrow \text{best\_edge}(E, \mathbb{X}_{\text{near}}, \mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$
    $\mathbb{X}_{\text{near}} \leftarrow \text{near\_vertices\_children}(\mathbf{x}_{\text{new}}, G)$
    $E \leftarrow \text{rewire}(E, \mathbb{X}_{\text{near}}, \mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$
end if
$G = (V, E)$
return $\mathbf{x}_{\text{new}}$

Function : best_edge(in : E, $\mathbb{X}_{\text{near}}$, $\mathbf{x}_{\text{min}}$, $\mathbf{x}_{\text{new}}$ out : E)

for all $\mathbf{x}_{\text{near}} \in \mathbb{X}_{\text{near}} \setminus \{\mathbf{x}_{\text{min}}\}$ **do**
    $(\mathbf{x}_{\text{new}}, \mathbf{u}^*) \leftarrow \text{steer}(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})$
    if $\text{collision\_free\_path}(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})$ and $\text{cost}(\mathbf{x}_{\text{new}}) > \text{cost}(\mathbf{x}_{\text{near}}) + \text{cost}(\{(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})\})$ **then**
        $\text{cost}(\mathbf{x}_{\text{new}}) \leftarrow \text{cost}(\mathbf{x}_{\text{near}}) + \text{cost}(\{(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})\})$
        $\mathbf{x}_{\text{min}} \leftarrow \mathbf{x}_{\text{near}}$
    end if
end for
$E \leftarrow E \cup \{(\mathbf{x}_{\text{min}}, \mathbf{x}_{\text{new}})\}$
return E

Function : rewire(in : E, $\mathbb{X}_{\text{near}}$, $\mathbf{x}_{\text{nearest}}$, $\mathbf{x}_{\text{new}}$ out : E)

for all $\mathbf{x}_{\text{near}} \in \mathbb{X}_{\text{near}} \setminus \{\mathbf{x}_{\text{nearest}}\}$ **do**
    $(\mathbf{x}_{\text{near}}, \mathbf{u}^*) \leftarrow \text{steer}(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})$
    if $\text{collision\_free\_path}(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})$ and $\text{cost}(\mathbf{x}_{\text{near}}) > \text{cost}(\mathbf{x}_{\text{new}}) + \text{cost}(\{(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})\})$ **then**
        $\mathbf{x}_{\text{parent}} \leftarrow \text{parent}(\mathbf{x}_{\text{near}})$
        $E \leftarrow E \backslash \{(\mathbf{x}_{\text{parent}}, \mathbf{x}_{\text{near}})\}$
        $E \leftarrow E \cup \{(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})\}$
        $\text{cost}(\mathbf{x}_{\text{near}}) \leftarrow \text{cost}(\mathbf{x}_{\text{new}}) + \text{cost}(\{(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})\})$
    end if
end for
return E

Then, the system model is integrated from t_{nearest} to t_{rand}, to find a new state $\mathbf{x}_{\text{new}}$, that is

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{nearest}} + \int_{t_{\text{nearest}}}^{t_{\text{rand}}} \mathbf{f}(\mathbf{x}, \mathbf{u}^*)\, dt.$$

A collision test (`collision_free_path` function) is performed: if $\mathbf{x}_{\text{new}}$ and the path between $\mathbf{x}_{\text{nearest}}$ and $\mathbf{x}_{\text{new}}$ lie in $\mathbb{X}_{\text{free}}$ then $\mathbf{x}_{\text{new}}$ is added in V.

Next, the RRT* algorithm tries to find a better parent for $\mathbf{x}_{\text{new}}$, that is a parent providing a lower cost to $\mathbf{x}_{\text{new}}$ than $\mathbf{x}_{\text{nearest}}$. The `near_vertices_parents` function in line 16 will search the tree G for a set of other potential parents in a neighborhood $\mathbb{X}_{\text{near}} \subset V$ of $\mathbf{x}_{\text{new}}$. The state $\mathbf{x}_{\text{min}} \in \mathbb{X}_{\text{near}} \cup \{\mathbf{x}_{\text{nearest}}\}$ that is collision-free and minimizes the cost to $\mathbf{x}_{\text{new}}$ is chosen to be its new parent. Therefore, the connecting edge from $\mathbf{x}_{\text{min}}$ to $\mathbf{x}_{\text{new}}$ is added in E.
Afterward, the `near_vertices_children` function in line 18 will search the tree G for a set of potential children in a neighborhood $\mathbb{X}_{\text{near}} \subset V$ of $\mathbf{x}_{\text{new}}$. For each $\mathbf{x}_{\text{near}} \in \mathbb{X}_{\text{near}}$, if the cost to $\mathbf{x}_{\text{near}}$ passing by $\mathbf{x}_{\text{new}}$ is better than cost($\mathbf{x}_{\text{near}}$) and the path is collision-free, then the `rewire` function will replace the existing connecting edge by the connecting edge from $\mathbf{x}_{\text{new}}$ to $\mathbf{x}_{\text{near}}$.

These steps are repeated until the algorithm reaches K iterations. Thus, the RRT* algorithm will improve the optimality of the solution over time even after the first solution is found.

7.1.2 Important components

The RRT* algorithm inherits the properties of the RRT algorithm. Thus, most of the important elements of the RRT* algorithm are the ones of the RRT algorithm:

- Exploration space
- Random state generation
- Metric
- Node expansion methods
- Near vertices search
- Collision tests

The explication of the mentioned elements can be found in section 5.1.2. In this section, only the elements that have additional concerns and the newly introduced element are explained.

7.1.2.1 Node expansion methods

Node expansion methods are used to move the vehicle from one state to another state. In order to expand the tree, a control input $\mathbf{u}^*$ used to expand the tree from $\mathbf{x}_{\text{near}}$ to $\mathbf{x}_{\text{rand}}$ is determined. The control input can be computed randomly or using a specific criterion such as some guidance control laws. The system model, environment model, and other constraints are integrated until the vehicle reaches $\mathbf{x}_{\text{rand}}$, *i.e.* from t_{near} to t_{rand}, in steer function.

In the RRT* algorithm, node expansion methods are not only used in expanding the tree but also used to reconnect the tree with the existing vertices. Thus, it is highly recommend to use the control laws that can move the vehicle from one state to another state perfectly or mostly perfect in order to avoid the extra computational cost of recalculating the entire tree.

7.1.2.2 Near vertices search

The near vertices search is introduced in the RRT* algorithm to improve the quality of the obtained solution. The asymptotic optimality property of the RRT* algorithm comes from this procedure. The near vertices search can be considered as a generalization of the nearest neighbor search. While the nearest neighbor search returns the closest vertex to the interested state $\mathbf{x}$, the near vertices search return a collection of vertices close to $\mathbf{x}$.

In the RRT* algorithm described in [KF10, KF11], the `near_vertices` function uses the same metric function as `nearest_neighbor` function to determine the neighborhood $\mathbb{X}_{\text{near}}$ of the state $\mathbf{x}$. There are several ways to define the near vertices search. In the original RRT* algorithm [KF10, KF11], the near vertices search of $\mathbf{x}$ is defined to be a set of all vertices within the closed ball of radius r_n centered at $\mathbf{x}$ where r_n decreases with number of iterations. In this thesis, the k-nearest search is used for the near vertices search instead of using a closed ball of radius r_n. Please note that r_n and k have to be large enough to contains all possible near vertices (see section 7.2.2).

The original `near_vertices` function is divided into 2 functions in this thesis according to their purpose: `near_vertices_parents` function in line 16, and `near_vertices_children` function in line 18. The first one searches for the k-nearest vertices to arrive at $\mathbf{x}_{\text{new}}$ while the latter one searches for the k-nearest vertices from $\mathbf{x}_{\text{new}}$ to other vertices. Note that in case of using the Euclidean distance as the metric, both functions are the same and the latter is not required in the algorithm.

7.1.3 State generation using Artificial Potential Fields or APF

The random state $\mathbf{x}_{\text{rand}}$ is generally generated by a uniform distribution in such a way that $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$. In this paper, a biased random state generation using APF [AH83, Kha85] is introduced.

APF is a reactive approach where trajectories are not generated explicitly. Instead, the environments generate some forces leading the vehicle to the destination. However, problems such as local minima and oscillatory movement can make the goal non-reachable. The APF can be used to direct the randomly generated state to the goal, *i.e.* the orientation of the random state is generated using the APF. By combining with the RRT*, the disadvantages of the APF can be solved by the randomness of the RRT*, *i.e.* the vehicle leaves the local minima by trying to go to random states [BL90, BL91]. At the same time, it is expected that the APF also increases the rate of convergence to a solution of the RRT*.

An artificial vector $\mathbf{f}_{\text{APF}}$ used to direct the vehicle is induced by an artificial potential $U \in \mathbb{R}$, *i.e.* $\mathbf{f}_{\text{APF}} = -\nabla U \in \mathbb{R}^3$. The potentials can be defined in several ways according to the characteristics of the mathematical functions. In this paper, the following artificial potentials and vectors are used.

- The repulsive potential [Kha85] is usually used around an obstacle in order for the vehicle to avoid it. It can also be used to move the vehicle away from the starting state. The repulsive potential and force can be expressed as

$$U_{\text{init}} = \frac{1}{2} K_{\text{init}} \frac{1}{||\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{init}}||_2^2}, \tag{7.1}$$

$$\mathbf{f}_{\text{init}} = K_{\text{init}} \frac{\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{init}}}{||\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{init}}||_2^4}, \tag{7.2}$$

  where U_{init}, $\mathbf{f}_{\text{init}}$ are the repulsive potential and vector field centered at $\boldsymbol{\xi}_{\text{init}}$ and K_{init} is a constant.

- The attractive potential, opposing to the repulsive potential, is used to direct the vehicle to the desired destination. The attractive potential and force can be expressed as

$$U_{\text{goal}} = -\frac{1}{2} K_{\text{goal}} \frac{1}{||\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{goal}}||_2^2}, \tag{7.3}$$

$$\mathbf{f}_{\text{goal}} = -K_{\text{goal}} \frac{\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{goal}}}{||\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{goal}}||_2^4}, \tag{7.4}$$

  where U_{goal}, $\mathbf{f}_{\text{goal}}$ are the attractive potential and vector fields centered at $\boldsymbol{\xi}_{\text{goal}}$ and K_{goal} is a constant.

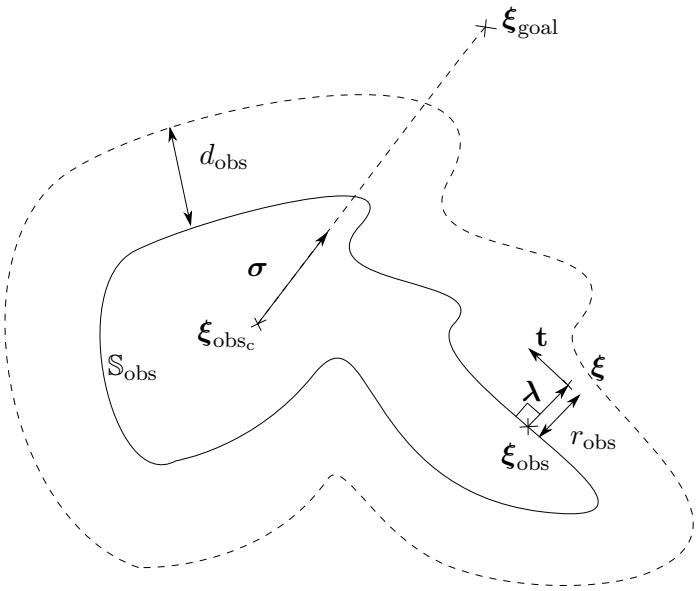


Figure 7.1: Example of rotational vector field around the obstacle

- The rotational vector field [Fal14] is used instead of the repulsive potential to guide the vehicle around the obstacles in order to avoid the local minimum problem. In figure 7.1, let $\boldsymbol{\xi}_{\text{obs}_c}$ denote the center of gravity of an obstacle, $\mathbb{S}_{\text{obs}}$ denote the set of the obstacle surface, d_{obs} denote a maximum distance of influence from the obstacle, r_{obs} denote the shortest distance of the vehicle from the obstacle, and $\mathbf{t}$, $\boldsymbol{\sigma}$, $\boldsymbol{\lambda}$ are unit vectors defined as

$$\boldsymbol{\sigma} = \frac{\boldsymbol{\xi}_{\text{goal}} - \boldsymbol{\xi}_{\text{obs}_c}}{||\boldsymbol{\xi}_{\text{goal}} - \boldsymbol{\xi}_{\text{obs}_c}||},$$
$$\boldsymbol{\lambda} = \frac{\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{obs}}}{||\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{obs}}||},$$
$$\mathbf{t} = \frac{\boldsymbol{\lambda} \times (\boldsymbol{\sigma} \times \boldsymbol{\lambda})}{||\boldsymbol{\lambda} \times (\boldsymbol{\sigma} \times \boldsymbol{\lambda})||}.$$

The vector field function can be defined as

$$\mathbf{f}_{\text{obs}} = K_{\text{obs}}(\frac{d_{\text{obs}} - r_{\text{obs}}}{r_{\text{obs}}})\mathbf{t}, \qquad r_{\text{obs}} \leqslant d_{\text{obs}} \tag{7.5}$$
$$\mathbf{f}_{\text{obs}} = 0, \qquad r_{\text{obs}} > d_{\text{obs}} \tag{7.6}$$

where $\boldsymbol{\xi}_{\text{obs}} \in \mathbb{S}_{\text{obs}}$ is the nearest point of an obstacle to $\boldsymbol{\xi}$, and K_{obs} is a constant. Note that for a spherical obstacle, $\boldsymbol{\lambda} = \frac{\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{obs}_c}}{||\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{obs}_c}||}$

Thus, the summation of artificial vector fields expressed as

$$\mathbf{f}_{\text{APF}} = \sum_{i=1}^{n} \mathbf{f}_{\text{i}} = \mathbf{f}_{\text{goal}} + \mathbf{f}_{\text{init}} + \mathbf{f}_{\text{obs}}, \tag{7.7}$$

is used as a basis to generate the orientation $(\gamma_{\text{rand}}, \chi_{\text{rand}})$ of a random state $\mathbf{x}_{\text{rand}}$.

Remark 3 *Some more adaptive artificial potential functions or navigation functions can be found in [FK11, FKA12, FK12] which will be useful for future works in practical and experimental studies. In this thesis, only simple artificial potential functions are only to demonstrate the concept of this framework.*

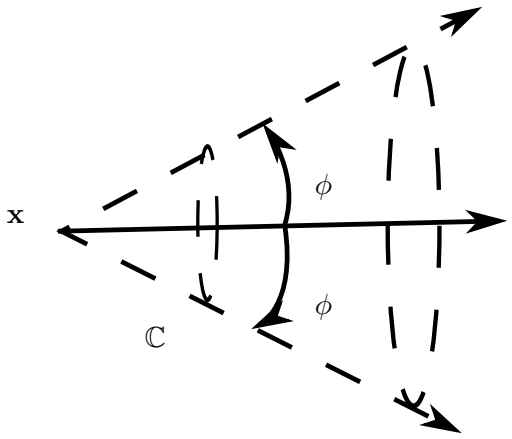


Figure 7.2: Definition of convex cone $\mathbb{C}(\mathbf{x}, \phi)$

Thus, the `random_state` function generates a random state $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$ by using the uniform distribution as in the original algorithm. The orientation of the randomly generated state is biased toward the destination using the direction of $\mathbf{f}_{\text{APF}}$ with a given margin, that is $\mathbf{x}_{\text{rand}}$ is chosen in the convex cone illustrated in figure 7.2 pointing toward $\mathbf{f}_{\text{APF}}$ with apex $\boldsymbol{\xi}_{\text{rand}}$ and apex angle $2\phi_{\text{APF}}$. Moreover, a bias toward the goal, used in RRT-*goalbias* [LK01], is also used to increase the rate of convergence to an optimal solution. It consists in choosing $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$ with a probability p.

This framework biased using APF is expected to be able to find a feasible and optimal solution with less number of iterations than the existing RRT* algorithm. This will be demonstrated in simulations in section 7.4.2 for the application considered in section 7.3. The properties of this framework are analyzed in the next section.

7.2 Property Analysis

7.2.1 Probabilistic completeness

The algorithm is said to be probabilistic complete if it finds a feasible path with probability approaching one as the number of iterations approaches infinity [KF10]. Since the RRT* algorithm returns a connected tree G including $\mathbf{x}_{\text{init}}$ on $\mathbb{X}_{\text{free}}$ as the RRT algorithm does, there always exists a collision-free path starting from $\mathbf{x}_{\text{init}}$ to any vertex in the tree G. Thus, if there exists a feasible path to $\mathbb{X}_{\text{goal}}$, then this framework is automatically probabilistically complete.

7.2.2 Asymptotic optimality

In this section, a set of sufficient conditions is proposed to guarantee the convergence of the RRT* algorithm to an optimal solution for the aerial vehicle path planning, *i.e.* the probability to find an optimal solution approaches one when the number of iterations approaches infinity.

The convergence to an optimal solution is guaranteed if two conditions are met. The first condition is that there exists an optimal path with sufficient free space in its neighborhood. The second condition is that the system is local controllable, *i.e.* the system can be moved around in its entire neighborhood using only admissible control inputs. With these two conditions, asymptotic optimality of the RRT* algorithm is guaranteed. The aerial system considered in this paper is assumed to be controllable in its domain of use. Therefore, it is only required to assume that there exists an optimal trajectory with free space around it.

A trajectory $\mathcal{X}_\epsilon$ is said ϵ-collision-free, for $\epsilon \in (0, \bar{\epsilon})$ where $\bar{\epsilon} \in \mathbb{R}_{>0}$ if all states along $\mathcal{X}\epsilon$ are at least ϵ away from the obstacles. Assume that there exist an optimal feasible trajectory $\mathcal{X}^*$ and a function $\mathcal{X}\epsilon$ such that $\mathcal{X}\epsilon$ is an ϵ-collision-free trajectory for all $\epsilon > 0$ that $\mathcal{X}\epsilon(0) = \mathbf{x}_{\text{init}}$ and $\mathcal{X}\epsilon(t_{\text{f}}) \in \mathbb{X}_{\text{goal}}$. Moreover, $\mathcal{X}\epsilon$ converges to $\mathcal{X}^*$ when ϵ approaches zero. Based on these assumptions and local controllability of the system, it is shown in [KF10] that the RRT* algorithm can execute the reconnection process around $\mathcal{X}\epsilon$ infinitely often and then, approaches $\mathcal{X}^*$ asymptotically.

However, this result is true for all ϵ provided that the area of research around neighborhood in the `near_vertices` function (see section 7.1.2.2) is large enough. Indeed, if the area of research contains a ball of volume $\gamma_{\text{RRT}^*_1} \ln(n)/n$, $\gamma_{\text{RRT}^*_1}$ needs to be large enough to ensure asymptotic optimality of the algorithm. It is proven in [KF10] the asymptotic optimality is ensured if $\gamma_{\text{RRT}^*_1} > 2(1 - 1/d)\left(\frac{\mu(\mathbb{X}_{\text{free}})}{\zeta_d}\right)$, where d is the dimension of the exploration space $\mathbf{x}$, $\mu(\mathbb{X}_{\text{free}})$ is the volume of $\mathbb{X}_{\text{free}}$, and ζ_d is the volume of the unit ball in d-dimensional euclidean space.

Since the uniform distribution is used for the state generation, the expected number of vertices contained in a ball of volume $\gamma_{\mathrm{RRT}^*_1} \ln(n)/n$ after iteration n is $\gamma_{\mathrm{RRT}^*_1} \ln(n)/\mu(\mathbb{X}_{\mathrm{free}})$. The proof is straightforward since the number of vertices contained in the ball follows a binomial distribution with parameters n and $\gamma_{\mathrm{RRT}^*_1} \ln(n)/n\mu(\mathbb{X}_{\mathrm{free}})$. Thus, if k-nearest search is used in the same manner, k needs to verify $k > \gamma_{\mathrm{RRT}^*_2} \ln(n)$. The asymptotic optimality is ensured if $\gamma_{\mathrm{RRT}^*_2} > 2^{d+1}e(1-1/d)$ as proven in [KF10]. Note that $\gamma_{\mathrm{RRT}^*_2}$ depends only on d not the problem instance, unlike the original RRT*. Thus, $\gamma_{\mathrm{RRT}^*_2} = 2^{d+1}e$ is a valid choice for all problem instances.

If all these assumptions hold, the RRT* algorithm can find an optimal trajectory almost surely. A thorough proof can be found in [KF11].

Remark 4 *The APF bias that is used in the framework does not contradict this assumption since a given margin ϕ_{APF} described in section 7.1.3 is used for the uniform state generation. Indeed, to ensure free space in the neighborhood, it is only required that any states $\mathbf{x}$ on the optimal path are in the convex cone $\mathbb{C}(\mathbf{x}_{APF}, \phi_{APF})$ (see figure 7.2 for a definition of $\mathbb{C}$), where $\mathbf{x}_{APF}$ is a state on the vector field at position $\boldsymbol{\xi}$ ($\boldsymbol{\xi}_{APF} = \boldsymbol{\xi}$).*

7.3 Application for a hypersonic aerial vehicle

The performance of the framework is demonstrated using the Dubins' path in a heterogeneous environment as the user-defined metric d [PHB15a]. The framework is simulated on 2D and 3D applications for a hypersonic aerial vehicle, namely an interceptor missile. In a 2D application, only the RRT* algorithm without heuristic is used. It is to verify the capability of the original RRT* algorithm in find an optimal solution. After verifying the capability of the RRT* algorithm, the framework is then fully used in a 3D application. In the following sections, the environmental and system modeling are described. Then, the problem statement of the case study is described. The calculation of the 3D path based on a simplified heterogeneous environment and a Dubins-like vehicle is detailed. Then, how to use Dubins' path for the metric is explained.

7.3.1 Environment modeling

The environment is considered heterogeneous in a 2-dimensional vertical plane because of variation of air density ρ, decreasing exponentially with altitude. The environment model can be expressed as:

$$\rho = \rho_0 e^{-z/z_r} \tag{7.8}$$

where ρ_0 is the air density at standard atmosphere at the sea level and z_r is the reference altitude.

7.3.2 System modeling

A simplified hypersonic Dubins-like vehicle is used to simulate results. In this paper, an unpowered interceptor missile during midcourse phase is chosen. For the hypersonic missile, wind speed has so few effect to the system that a zero wind assumption is applied. Then, the translational velocity $\mathbf{v}$ is assumed to coincide with the apparent velocity. Besides, a hypersonic missile is studied. Thus, the gravity can be neglected, *i.e.* $g = 0$ in system (2.8), which is a strong hypothesis that is only valid for missile-like aircraft flying with a high Mach number in a short distance. The propulsion of the vehicle is considered coincide with the axis of the vehicle, *i.e.* $T_2 = T_3 = 0$ in system (2.8). No external perturbation force is considered in the simplified model neither, *i.e.* $F_{p_1} = F_{p_2} = F_{p_3} = 0$ in system (2.8). Moreover, the drag can be ignored since the objective is to find the shortest path between two states, *i.e.* the path of minimum length. Thus, the dynamics of the velocity does not need to be considered. Moreover, the aerodynamic coefficient C_{L_2} is equal to C_{L_3} because an axisymmetric missile is considered here (cf. system (2.8)).

7.3.2.1 3D system modeling

By simplifying system (2.8) according to the mentioned conditions, the only external force that effects the motion of the vehicle is the lift force, *i.e.* $\mathbf{f}_{\mathrm{L}} = \frac{1}{2m}\rho(z)SC_L(\alpha)v^2\mathbf{e}_1^{\mathrm{Y}}$. Therefore, by applying a change of variables from t to curvilinear abscissa $s(t) = \int_0^t v(u)\,du$, the equation of motions of a Dubins-like aerial vehicle can be written as

$$\begin{cases} x' = \dfrac{dx}{ds} = \cos\gamma\cos\chi, \\ y' = \dfrac{dy}{ds} = \cos\gamma\sin\chi, \\ z' = \dfrac{dz}{ds} = \sin\gamma, \\ \gamma' = \dfrac{d\gamma}{ds} = \dfrac{1}{2m}\rho(z)SC_L(\alpha)\cos\alpha = \dfrac{1}{2m}\rho(z)SC_{L_{\max}}\dfrac{C_L(\alpha)}{C_{L_{\max}}}\cos\alpha = c(z)\mu, \\ \chi' = \dfrac{d\chi}{ds} = \dfrac{1}{2m}\rho(z)SC_L(\alpha)\dfrac{\sin\alpha}{\cos\gamma} = \dfrac{1}{2m}\rho(z)SC_{L_{\max}}\dfrac{C_L(\alpha)}{C_{L_{\max}}}\dfrac{\sin\alpha}{\cos\gamma} = c(z)\dfrac{\eta}{\cos\gamma}, \end{cases} \tag{7.9}$$

where $C_L(\alpha)$ is the lift coefficient at α, $C_{L_{\max}}$ is the maximum lift coefficient at $\alpha_{\max}$, $\mu = \frac{C_L(\alpha)}{C_{L_{\max}}}\cos\alpha$, $\eta = \frac{C_L(\alpha)}{C_{L_{\max}}}\sin\alpha$ are the normalized control inputs bounded by condition $\sqrt{\mu^2+\eta^2} \leqslant 1$, $\rho(z)$ is the air density, and $c(z) = \frac{1}{2m}\rho(z)SC_{L_{\max}}$ is the maximum path curvature of the vehicle. Note that μ, η are directly related the angle of attack and the sideslip angle.

7.3.2.2 2D system modeling

The equations of motion (7.9) can be deduced in 2D as

$$\begin{cases} x' = \dfrac{dx}{ds} = \cos\gamma, \\ z' = \dfrac{dz}{ds} = \sin\gamma, \\ \gamma' = \dfrac{d\gamma}{ds} = c(z)u, \quad |u| \leqslant 1. \end{cases} \tag{7.10}$$

where $u \in \mathbb{R}$ is the control input ($u \in [-1, 1]$) and $\gamma = \chi$ is the orientation of the vehicle in 2D plane.

As a consequence of the environmental model (7.8), the path curvature can be written in the same way as

$$c(z) = c_0 e^{-z/z_r}. \tag{7.11}$$

where c_0 is the maximum curvature at sea level. At this point, it can be noticed that the maneuverability of the vehicle is exponentially decreasing with altitude since $c(z)$ is exponentially decreasing. Thus, it is a challenging problem to control such a vehicle at high altitudes.

Remark 5 *Our objective is to find a reference trajectory for such a vehicle. It is meant to follow the reference trajectory with its own controller. Therefore, elements related to the controller [DSF00] such as model uncertainties and measurement errors are not considered here.*

7.3.3 Problem formulation

7.3.3.1 System notations for 3D problems

In the 3D application, let $\mathbf{x} = \left(\boldsymbol{\xi}^\top, \gamma, \chi\right)^\top \in \mathbb{X} = \mathbb{R}^5$ be the measurable state of the system and $\mathbf{u}^* = (\mu, \eta)^\top \in \mathbb{U}$ be an admissible control input. The set of admissible control inputs is defined as

$$\mathbb{U} = \{\mathbf{u}^* \in \mathbb{R}^2 : ||\mathbf{u}^*|| \leqslant 1\} \tag{7.12}$$

The differential system (7.9) is rewritten as

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{u}^*). \tag{7.13}$$

where $\mathbf{f}(\cdot)$ is defined in section 7.3.2 equation (7.9).

7.3.3.2 System notations for 2D problems

While in the 2D application, let $\mathbf{x}(t) = (\boldsymbol{\xi}^\top, \gamma)^\top \in \mathbb{X} = \mathbb{R}^3$ be the measurable state of the system, $\mathbf{u}^* \in \mathbb{U} = [-1, 1]$ be the admissible control input and consider the differential system

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{u}^*), \tag{7.14}$$

where $\mathbf{f}(\cdot)$ is defined in section 7.3.2 equation (7.10).

7.3.3.3 Common notations

The obstacle region $\mathbb{X}_{\text{obs}}$ is defined by the exploration space occupied by no-fly zones such as city areas and radar detection zones and the set of admissible states is the remaining of the exploration space, *i.e.* $\mathbb{X}_{\text{free}} = \mathbb{X} \setminus \mathbb{X}_{\text{obs}}$.

The path planning starts at the initial state $\mathbf{x}_{\text{init}} \in \mathbb{X}_{\text{free}}$. The destination of the path planning is given by a rendez-vous set $\mathbb{X}_{\text{goal}} \subset \mathbb{X}_{\text{free}}$.

7.3.3.4 Objective

The objective of path planning algorithm is to find a collision-free trajectory $\mathcal{X}(s) : [0, s_f] \to \mathbb{X}_{\text{free}}$ with $\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{u}^*)$, that starts at $\mathbf{x}_{\text{init}}$, reaches the goal region $\mathbb{X}_{\text{goal}}$, *i.e.* $\mathbf{x}(0) = \mathbf{x}_{\text{init}}$ and $\mathbf{x}(s_f) \in \mathbb{X}_{\text{goal}}$ and minimizes the cost function

$$J = \int_0^{s_f} ds. \tag{7.15}$$

7.3.4 RRT* configurations

7.3.4.1 Exploration space

1. Obstacle space $\mathbb{X}_{\text{obs}}$: in this chapter, only the static obstacles such as the coverage areas of the radar station, are considered. Thus, $\mathbb{X}_{\text{obs}}$ is defined as

   - 3D problems:

   $$\begin{aligned} \mathbb{P}_{\text{obs}} = \{\boldsymbol{\xi} \in \mathbb{R}^3 : \quad & \boldsymbol{\xi} \text{ is the position surrounded} \\ & \text{by bounders of obstacles}\}, \\ \mathbb{V}_{\text{obs}} = \{(\gamma, \chi)^\top \in \mathbb{R}^2\}, & \end{aligned} \tag{7.16}$$

   - 2D problems:

   $$\begin{aligned} \mathbb{P}_{\text{obs}} = \{\boldsymbol{\xi} \in \mathbb{R}^2 : \quad & \boldsymbol{\xi} \text{ is the position surrounded} \\ & \text{by bounders of obstacles}\} \\ \mathbb{V}_{\text{obs}} = \{\gamma \in \mathbb{R}\} & \end{aligned} \tag{7.17}$$

2. Collision-free exploration space $\mathbb{X}_{\text{free}} = \mathbb{X}/\mathbb{X}_{\text{obs}}$: this is the space where the RRT algorithm will try to explore to find a solution.

3. Destination set $\mathbb{X}_{\text{goal}} \in \mathbb{X}_{\text{free}}$: this space in 3D problem, which has the same illustration as in figure 7.2, is defined as

   - 3D problems:

$$\begin{aligned} \mathbb{X}_{\text{goal}} &= \mathbb{P}_{\text{goal}} \times \mathbb{V}_{\text{goal}}, \\ \mathbb{P}_{\text{goal}} &= \{\boldsymbol{\xi}_{\text{rdv}}\}, \\ \mathbb{V}_{\text{goal}} &= \{(\gamma, \chi)^{\top} \in \mathbb{R}^2 : (\gamma, \chi) \in \mathbb{C}(\mathbf{x}_{\text{rdv}}, \phi_f)\}, \end{aligned} \tag{7.18}$$

     where $\mathbf{x}_{\text{rdv}}$ is the rendez-vous state, $\mathbb{C}(\mathbf{x}_{\text{rdv}}, \phi_f)$ is the convex cone (see figure 7.2) pointing toward the orientation of the vehicle defined by γ_{rdv} and χ_{rdv} with apex $\boldsymbol{\xi}_{\text{rdv}}$ and apex angle $2\phi_f$ where ϕ_f is a maximal acceptable orientation error defined by the detection range of the embedded radar or infrared sensor.

   - 2D problems:

$$\begin{aligned} \mathbb{X}_{\text{goal}} &= \mathbb{P}_{\text{goal}} \times \mathbb{V}_{\text{goal}}, \\ \mathbb{P}_{\text{goal}} &= \{\boldsymbol{\xi} \in \mathbb{R}^2 : \boldsymbol{\xi} = \boldsymbol{\xi}_{\text{rdv}}\}, \\ \mathbb{V}_{\text{goal}} &= \{\gamma \in \mathbb{R} : \gamma \in \mathbb{C}(\mathbf{x}_{\text{rdv}}, \phi_f)\}, \end{aligned} \tag{7.19}$$

7.3.4.2 Random state generation

- 2D problems: the goal of 2D application is to show and verify the performance of the original RRT* algorithm in solving trajectory planning problems for aerial vehicles. Thus, the random state $\mathbf{x}_{\text{rand}}$ is generated such that $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{free}}$. A uniform distribution is used. Moreover, the RRT-*goalbias* [LK01] is used. It means that `random_state` function returns $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$ with a probability p and returns randomly generated state using uniform distribution with a probability $(1 - p)$.

- 3D problems: the random state is generated using the methodology shown in section 7.1.3.

7.3.4.3 Metric

The Euclidean metric is often used in such an algorithm. It finds a shortest line-of-sight distance between two positions. Moreover, the kd-tree [Ben75] can be used to enhance the robustness of the algorithm. Thus, it is fast and easy to implement. However, for a nonholonomic vehicle, the Euclidean metric is not appropriated for several reasons. The main reason is that it does not take the orientation of the vehicle into account. A suitable

metric can be, for example, based on the shortest Dubins' path used in [Dub57] even if it is not perfectly realistic. Although more complex to compute, the most interesting metric in our framework consists in considering the cost from $\mathbf{x}_{\text{near}}$ to $\mathbf{x}_{\text{rand}}$, *i.e.* $\text{cost}(\{(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{rand}})\})$.

For the application considered in this chapter, a user-defined metric d based on the shortest 2D Dubins' path in a heterogeneous environment (see Appendix D for calculation detail) is used in the 2D application. 3D Dubins' paths in a heterogeneous environment are developed and used as a metric in the 3D application (see section 7.3.5). There are two cases to consider when using Dubins' paths as the user-defined metric d:

- $\mathbf{x}_{\text{rand}} \notin \mathbb{X}_{\text{goal}}$: the shortest CSC path shown in figure 7.4 is used to calculate the nearest neighbor.
- $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$: it is more interesting to consider the shortest path between each $\mathbf{x} \in G$ and a set of goal states $\mathbb{X}_{\text{goal}}$ than considering a single $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$. Indeed, there can exist the shortest path to another element of $\mathbb{X}_{\text{goal}}$ than the shortest path to $\mathbf{x}_{\text{rand}}$. To this manner, the degenerated form (CS) of CSC path needs to be considered first. Indeed, if the shortest path to the set $\mathbb{X}_{\text{goal}}$ is a CS path, then it arrives with the arrival orientation within the arrival cone $\mathbb{C}$, the expected shortest path is this CS path. If no CS path arrives in $\mathbb{X}_{\text{goal}}$, the shortest path is necessarily a CSC path whose arrival orientation is the extremity of the arrival cone $\mathbb{C}$.

Thus, for each $\mathbf{x} \in \mathbb{G}$, the approach first consists in finding the shortest CS path to the desired final position, *i.e.* $\boldsymbol{\xi}_{\text{rand}} = \boldsymbol{\xi}_{\text{rdv}}$. If the final orientation is in the arrival cone, it is the expected solution. If not, the solution is the shortest CSC path to one of the extremities of $\mathbb{X}_{\text{goal}}$. The illustrations of these two cases are the same as shown in figures 5.3 and 5.4.

7.3.4.4 Node expansion method

In both 2D and 3D applications, node expansion method is used to move the vehicle from one state to another. In this chapter, the Dubins' path in a heterogeneous environment is also used to compute the path between two vehicle states. The control input $||\mathbf{u}^*|| = \pm 1$ is used for curves of maximum curvature C and $||\mathbf{u}^*|| = 0$ is used for straight line S. The control input $||\mathbf{u}^*||$ is applied to the system (7.9) during an integration step Δs for *epsilon* steps in 2D applications or until the arrival state is reached or until an obstacle is encountered for 3D applications. Then, a new state $\mathbf{x}_{\text{new}}$ is obtained.

7.3.4.5 Near vertices search

In this framework, the k-nearest algorithm is used to determine a set of near vertices $\mathbb{X}_{\text{near}}$ of the state $\mathbf{x}$. It selects the first k-nearest vertices according to the user-defined metric d

and returns them to the set $\mathbb{X}_{\text{near}}$. To ensure asymptotic optimality, $k(n)$ is chosen with respect the condition in section 7.2.

7.3.4.6 Collision tests

The path between $\mathbf{x}_{\text{near}}$ and $\mathbf{x}_{\text{new}}$ are verified if they lie in $\mathbb{X}_{\text{free}}$. If it is collision-free, the path between $\mathbf{x}_{\text{near}}$ and $\mathbf{x}_{\text{new}}$ is added to the tree G.

7.3.5 3-dimensional Dubins' paths in heterogeneous environment

In this section, a user-defined metric d is defined based on the shortest 3-dimensional Dubins's path [PHB15a] developed from [HG10] and [HP13].

It is shown in [Dub57, BCL91] that the shortest 2-dimensional path between two fixed states of Dubins' vehicle is composed of straight line (S) and arc of circle of minimum turning radius (C), *i.e.* CSC or CCC path. In [SL01], it is proven that CSC path and not CCC path is the shortest path if two states are sufficiently far from each other.

In [Sus95], it is also demonstrated that, for sufficiently close distance between two states in a 3-dimensional plane, the helicoidal arc can be shorter than the CSC path. Then, the shortest path is proven to be a helicoidal arc, a CSC path, a CCC path or a degenerated form of these Dubins' paths, *i.e.* S, C, CS, SC, and CC paths. Later, in [Sha07], Dubins' path in 2-dimensional plane is extended to 3-dimensional plane for multiple UAVs path planning. Suboptimal CCSC paths are used.

In this paper, a 3-dimensional length-optimal path between two given states for an aerial vehicle in a heterogeneous environment, *i.e.* variable turning radius, is considered as a user-defined metric d used in the RRT* algorithm. The CSC paths are only considered under the hypothesis of "$\mathbf{x}_0$ and $\mathbf{x}_{\text{f}}$ are sufficiently far from each other".

In the following sections, the computation of the curve C of Dubins' path in a heterogeneous environment in a particular 2-dimensional plane is described. Then, the methodology of 3-dimensional path generation is demonstrated.

7.3.5.1 Computation of the curve C in a particular 2D plane

In [HP13], the shortest Dubins' path in a heterogeneous environment in a 2-dimensional plane where the curvature of the vehicle decreases exponentially with altitude is demonstrated. Let P denote a plane with its unit normal vector $\mathbf{b}$ as shown in figure 7.3. The position (x_p, y_p, z_p) is associated with vector basis $(\mathbf{e}_1^{\text{p}}, \mathbf{e}_2^{\text{p}}, \mathbf{e}_3^{\text{p}})$, of the vehicle on the plane. It can be expressed in frame $\mathcal{I}$ in function of angles ϕ and ψ as shown in figure 7.3.

In this 2-dimensional system ($y_p = 0$), $\gamma_p = \angle(\mathbf{e}_1^{\mathrm{p}}, \mathbf{v})$ is the turning angle. In this coordinate system, the dynamics of the vehicle can be modeled as:

$$\begin{aligned} x_p' &= \frac{dx_p}{ds} = \cos\gamma_p, \\ z_p' &= \frac{dz_p}{ds} = \sin\gamma_p, \\ \gamma_p' &= \frac{d\gamma_p}{ds} = c(z_p)u_p \ \text{ where } \ u_p \in [-1, 1]. \end{aligned} \tag{7.20}$$

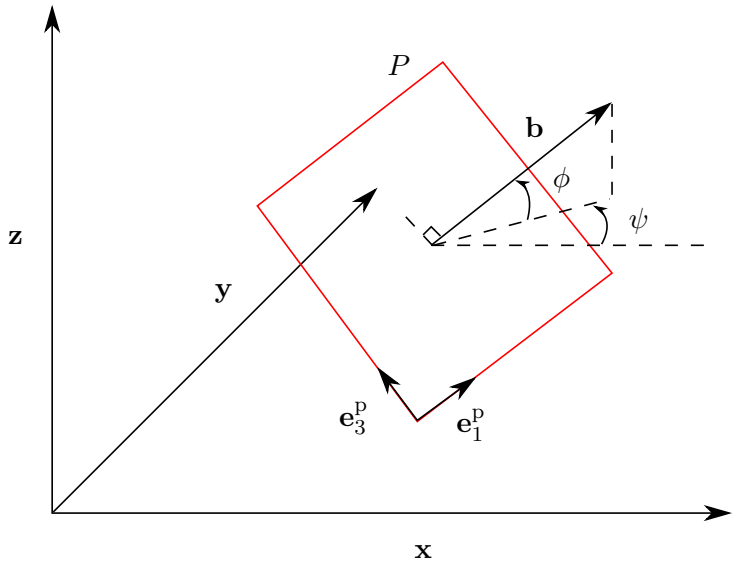


Figure 7.3: Definition of a plane P with a normal vector $\mathbf{b}$

As a consequence of the calculation on the plane P, the environment model, *i.e.* equation (7.8), on the plane P is rewritten as

$$\rho(z_p) = \rho_0 e^{-z/z_r} = \rho_0 e^{-z_p\cos\phi/z_r}. \tag{7.21}$$

Moreover, curvature equation (7.11) can be written as

$$c(z_p) = c_0 e^{-z/z_r} = c_0 e^{-z_p\cos\phi/z_r}. \tag{7.22}$$

Using the same and straightforward calculation as in Appendix D, γ_p, x_p, and z_p can be written in functions of $\zeta(s)$ as follows:

$$\begin{cases} \gamma_p(\zeta) = 2\arctan\zeta + k(s)\pi, \\ x_p(\zeta) = \dfrac{z_r}{\cos\phi}(\gamma_p(\zeta) - \gamma_{p_0}) - \dfrac{z_r}{\cos\phi}(A+B)s, \\ z_p(\zeta) = -\dfrac{z_r}{\cos\phi}\ln\left(\dfrac{1+\zeta_0^2}{A+B\zeta_0^2}\dfrac{A+B\zeta^2}{1+\zeta^2}\right), \end{cases} \tag{7.23}$$

where $k(s)$ is an integer depending on the distance s. For C_1 curve, $k(s)$ is calculated as

$$k(s) = \left\lfloor sA\sqrt{\frac{B}{A}} / \left(u_p \frac{\pi}{2} - \arctan\left(\sqrt{\frac{B}{A}} \zeta_0 \right) \right) \right\rfloor, \; u_p = \pm 1 \tag{7.24}$$

where $\lfloor . \rfloor$ is a floor division. Otherwise, $k(s) = 0$ for C_2 curve.

7.3.5.2 3D paths generation

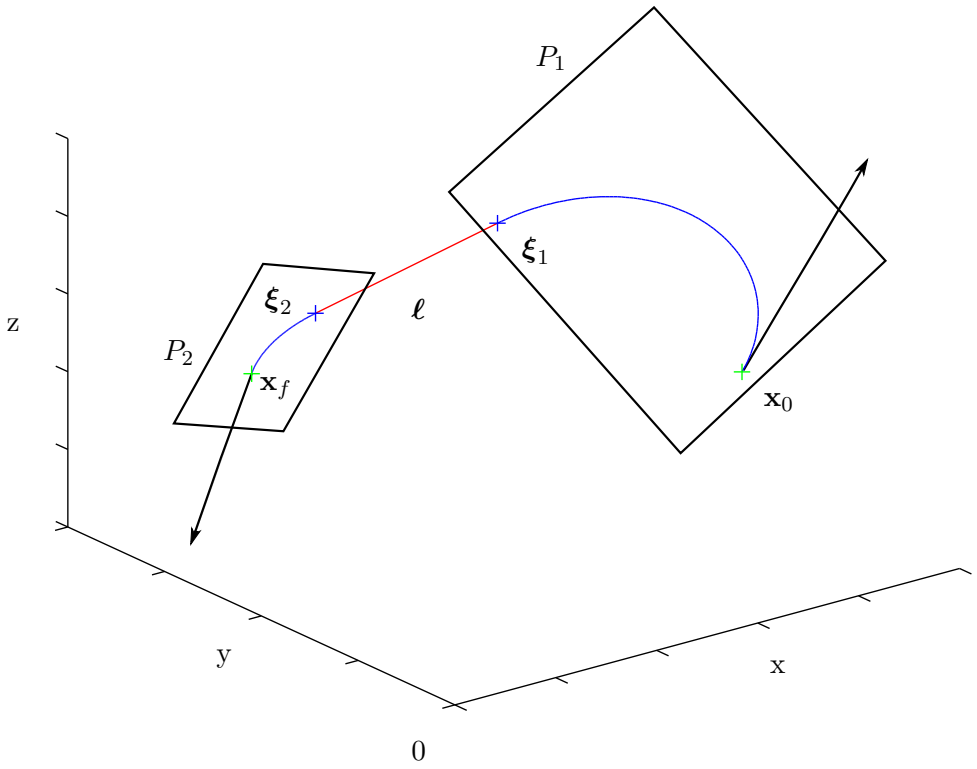


Figure 7.4: Example of Dubins' path in 3D

In order to find the shortest Dubins' path in a heterogeneous environment shown in figure 7.4, let $\boldsymbol{\ell} \in \mathbb{R}^3$ denote a line segment which lies in both plane P_1 and P_2, *i.e.* $\boldsymbol{\ell} \in P_1$ and $\boldsymbol{\ell} \in P_2$.

In the followings, the cross product of $\mathbf{u}$ and $\mathbf{v}$ is defined by $\mathbf{u} \times \mathbf{v}$. In order to find both curves, the following normal vector to each particular plane P_1 and P_2 must be defined:

- The unit vector perpendicular to the first plane:

$$\mathbf{b}_1 = \frac{\boldsymbol{\ell} \times \mathbf{v}_0}{||\boldsymbol{\ell} \times \mathbf{v}_0||}; \tag{7.25}$$

- The unit vector perpendicular to the second plane:

$$\mathbf{b}_2 = \frac{\boldsymbol{\ell} \times \mathbf{v}_\mathrm{f}}{||\boldsymbol{\ell} \times \mathbf{v}_\mathrm{f}||}; \tag{7.26}$$

Remark 6 *In case $\boldsymbol{\ell} \times \mathbf{v}_0 = \mathbf{0}$ or $\boldsymbol{\ell} \times \mathbf{v}_f = \mathbf{0}$, it means that there is no curve. Thus, the CSC path degrades to CS, SC, or S path.*

γ_{p_1} on plane P_1 and γ_{p_2} on plane P_2 in equation (7.23) are defined as follows

$$\gamma_{p_1} = \angle(\mathbf{e}_1^{p_1}, \boldsymbol{\ell}) \tag{7.27}$$

$$\gamma_{p_2} = \angle(\mathbf{e}_1^{p_2}, \boldsymbol{\ell}) \tag{7.28}$$

Then, the position of $\boldsymbol{\xi}_1$ on plane P_1 and $\boldsymbol{\xi}_2$ on plane P_2 can be found using the calculation shown in Section 7.3.5.1. Therefore, $\boldsymbol{\xi}_1 = (x_1, y_1, z_1)^\top$ and $\boldsymbol{\xi}_2 = (x_2, y_2, z_2)^\top$ in frame $\mathcal{I}$ can be found as follows:

$$\begin{cases} x_1 = -z_{p_1} \sin\phi_1 \cos\psi_1 - x_{p_1} \sin\psi_1 + x_0 \\ y_1 = -z_{p_1} \sin\phi_1 \sin\psi_1 + x_{p_1} \cos\psi_1 + y_0 \\ z_1 = z_{p_1} \cos\phi_1 + z_0 \end{cases} \tag{7.29}$$

$$\begin{cases} x_2 = -z_{p_2} \sin\phi_2 \cos\psi_2 - x_{p_2} \sin\psi_2 + x_f \\ y_2 = -z_{p_2} \sin\phi_2 \sin\psi_2 + x_{p_2} \cos\psi_2 + y_f \\ z_2 = z_{p_2} \cos\phi_2 + z_f \end{cases} \tag{7.30}$$

Recall that both curves are obtained by considering $(x_{p_1}, y_{p_1}, z_{p_1}) = (x_{p_2}, y_{p_2}, z_{p_2}) = (0, 0, 0)$ as an origin in this coordinate system, *i.e.* $(x_{p_1}, y_{p_1}, z_{p_1}) = \mathbf{x}_0$ and $(x_{p_2}, y_{p_2}, z_{p_2}) = \mathbf{x}_f$ in frame $\mathcal{I}$, and (ϕ_1, ψ_1) and (ϕ_2, ψ_2) are orientation of $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively.

The orientations of $\mathbf{v}_1$ and $\mathbf{v}_2$ can be found by rotating $\mathbf{v}_0$ and $\mathbf{v}_\mathrm{f}$ by $\Delta\gamma_1 = \gamma_{p_1} - \gamma_{p_0}$ and $\Delta\gamma_2 = \gamma_{p_2} - \gamma_{p_f}$ around the vector $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively.

Given an axis $\mathbf{b}$ and an angle $\delta\gamma$, the rotation matrix can be found using the following formulation:

$$R = \cos\Delta\gamma \mathbf{I} + \sin\Delta\gamma(\mathbf{b} \times \mathbf{b}) + (1 - \cos\Delta\gamma)(\mathbf{b} \otimes \mathbf{b}) \tag{7.31}$$

where $\times$ is the cross product operation and $\otimes$ is the tensor product operation.

By applying this formulation (7.31), We have

$$\mathbf{v}_1 = R_{b_1} \mathbf{v}_0 \tag{7.32}$$

$$R_{b_1} = \cos\Delta\gamma_1 \mathbf{I}_3 + \sin\Delta\gamma_1 \mathbf{C}_1 + (1 - \cos\Delta\gamma_1)\mathbf{D}_1 \tag{7.33}$$

$$\mathbf{v}_2 = R_{b_2} \mathbf{v}_\mathrm{f} \tag{7.34}$$

$$R_{b_2} = \cos\Delta\gamma_2 \mathbf{I}_3 + \sin\Delta\gamma_2 \mathbf{C}_2 + (1 - \cos\Delta\gamma_2)\mathbf{D}_2 \tag{7.35}$$

where $\mathbf{I}_3$ is an identity matrix of order 3. $\mathbf{C}_1$ and $\mathbf{C}_2$ are the cross products of $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively. $\mathbf{D}_1$ and $\mathbf{D}_2$ are the tensor products of $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively. They are

defined as follows:

$$\mathbf{C}_1 = \begin{bmatrix} 0 & -b_{1_z} & b_{1_y} \\ b_{1_z} & 0 & -b_{1_x} \\ -b_{1_y} & b_{1_x} & 0 \end{bmatrix}$$

$$\mathbf{D}_1 = \begin{bmatrix} b_{1_x}^2 & b_{1_x}b_{1_y} & b_{1_x}b_{1_z} \\ b_{1_x}b_{1_y} & b_{1_y}^2 & b_{1_y}b_{1_z} \\ b_{1_x}b_{1_z} & b_{1_y}b_{1_z} & b_{1_z}^2 \end{bmatrix}$$

$$\mathbf{C}_2 = \begin{bmatrix} 0 & -b_{2_z} & b_{2_y} \\ b_{2_z} & 0 & -b_{2_x} \\ -b_{2_y} & b_{2_x} & 0 \end{bmatrix}$$

$$\mathbf{D}_2 = \begin{bmatrix} b_{2_x}^2 & b_{2_x}b_{2_y} & b_{2_x}b_{2_z} \\ b_{2_x}b_{2_y} & b_{2_y}^2 & b_{2_y}b_{2_z} \\ b_{2_x}b_{2_z} & b_{2_y}b_{2_z} & b_{2_z}^2 \end{bmatrix}$$

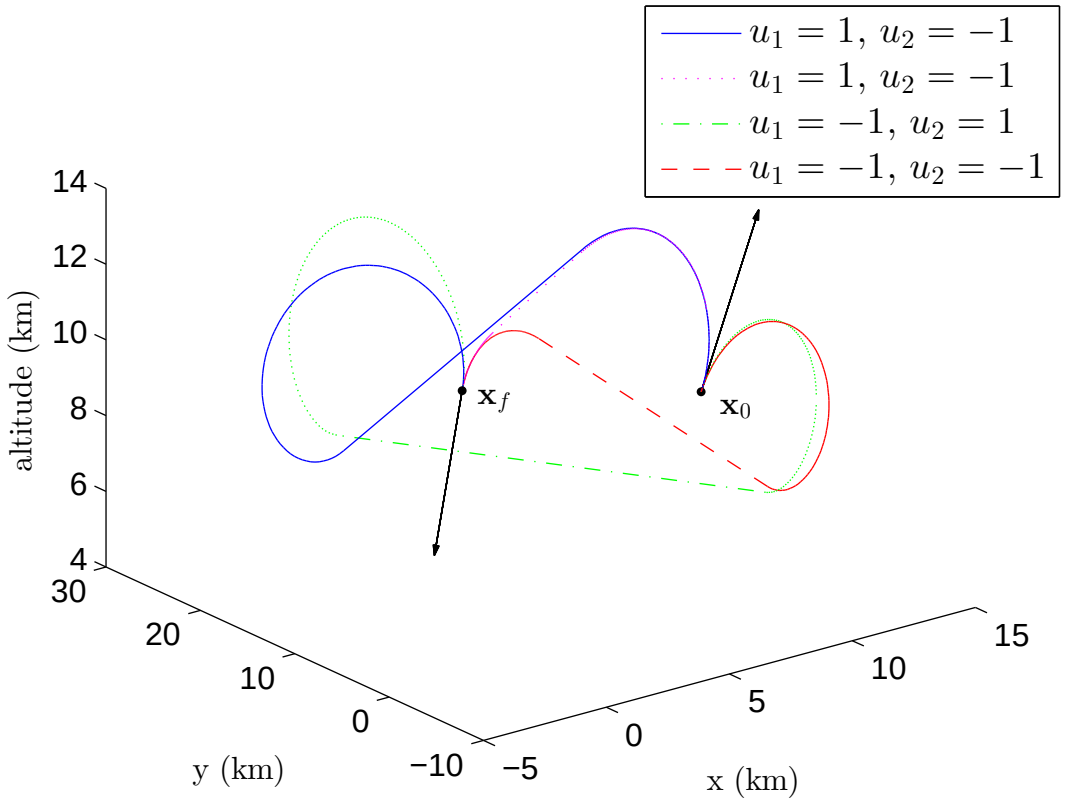


Figure 7.5: Four possible CSC paths between two states

Once two curves have been found, for example, a Newton method, is used to find the

parameter $\boldsymbol{\ell}$ by verifying the objective function $F(\boldsymbol{\ell}) = \boldsymbol{\ell} - (\boldsymbol{\xi}_2 - \boldsymbol{\xi}_1) = \mathbf{0}$. Thus, the line segment $\boldsymbol{\ell}$, which is on both plane P_1 and plane P_2, connecting both curves is found.

Remark 7 *With this methodology, the conditions* $\boldsymbol{\ell} \times \mathbf{v}_1 = \mathbf{0}$ *and* $\boldsymbol{\ell} \times \mathbf{v}_2 = \mathbf{0}$ *are automatically verified.*

There can exist four types of CSC paths where $u_p = \pm 1$ for both circular arcs shown in figure 7.5. The path with the shortest length, presented by the violet dotted curve, is chosen and its distance is used as a distance metric $d(\mathbf{x}_1, \mathbf{x}_2)$ between two states.

7.4 Simulation results

Here, MATLAB is used as a simulator to visualize and analyze the results. As author is interested in the methodology of solving the problems, no robustness of coded algorithms is analyzed. Moreover, MATLAB is used to simulated the results which is not the real language used on the real embedded system. Therefore, the computational effort is discussed in number of iterations instead of the computation time.

7.4.1 2D application

Two scenarios are analyzed with the initial state $\mathbf{x}_{\text{init}} = (0\text{km}, 0\text{km}, \pi/2)$,

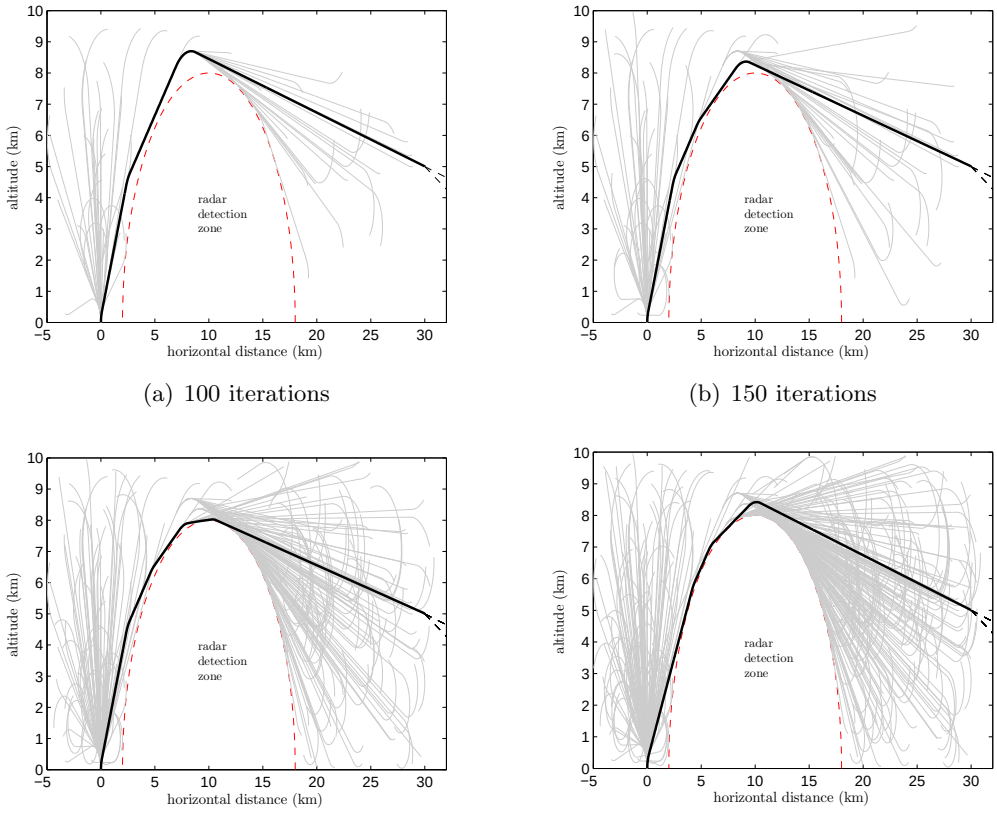
$$\begin{aligned} \mathbb{P}_{\text{goal}} &= \{\boldsymbol{\xi} \in \mathbb{R}^2 : \|\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{pip}}\| < 500\text{m}\}, \\ \mathbb{V}_{\text{goal}} &= \{\gamma \in \mathbb{R} : \gamma \in \mathbb{C}(\boldsymbol{\xi}, \gamma_{\text{pip}}, \pi/8)\}, \end{aligned} \tag{7.36}$$

with $\boldsymbol{\xi}_{\text{pip}} = (30\text{km}, 5\text{km})^\top$ and $\gamma_{\text{pip}} = -\pi/12$ for both scenarios. Moreover, the bias to $\mathbb{X}_{\text{goal}}$ is set to $p = 0.1$, the integration distance $\Delta s = 1\text{km}$, $\epsilon = 3$ steps, and $\rho_0 = 1.225\text{km/m}^3$, $z_r = 7.5\text{km}$ for the environment model.

The first scenario has only one obstacle which is a coverage area of the detection radar station. It is centered at (10km,0km) with 8km detection radius. While the second scenario has two obstacle areas that increase the complexity of the problem. In this scenario, the obstacles are a fixed direction radar detection beam whose origins are (−8km,0km) and (19km,0km). The initial and final conditions and obstacles are chosen while considering the feasibility and the interceptor missile mission.

In the following figures, $\mathbb{X}_{\text{goal}}$ is represented as a point with two dashed lines, $\mathbb{X}_{\text{obs}}$ is represented by space surrounded by red dashed curves, the exploration tree is represented by the thick grey solid curve is the solution found by the RRT* algorithm.

Figure 7.6 shows one of the simulation results for scenario 1. Subfigures show the improvement of the solutions while the number of iterations increases. Figure 7.6(d) shows



(a) 100 iterations

(b) 150 iterations

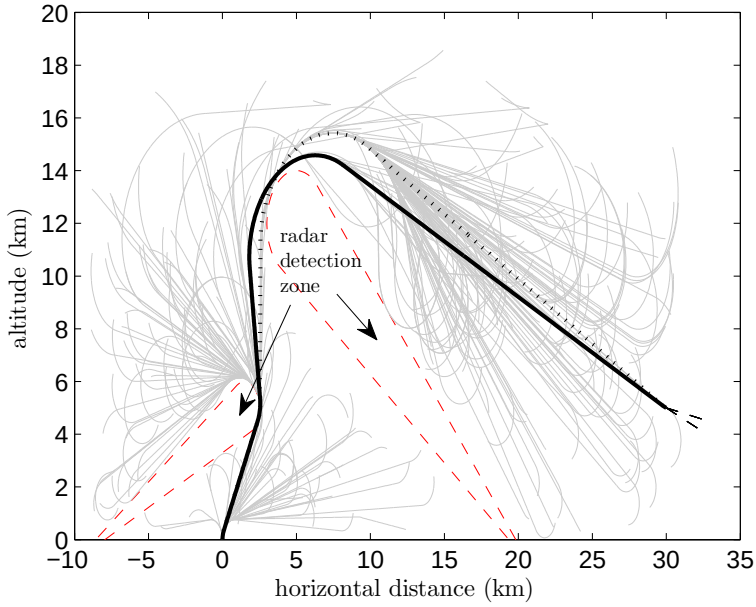
(c) 300 iterations

(d) 400 iterations

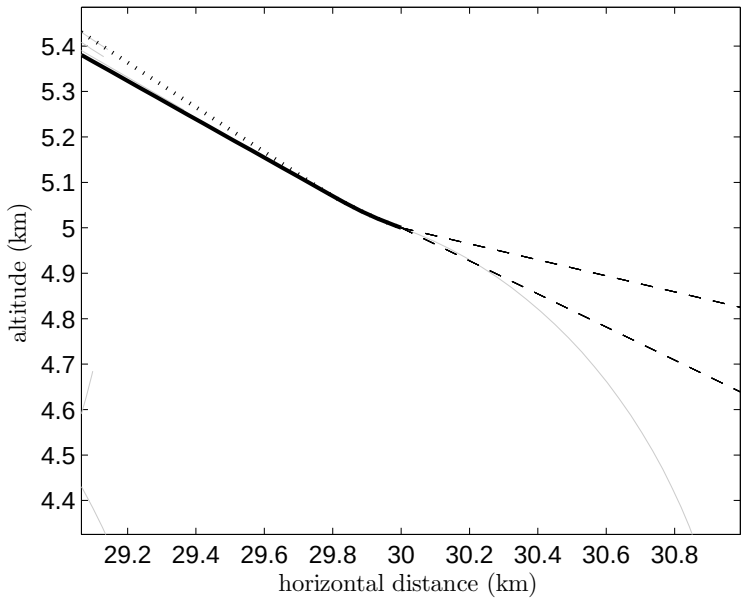
Figure 7.6: Exploration tree expansion and results for scenario 1

the shortest length solution with 33.5km found within 400 iterations. Results from 100 Monte-Carlo simulations show that the first solution is acquired around 69^{th} iteration and the average shortest path is 34.2km long. As we can see from the improvement of the obtained solutions over time, the search continues after the first solution is found which is the advantage of this algorithm.

In scenario 2, several obstacles increase the complexity of the problem. A simulation result for scenario 2 is shown in figure 7.7. After 400 iterations, several solutions were found but just two solutions are presented in figure 7.7(a). The first obtained solution is represented by the thick dotted curve and the last obtained solution is represented by the thick solid curve with a length of 43.5km. As it is not clear that these trajectories respect the final constraints in figure 7.7(a), figure 7.7(b) represents the enlarged area around $\mathbb{X}_{\text{goal}}$ which shows that the vehicle makes a small turn at the end of the trajectory to arrive in $\mathbb{X}_{\text{goal}}$ while respecting the final constraints. According to the 100 Monte-Carlo simulations, the mean iteration where the first solution is obtained is 152 and the mean path length solution is 44.2km.



(a) 400 iterations



(b) Enlarged image at $\mathbb{X}_{\text{goal}}$ area

Figure 7.7: Simulation result for scenario 2

7.4.2 3D application

In order to test the algorithm framework, the scenario shown in figure 7.8 is considered with the following configurations: the initial state $\mathbf{x}_{\text{init}} = (1\text{km}, 1\text{km}, 1\text{km}, \pi/2, 0)^\top$,

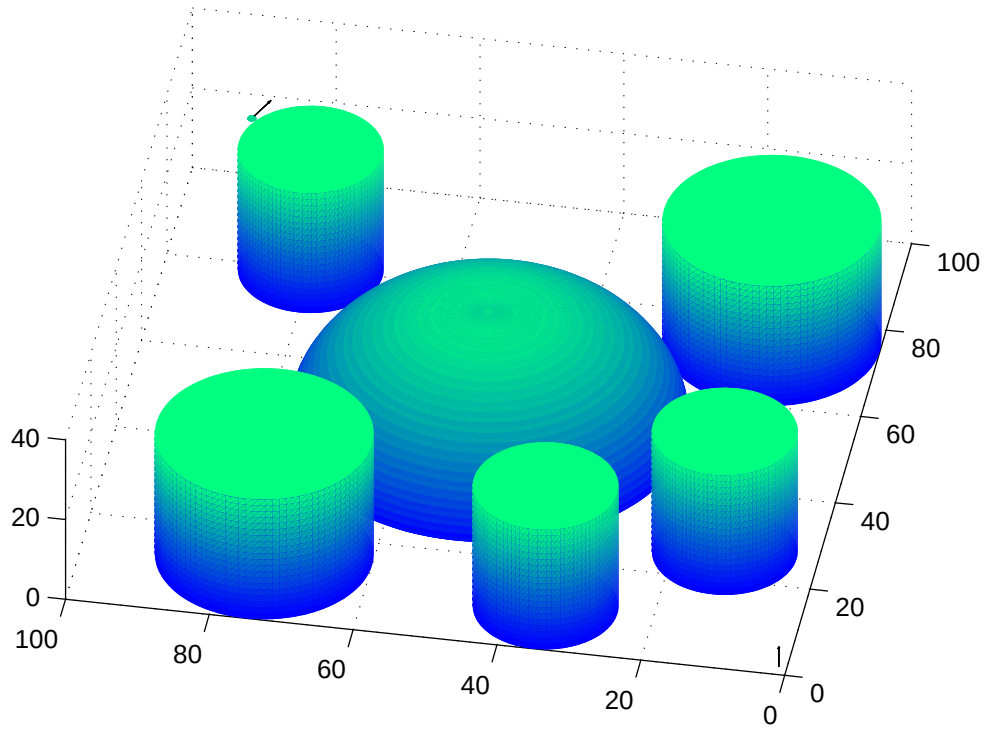


Figure 7.8: Illustration of the scenario

$$\begin{aligned}
\mathbb{X}_{\text{goal}} &= \mathbb{P}_{\text{goal}} \times \mathbb{V}_{\text{goal}}, \\
\mathbb{P}_{\text{goal}} &= \{\boldsymbol{\xi}_{\text{rdv}}\}, \\
\mathbb{V}_{\text{goal}} &= \{(\gamma, \chi)^\top \in \mathbb{R}^2 : (\gamma, \chi) \in \mathbb{C}(\mathbf{x}_{\text{rdv}}, 5^\circ)\},
\end{aligned} \tag{7.37}$$

with $\mathbf{x}_{\text{rdv}} = (90\text{km}, 90\text{km}, 25\text{km}, 0, -\pi/8)^\top$.
In figure 7.8, the arrows represent the orientations of the initial state $\mathbf{x}_{\text{init}}$ and of the rendez-vous state

$$\mathbf{x}_{\text{rdv}} = \left(\boldsymbol{\xi}_{\text{rdv}}^\top, \gamma_{\text{rdv}}, \chi_{\text{rdv}}\right)^\top.$$

$\mathbb{X}_{\text{obs}}$ is represented by the 3-dimensional shaded surfaces (in gradient color). The nature of obstacles can be no-fly zones such as the area above and around cities which are represented by cylinders and radar detection zones which are represented by a half sphere.

Here, 3 different algorithms are considered:

- RRT
- RRT*

- RRT* biased by APF with a given margin for the orientation of the randomly generated state $\mathbf{x}_{\text{rand}}$ of 10° ($\phi_{\text{APF}} = 10^\circ$)

For all algorithms, a bias toward the goal consisting in generating $\mathbf{x}_{\text{rand}} \in \mathbb{X}_{\text{goal}}$ with a probability $p = 0.1$ is used.

In the following figures, the exploration tree is represented by the thick light grey solid curve is the final solution found by the algorithms. Figure 7.9 shows a result obtained after 1000 iterations by the RRT algorithm. It shows that the RRT algorithm is capable of finding a solution for the problem. However, the obtained trajectory is clearly not the shortest since some loops and turns can be observed. This is because the RRT algorithm does not consider any optimal criteria. Moreover, no reconnection is done in order to improve the path quality.

The other two algorithms are employed on the same scenario. Figure 7.10 shows one of the best results obtained by the RRT* and by the RRT* biased by APF after 200 iterations. As we can see from the results, the RRT* algorithm can find a better solution than the RRT. In figure 7.10, the difference between the two RRT* algorithms is that the exploration tree of the RRT* biased by APF (see figure 7.10(b)) tends to direct to the destination while the other one extends in every direction (see figure 7.10(a)).

For in-depth analysis, for each algorithm, 100 Monte Carlo simulations within 200 iterations are simulated to obtain statistic results. The average iteration needed to obtain the first solution, the average length of the first solution, and the average length of the final solution are observed in table 7.1. According to the statistic results in table 7.1, the RRT algorithm has the worst performance in all observed values, which is expected because no optimal criterion and reconnection are considered. For the RRT* algorithm, the first solution is obtained at mostly the same iteration as the RRT algorithm but with a better result. Moreover, the optimality of the final solution is improved with respect to the number of iterations. While the RRT* biased by APF is the fastest to find the first solution. On top of that the solutions (both first and final solutions) are more optimal than the others. The reason is that the heuristic using the APF helps to bias the search toward the feasible solutions rather than blindly search the exploration space, *i.e.* increase the probability of reconnection of the RRT*. Obviously, the RRT* can achieve the same final performance if more iterations are given.

Monte-Carlo simulations of several scenarios which are not shown here have also been simulated. For less complex scenarios, *i.e.* less obstacles, the results of the RRT* and the RRT* biased with APF are mostly the same. Since there are few obstacles, the algorithms converge rapidly because the probability of reconnection is already high. Thus, it is reasonable that the performance does not improve as much in less complex environments.

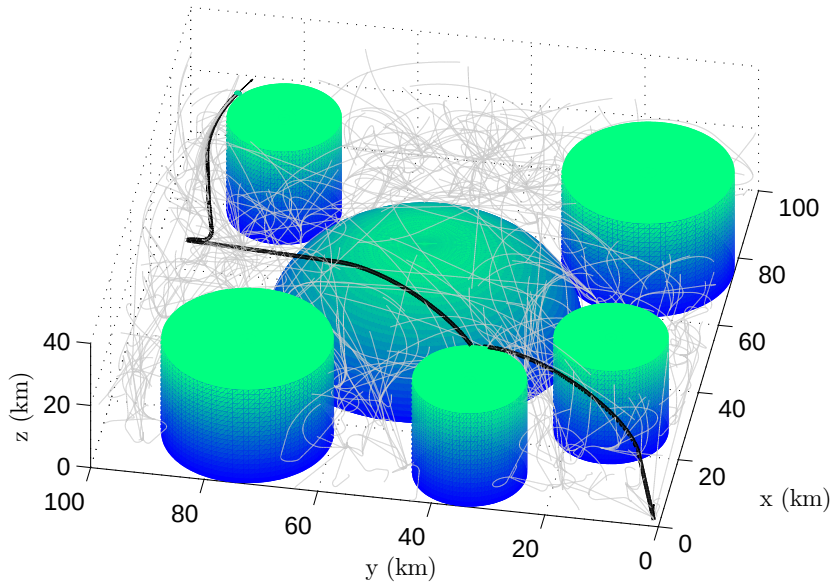


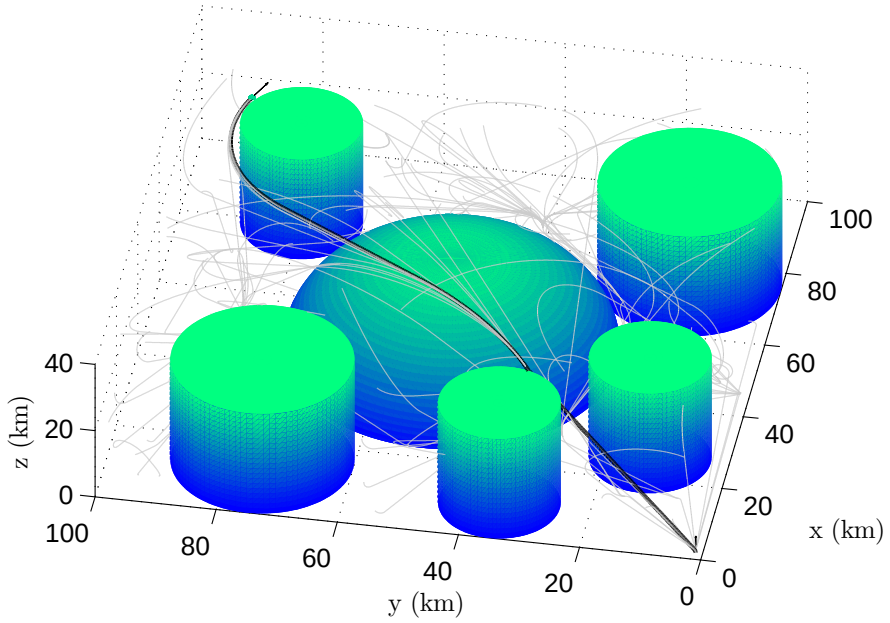
Figure 7.9: Exploration trees and results after 1000 iterations of RRT algorithm: path length 200km

Method	1st solution (iteration)	Average length of the 1st solution (km)	Average length of the final solution (km)
RRT	85	241	239
RRT*	82	197	185
RRT* biased by APF	46	169	155

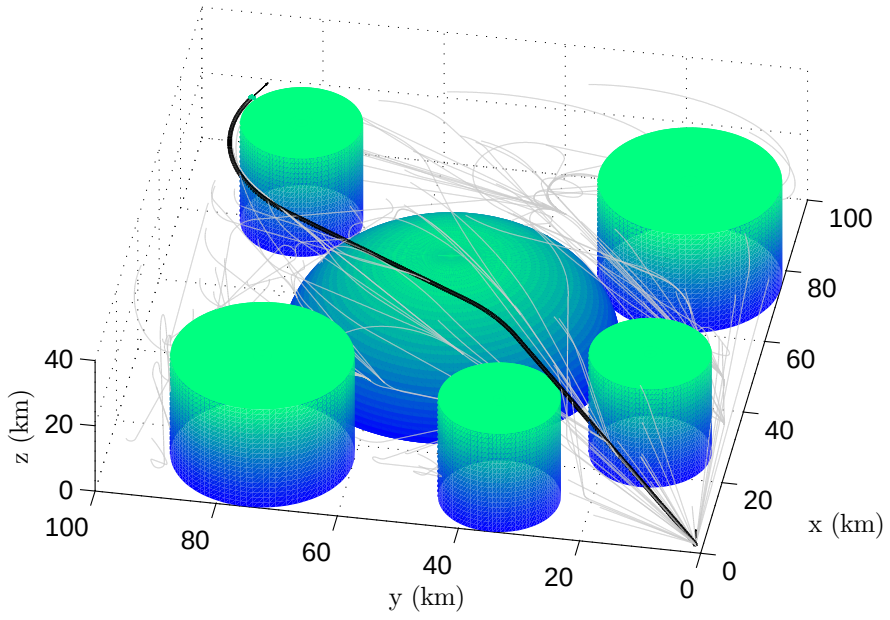
Table 7.1: Results of 100 Monte-Carlo simulations within 200 iterations

7.5 Conclusions

In this chapter, an efficient trajectory planning framework for aerial vehicles traveling in 2D and 3D space while avoiding obstacles is presented. The algorithm is based on the RRT* algorithm to find an optimal trajectory. In 2D application, the original RRT* algorithm is used. It is shown that the RRT* algorithm is capable of finding a solution for the trajectory planning of the aerial vehicle in vertical plane while avoiding obstacles. The solution keeps on improving during the remaining time that results in finding a near-optimal solution. The trajectories obtained by this framework is based on the simplified system model. Thus, it is not totally executable by the real system. However, the simplified system is more adaptable to the aerial vehicles model than the Dubins' car model. This makes the obtained trajectories be close to the real executable trajectory. Hence, they can be used as reference



(a) RRT*: path length 140.36km



(b) RRT* biased by APF: path length 139.89km

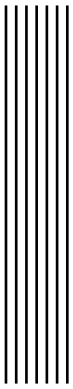
Figure 7.10: Exploration trees and results after 200 iterations

trajectories which facilitate the control using, for example, MPC algorithm. However, the 2D trajectory is not interesting for the aerial vehicle trajectory planning because the aerial

vehicles usually fly in 3D space.

In the 3D application, the APF is integrated in the RRT* algorithm to accelerate the convergence speed to the optimal solution. The capability of this framework is demonstrated on a missile application. The 3D Dubins' paths in a heterogeneous environment are used for the metric. As a result, the approach shows good performance in simulation results. Compared to the RRT and the RRT* algorithms, better solutions can be found with much less number of iterations due to the integration of the APF. Thus, real-time constraints will be easier to verify if the algorithm is implemented on board the vehicle. The possibility of implementing the algorithm on the real system will be discussed in perspectives and future works in the following chapter.

Note that the researches on 2D application described in this chapter are published in a paper presented in "The IEEE International Conference on Robotics and Automation" under the title "Shortest path for aerial vehicles in heterogeneous environment using RRT*" in 2015 and the researches on 3D path planning described in this chapter are submitted to "The IEEE Transactions on Control Systems Technology" under the title "3D trajectory planning of aerial vehicles using RRT*" in 2015. Moreover, the development of the 3D Dubins' paths is published in a paper presented in "The IFAC Workshop on Advances Control and Navigation for Autonomous Aerospace Vehicles" under the title "3D-shortest paths for a hypersonic glider in a heterogeneous environment."



Conclusions and perspectives

In the new era of autonomous aerial vehicles, trajectory planning module becomes an important part of autonomous systems. The aim of the trajectory planning is to find a feasible and optimal trajectory for autonomous vehicles. The obtained trajectory is used as a decision for autonomous vehicles to execute missions by themselves. In the manuscript of this PhD thesis, trajectory planning algorithms for aerial vehicles with constraints have been researched. It is divided into two parts: trajectory planning using a realistic model and path planning using a simplified model.

The aim of the trajectory planning by using a realistic model is to obtain directly the executable trajectory. The sequence of control inputs is obtained along with the trajectory. Thus, there is no need for path following algorithms to execute this trajectory. This approach is ideal for any trajectory planning algorithms. The RRT algorithm is used as a basis path planner. The principle of the RRT is to explore non-convex high-dimensional space by growing the search tree toward unexplored areas. MATLAB is used to simulate the results for analysis. Moreover, as author is interested in developing the methodology that can solve the problems, no robustness of coded algorithms is analyzed. Moreover, MATLAB is used to simulated the results which is not the real language used on the real embedded system. Therefore, the computational effort is discussed in number of iterations instead of the computation time.

First, the RRT algorithm is tested in a missile application to intercept a target at the PIP. The Dubins' paths are used to determine the distance between two vehicle states. The classical guidance laws are used to expand the exploration tree toward unexplored areas. The results show that the RRT algorithm is capable of finding a feasible solution while the classical guidance laws have some difficulties in solving some scenarios. However,

no optimal criterion is considered in the algorithm and the computational effort can be large. Therefore, a preprocessing of the exploration space is proposed in order to solve these problems.

Two methods are proposed using APF or Dubins' paths in a heterogeneous environment for preprocessing. However, it is very difficult to find a suitable APF function for a nonholonomic system. Moreover, several tuning parameters are required. Thus, it is not suitable for our situation. Yet, some studies in this topic is shown in Appendix C. Therefore, the shortest Dubins' path in a heterogeneous environment is used as a preprocessing method. Since Dubins' paths are considered optimal solutions, the exploration space is then reduced and hypothetically contains optimal and suboptimal solutions. According to the results of simulations, the quality of obtained trajectories improves a lot compared to the results obtained using the RRT algorithm. The results are also obtained with less number of iterations. However, there is no theoretical proof that the preprocessed exploration space contains the optimal solution except for the hypothesis. Moreover, the preprocessed exploration space makes the RRT algorithm lose its charm as the method of exploration for high-dimensional problems without a need of approximation. Therefore, another approach is proposed.

Since the complete system model makes the problem difficult to prove the optimality of the solution, the system model is simplified. Instead of finding a trajectory corresponding to the complete system model, the trajectory planning algorithm focuses on finding a reference trajectory using a simplified system model which is easy to follow by using path following algorithms such as MPC. The optimal RRT or RRT* is used as a basis path planner to find an optimal reference trajectory between two vehicle states. The RRT* is the RRT algorithm with the asymptotic optimality property, *i.e.* almost-sure convergence to an optimal solution, along with probabilistic completeness guarantees.

The RRT* algorithm is tested in hypersonic aerial vehicle applications such as interceptor missiles. The Dubins' paths in a heterogeneous environment are used to determine the distance between two vehicle states in 2D plane. These Dubins' paths are different from the original Dubins' paths that the maximum path curvature of the vehicle is no longer constant and depends on position of the vehicle. These Dubins' paths are also used as node expansion method since they are calculated using the same simplified model used in the RRT* algorithm. The simulation results are very satisfying since an optimal solution can be obtained. However, it is not interesting enough to consider the 2D problem. Thus, 3D Dubins' paths in a heterogeneous environment are developed. Moreover, in order to increase the convergence rate to the optimal solution, the integration of the APF is proposed.

The APF is used as a heuristic to guide randomly generated nodes toward the goal. The

APF is generated around the obstacles and tends to move toward the desired destination. By combining the APF with the RRT*, the convergence rate of the RRT* algorithm to the optimal solution increases while the RRT* algorithm solves the local minima problems of the APF with the randomly generated states. In the hypersonic aerial vehicle application, the 3D Dubins' paths in a heterogeneous environment are developed based on the assumption that two vehicle states are sufficiently far from each other so that the shortest path is a CSC path. Dubins' paths are used as a metric and also as a node expansion method. The simulation results show that the APF does not increase the convergence rate to the optimal solution as much as in simple scenarios where there is no or less obstacles. However, compared to the RRT and the RRT* algorithm, better solutions are found with less number of iterations in a very complex environment where there are a lot of obstacles. Thus, real-time constraints will be easier to verify; assuming that less number of iterations is equal to less computation time, if the algorithm is implemented on board the vehicle.

The work in this thesis show promising results in finding a 3D optimal trajectory for aerial vehicles in a heterogeneous environment. In future work, replanning ability of the algorithm can be studied in order to implement the algorithm on board the vehicle. The replanning ability of the algorithm is very important to missions in an unknown or dynamic environment since it always searches for a solution. Anytime algorithm also has to be considered in order to always have a solution during the mission. In the application, the 3D Dubins' paths can also be studied with presence of constant wind. The realistic model can also be considered using the Dubins' paths in a heterogeneous environment as metric. However, suitable control laws must be used. The only requirement of the suitable control laws is that they must be capable of arriving at the desired state. At the same time, the experimental test of the algorithms on the real UAVs can also be studied. Since the framework is proposed in this thesis, the only thing required to implement this framework on the real system is the real data of the vehicle to be used in the system model. This study can be done by Phung Duc Kien, a postdoctoral researcher, at Onera.

Part V

Appendix

A Calculation of time-vary gains for Kappa guidance

Note that the calculation in this chapter is reformulated based on the demonstration in [Lin91] and is adapted to the system and notations in this thesis. Recall equation (3.9):

$$\mathbf{a}_{\text{com}} = \mathbf{a}_1 + \mathbf{a}_2,$$

with

$$\mathbf{a}_1 = \frac{K_1}{t_{\text{go}}}(\mathbf{v}_{\text{pip}} - \mathbf{v}_{\text{m}_0}),$$
$$\mathbf{a}_2 = \frac{K_2}{t_{\text{go}}^2}(\mathbf{r} - \mathbf{v}_{\text{c}} t_{\text{go}}),$$

choosing $||\mathbf{v}_{\text{pip}}|| = ||\mathbf{v}_{\text{m}_0}||$, K_1 and K_2 are the gains.

Let δ be the predicted velocity angle error of the present and final vectors, and σ be the heading error angle, R be the length of the LOS. To reduce the effect of an uncontrolled axial acceleration command, let the first term of the equation (3.9) at the normal direction to the velocity vector be $(K_1/t_{\text{go}})v\sin\delta$. The second term at the normal direction to the velocity vector is reduced as $-(K_2/t_{\text{go}})v\sin\delta/\cos\delta$. Moreover, R can be estimated as $R \approx t_{\text{go}} v\cos\sigma$. Thus, the normal acceleration in the body-fixed coordinates is written as

$$a = \frac{K_1}{R}v^2\sin\delta\cos\sigma - \frac{K_2}{R}v^2\sin\sigma. \tag{A.1}$$

The instantaneous curvature κ of the vehicle trajectory is defined as

$$\kappa = \frac{d\hat{\gamma}}{ds} = \frac{1}{v}\frac{d\hat{\gamma}}{dt} = \frac{1}{v^2}a, \tag{A.2}$$

where $\hat{\gamma}$ is the flight angle about some inertial reference and $ds/dt = v$. From equations (A.1) and (A.2), the optimal curvature κ is

$$\kappa = \frac{K_1}{R}\sin\delta\cos\sigma - \frac{K_2}{R}\sin\sigma. \tag{A.3}$$

A.1 Analytic optimal control in vertical plane

In aeronautics, it is more convenient to use the guidance law in vehicle velocity coordinates (equation (A.1)) than the one in inertial coordinates (equation (3.9)). The optimal guidance gains K_1 and K_2 can be derived with a performance index

$$J = \max v_{\mathrm{f}}, \tag{A.4}$$

and boundary conditions $\delta(t_{\mathrm{f}}) = \sigma(t_{\mathrm{f}}) = 0$. Maximizing the terminal speed in performance index is equivalent to minimizing the total loss in the kinetic energy of the flight path. Thus, it promises the extended range with more favorable endgame conditions which is the goal of this optimal trajectory shaping.

In the vertical plane, the flight path angle γ is equal to $\hat{\gamma}$ and δ, σ can be written as

$$\delta(R) = \gamma_{\mathrm{f}} - \gamma, \tag{A.5}$$

$$\sigma(R) = \gamma + \theta, \tag{A.6}$$

where θ is the inertial LOS angle and

$$\frac{dR}{dt} = -v\cos\sigma, \tag{A.7}$$

$$\frac{d\theta}{dt} = \frac{v\sin\sigma}{R}. \tag{A.8}$$

The state equations governing the states γ and σ and the control κ are derived from (A.2), (A.6), and (A.7) and (A.8) as

$$\frac{d\gamma}{dR} = -\kappa\sec\sigma, \tag{A.9}$$

$$\frac{d\sigma}{dR} = -\kappa\sec\sigma - \frac{\tan\sigma}{R}. \tag{A.10}$$

In order to obtain the optimum guidance law $\mathbf{a}_{\mathrm{c}}(t)$, the optimal control theory needs to be applied to find an optimal control law $\kappa(R)$ respecting to the state equations (A.9) and (A.10) and the boundary conditions $\delta(t_{\mathrm{f}}) = 0$, $\sigma(t_{\mathrm{f}}) = 0$ and $R(t_{\mathrm{f}}) = 0$ such that the missile terminal speed is maximized. Thus, we maximize the cost function

$$G = \int_R^0 \frac{dv}{dR}dR, \tag{A.11}$$

where G is a function of the state variables σ and γ, the control variable κ, and the missile characteristics.

By neglecting mass change due to fuel consumption, the equations of motion can be written as

$$\dot{v} = \frac{T\cos\alpha - QSC_{\mathrm{D}}}{m} - g\sin\gamma, \tag{A.12}$$

$$v\dot{\gamma} = \frac{T\sin\alpha + QSC_{\mathrm{L}}}{m} - g\cos\gamma, \tag{A.13}$$

where $T = ||\mathbf{f}_{\mathrm{thrust}}||$ is the thrust of the missile, α is the angle of attack, $Q = \frac{1}{2}\rho v^2$ is the dynamic pressure, C_{D} and C_{L} are the aerodynamic coefficients, g is the magnitude of the gravity, and γ is the flight path angle of the vehicle.

By assuming $\cos\alpha \approx 1-\alpha^2/2$, the derivation of G can be simplified to obtain an analytic solution of κ. Thus, the equation (A.12) can be written as

$$\frac{dv}{dR} = \frac{QSC_{\mathrm{A}} - T + mg\sin\gamma}{mv\cos\sigma} + \frac{T/2 - QSC_{\mathrm{A}}/2 + QSC_{\mathrm{N}\alpha}}{mv\cos\sigma}\alpha^2. \tag{A.14}$$

Moreover, by assuming $\sin\alpha \approx \alpha$ using equations (A.2) and (A.13), the angle of attack can be formulated as

$$\alpha = \frac{mv^2(\kappa - g\cos\gamma/v^2)}{T + QS(C_{\mathrm{N}\alpha} - C_{\mathrm{A}})}. \tag{A.15}$$

By substituting (A.15) and (A.14) into (A.11), the cost function G is expressed as

$$G = \frac{QSC_{\mathrm{A}}}{mv}I - \int_R^0 \frac{T - mg\sin\alpha}{mv\cos\sigma}dR, \tag{A.16}$$

where

$$I = \int_R^0 \left(1 + \frac{m^2v^4(\kappa - g\cos\alpha/v^2)^2(T/2 + QSC_{\mathrm{N}\alpha} - QSC_{\mathrm{A}}/2)}{QSC_{\mathrm{A}}(T + QS(C_{\mathrm{N}\alpha} - C_{\mathrm{A}}))^2\cos\sigma}\right)dR. \tag{A.17}$$

Moreover, if the effect of gravity is neglected, equation (A.16) can be rewritten as

$$G_1 = \frac{QSC_{\mathrm{A}}}{mv}I_1 - \int_R^0 \frac{T}{mv\cos\sigma}dR, \tag{A.18}$$

where

$$I_1 = \int_R^0 \left(1 + \frac{\kappa^2}{2F^2}\right)\sec\sigma dR \text{ and } F^2 = \frac{QSC_{\mathrm{A}}(T + QS(C_{\mathrm{N}\alpha} - C_{\mathrm{A}}))^2}{2m^2v^4(T/2 + QSC_{\mathrm{N}\alpha} - QSC_{\mathrm{A}}/2)}. \tag{A.19}$$

A.1.1 Vehicle without propulsion

For the power-off stage of the vehicle, the second term of equation (A.18) does not exist. Then, maximizing I_1 is equivalent to maximizing G_1, *i.e.* I_1 becomes the new cost function. The Hamiltonian is

$$H = \left(1 + \frac{\kappa^2}{2F^2}\right)\sec\sigma - \lambda_\sigma\frac{\tan\sigma}{R} - (\lambda_\sigma + \lambda_\gamma)\kappa\sec\sigma, \tag{A.20}$$

and the adjoint equations can be written as

$$\frac{d\lambda_\sigma}{dR} = -\frac{\partial H}{\partial \sigma} = \left(1 + \frac{\kappa^2}{2F^2}\right)\sec\sigma\tan\sigma - \frac{\lambda_\sigma}{R}\sec^2\sigma - (\lambda_\sigma + \lambda_\gamma)\kappa\sec\sigma\tan\sigma, \tag{A.21}$$

$$\frac{d\lambda_\gamma}{dR} = -\frac{\partial H}{\partial \gamma} = 0 \Longrightarrow \lambda_\gamma = C_0, \tag{A.22}$$

and

$$\frac{\partial H}{\partial \kappa} = \frac{\kappa}{F^2} - (\lambda_\sigma + \lambda_\gamma)\sec\sigma = 0. \tag{A.23}$$

Thus, the optimal variable κ is expressed as

$$\kappa = F^2(\lambda_\sigma + C_0). \tag{A.24}$$

Then, the equation (A.21) can be rewritten as

$$\frac{d\lambda_\sigma}{dR} = \frac{\sec^2\sigma}{F^2}\left[\left(1 + \frac{\kappa^2}{2}\right)\sin\sigma + \frac{\kappa + C}{R}\right],\ C = -C_0F^2. \tag{A.25}$$

By substituting equation (A.25) in the derivative of equation (A.2) and , we have

$$\begin{aligned}\frac{d\kappa}{dR} &= \left[\left(1 + \frac{\kappa^2}{2}\right)\sin\sigma + \frac{\kappa + C}{R}\right]\sec^2\sigma,\\ \frac{d\kappa}{dR} &= \left[\left(1 + \frac{\kappa^2}{2}\right)\sin\sigma + \frac{\kappa + C}{R}\right](1 + sin2\sigma + ...).\end{aligned} \tag{A.26}$$

Moreover, by neglecting higher-order terms of $\sin\sigma$ and $\kappa^2\sin\sigma$ of (A.26), we have

$$R\frac{d\kappa}{dR} + F^2R\sin\sigma - \kappa - C = 0. \tag{A.27}$$

From equation (A.10), we have

$$\frac{d(R\sin\sigma)}{dR} = -R\kappa. \tag{A.28}$$

Then, the equation (A.28) is substituted into equation (A.27),

$$\frac{d^2}{dR^2}(R\sin\sigma) - \frac{2}{R}\frac{d(R\sin\sigma)}{dR} - F^2R\sin\sigma + C = 0. \tag{A.29}$$

The following solution is obtained by solving equation (A.29),

$$R\sin\sigma = \frac{C}{F^2} + C_1e^{FR}(FR - 1) + C_2e^{-FR}(FR + 1). \tag{A.30}$$

At the boundary conditions $t = t_\mathrm{f}$, $R_\mathrm{f} = 0$, $\sigma_\mathrm{f} = 0$,

$$C = (C_1 + C_2)F^2 \tag{A.31}$$

Thus, equation (A.30) can be rewritten as

$$R\sin\sigma = C_1[e^{FR}(FR - 1) + 1] + C_2[e^{-FR}(FR + 1) - 1] \tag{A.32}$$

By differentiating equation (A.32) using equation (A.28), we obtain

$$\kappa = -C_1 F^2 e^{FR} + C_2 F^2 e^{-FR} \tag{A.33}$$

$$\frac{\sin\delta\cos\sigma}{F} = C_1(1 - e^{FR}) + C_2(1 - e^{-FR}) \tag{A.34}$$

where

$$C_1 = \frac{(1 - e^{-FR})R\sin\sigma + [1 - e^{-FR}(FR+1)]\sin\delta\cos\sigma/F}{e^{FR}(FR-2) - e^{-FR}(FR+2) + 4}$$
$$C_2 = \frac{(e^{FR} - 1)R\sin\sigma + [e^{FR}(FR-1) + 1]\sin\delta\cos\sigma/F}{e^{FR}(FR-2) - e^{-FR}(FR+2) + 4}$$

Then, the optimal curvature κ can be reformulated in a form of equation (A.3)

$$\kappa = \frac{K_1}{R}\sin\delta\cos\sigma - \frac{K_2}{R}\sin\sigma,$$

where

$$\begin{aligned} K_1 &= \frac{2F^2R^2 - FR(e^{FR} - e^{-FR})}{e^{FR}(FR-2) - e^{-FR}(FR+2) + 4}, \\ K_2 &= \frac{F^2R^2(e^{FR} + e^{-FR} - 2)}{e^{FR}(FR-2) - e^{-FR}(FR+2) + 4}. \end{aligned} \tag{A.35}$$

A.1.2 Vehicle with propulsion

In general, the cost function is

$$G_1 = \frac{QSC_{\mathrm{A}} - T}{mv}\left(1 + \frac{\kappa^2}{2\overline{F}}\right)\sec\sigma dR, \;\; \overline{F} = \frac{(QSC_{\mathrm{A}} - T)(T + QS(C_{\mathrm{N}\alpha} - C_{\mathrm{A}}))^2}{2m^2v^4(T/2 + QSC_{\mathrm{N}\alpha} - QSC_{\mathrm{A}}/2)}. \tag{A.36}$$

For no thrust effects of $(QSC_{\mathrm{A}} - T \geq 0)$ with $\overline{F} \geq 0$, the optimal solutions are shown previously. At the propulsive stage, $T > QSC_{\mathrm{A}}$ so $\overline{F} < 0$. Therefore, a new trajectory-shaping coefficient F_2^2 needs to be defined as

$$F_2^2 = -\overline{F} = \frac{(T - QSC_{\mathrm{A}})(T + QS(C_{\mathrm{N}\alpha} - C_{\mathrm{A}}))^2}{2m^2v^4(T/2 + QSC_{\mathrm{N}\alpha} - QSC_{\mathrm{A}}/2)}. \tag{A.37}$$

Thus, the new cost function is defined as

$$G_2 = \frac{QSC_{\mathrm{A}} - T}{mv}\left(1 - \frac{\kappa^2}{2F_2^2}\right)\sec\sigma dR. \tag{A.38}$$

The Hamiltonian to maximizing equation (A.38) is

$$H_2 = \frac{QSC_{\mathrm{A}} - T}{mv}\left(1 + \frac{\kappa^2}{2F^2}\right)\sec\sigma - \lambda_\sigma\frac{\tan\sigma}{R} - (\lambda_\sigma + \lambda_\gamma)\kappa\sec\sigma. \tag{A.39}$$

Since the Hamiltonian is independent of the variable γ, we have $\lambda_\gamma = C_0$. Thus, the optimal variable κ is described below:

$$\kappa = -\frac{mv}{QSC_{\mathrm{A}} - T}F_2^2(\lambda_\sigma + C_0). \tag{A.40}$$

From equation (A.39) and (A.40), λ_σ for the unconstrained κ is given by

$$\frac{d\lambda_\sigma}{dR} = -\frac{(QSC_{\mathrm{A}} - T)\sec^2\sigma}{mvF_2^2}\left[\left(\frac{\kappa^2}{2} + F_2^2\right)\sin\sigma + \frac{\kappa + C}{R}\right],\ C = -\frac{mv}{QSC_{\mathrm{A}} - T}C_0F_2^2 \tag{A.41}$$

By substituting equation (A.41) into the derivative of equation (A.40), we have

$$\begin{aligned}\frac{d\kappa}{dR} &= \left[\left(\frac{\kappa^2}{2} + F_2^2\right)\sin\sigma + \frac{\kappa + C}{R}\right]\sec^2\sigma,\\ \frac{d\kappa}{dR} &= \left[\left(\frac{\kappa^2}{2} + F_2^2\right)\sin\sigma + \frac{\kappa + C}{R}\right](1 + sin2\sigma + ...).\end{aligned} \tag{A.42}$$

Moreover, by neglecting higher-order terms of $\sin\sigma$ and $\kappa^2\sin\sigma$ of (A.42), we have

$$R\frac{d\kappa}{dR} - F_2^2R\sin\sigma - \kappa - C = 0. \tag{A.43}$$

Then, equation (A.28) is substituted into equation (A.43), we have

$$\frac{d^2}{dR^2}(R\sin\sigma) - \frac{2}{R}\frac{d(R\sin\sigma)}{dR} + F_2^2R\sin\sigma + C = 0. \tag{A.44}$$

By solving equation (A.44), the following solution is obtained:

$$R\sin\sigma = -\frac{C}{F_2^2} + C_1[\cos(F_2R) + F_2R\sin(F_2R)] + C_2[\sin(F_2R) - F_2R\cos(F_2R)]. \tag{A.45}$$

At the boundary conditions $t = t_{\mathrm{f}}$, $R_{\mathrm{f}} = 0$, $\sigma_{\mathrm{f}} = 0$), we obtain

$$C_1 = \frac{C}{F_2^2}. \tag{A.46}$$

Thus, equation (A.45) can be written as

$$R\sin\sigma = C_1[\cos(F_2R) + F_2R\sin(F_2R) - 1] + C_2[\sin(F_2R) - F_2R\cos(F_2R)] \tag{A.47}$$

By differentiating equation (A.47) using equation (A.28), we obtain

$$\kappa = -C_1F_2^2\cos(F_2R) - C_2F_2^2\sin(F_2R), \tag{A.48}$$

where

$$\begin{aligned}C_1 &= \frac{F_2[\cos(F_2R) - 1]R\sin\sigma - [\sin(F_2R) - F_2R\cos(F_2R)]\sin\delta\cos\sigma/F_2}{F_2[2 - 2\cos(F_2R) - F_2R\sin(F_2R)]},\\ C_2 &= \frac{F_2\sin(F_2R)R\sin\sigma + [\cos(F_2R) + F_2R\sin(F_2R) - 1]\sin\delta\cos\sigma/F_2}{F_2[2 - 2\cos(F_2R) - F_2R\sin(F_2R)]}.\end{aligned}$$

Then, the optimal curvature κ can be reformulated in a form of equation (A.3)

$$\kappa = \frac{K_1}{R}\sin\delta\cos\sigma - \frac{K_2}{R}\sin\sigma,$$

where

$$\begin{aligned}K_1 &= \frac{F_2R[\sin(F_2R) - F_2R]}{2 - 2\cos(F_2R) - F_2R\sin(F_2R)},\\ K_2 &= \frac{F_2^2R^2[1 - \cos(F_2R)]}{2 - 2\cos(F_2R) - F_2R\sin(F_2R)}.\end{aligned} \tag{A.49}$$

B Dubins' path calculation

Here, author shows the demonstration of how to find a closed form closed-form solutions of Dubins' paths. First, the aerial vehicle dynamics in the form of Dubins' model is recalled:

$$\begin{cases} x' = \cos\theta, \\ z' = \sin\theta, \\ \theta' = c. \end{cases} \tag{B.1}$$

where $'$ is the derivative with respect to curvilinear abscissa s, θ is the vehicle orientation, and c is the path curvature. The optimal control problem is to minimize the cost function $J = \int ds$ which is the shortest length. The Hamiltonian can be written as

$$H = 1 + \lambda_1 \cos(\theta) + \lambda_2 \sin(\theta) + \lambda_3 c. \tag{B.2}$$

By using the minimum principal of Pontryagin, we know that in order to have the optimal control input c^*, we have $H(c^*) \equiv 0$. Moreover, the adjoined equations of the system must verify the following conditions:

$$\begin{cases} \lambda_1' = -\dfrac{\partial H}{\partial x} = 0, \\ \lambda_2' = -\dfrac{\partial H}{\partial z} = 0, \\ \lambda_3' = -\dfrac{\partial H}{\partial \theta} = \lambda_1 \sin(\theta) - \lambda_2 \cos(\theta) = \lambda_1 z' - \lambda_2 x'. \end{cases} \tag{B.3}$$

After system (B.3), λ_1 and λ_2 are constant and

$$\lambda_3(\theta) = \lambda_3(0) + \lambda_1 \tilde{z} - \lambda_2 \tilde{x}, \tag{B.4}$$

where $\tilde{x} = x - x_0$ and $\tilde{z} = z - z_0$.

Two types of solutions must be analyzed:

1. **The solution where c is not constraint:** in this case $\frac{\partial H}{\partial c} = \lambda_3 = 0$ and $\lambda_3' = 0$. z is a linear function of x (cf. B.4). Thus, $\theta' = c = 0$. The trajectory is a segment (S).

2. **The solution where cu is constraint:** in this case $H(c^*) \leq H(c)\ \forall c\ \theta' = c = u|c|_{\max}$ with $u = -sign(\lambda_3)$. The trajectory is an arc (C).

Therefore, the solutions are in forms of the combinations of arcs (C) and segments (S). Boissonnat and his team [BCL91] have demonstrated that the optimal path between two states are of two types: CSC (arc-segment-arc) or CCC (arc-arc-arc). It is possible to calculate the trajectories of these two types by using the simple geometric.

B.1 Trajectory calculation

The equations of the trajectory of CSC or CCC paths can be written geometrically as combinations of two simple movements. Integrating the system (B.1), Δx and Δz depends on θ and the length of curvature s depends on type of the trajectory.

The trajectory of CSC path is a combination of two circular movements, where c_0 and c_1 are the curvature of the vehicle trajectory for the first and the second arcs, and a linear movement. Thus,

$$\Delta x = x_f - x_0 = \frac{u_0(\sin(\theta_1) - \sin(\theta_0))}{|c_0|_{\max}} + \cos(\theta_1)(s_2 - s_1) + \frac{u_1(\sin(\theta_f) - \sin(\theta_1))}{|c_1|_{\max}}, \quad \text{(B.5)}$$

$$\Delta z = z_f - z_0 = \frac{u_0(\cos(\theta_0) - \cos(\theta_1))}{|c_0|_{\max}} + \sin(\theta_1)(s_2 - s_1) + \frac{u_1(\cos(\theta_1) - \cos(\theta_f))}{|c_1|_{\max}}. \quad \text{(B.6)}$$

Note that $\theta_2 = \theta_1$ for the linear movement and if $\Delta s = s_2 - s_1 = 0$ and $|c_0|_{\max} = |c_1|_{\max}$, The CSC path becomes a C path. However, the trajectory of CCC path is a combination of three different circular movements. Thus,

$$\Delta x = x_f - x_0 = \frac{u(\sin(\theta_1) - \sin(\theta_0))}{|c_0|_{\max}} - \frac{u(\sin(\theta_2) - \sin(\theta_1))}{|c_1|_{\max}} + \frac{u(\sin(\theta_f) - \sin(\theta_2))}{|c_2|_{\max}}, \quad \text{(B.7)}$$

$$\Delta z = z_f - z_0 = \frac{u(\cos(\theta_0) - \cos(\theta_1))}{|c_0|_{\max}} - \frac{u(\cos(\theta_1) - \cos(\theta_2))}{|c_1|_{\max}} + \frac{u(\cos(\theta_2) - \cos(\theta_f))}{|c_2|_{\max}}. \quad \text{(B.8)}$$

θ can also be written as a function of s.

$$\theta_i = \theta_{i-1} + u_{i-1}\frac{(s_i - s_{i-1})}{|c_i|_{\max}}. \quad \text{(B.9)}$$

The trajectories can be calculated by using these equations with known values of the following variables: angle at the end of each movement θ_i and intermediate length at the end of each movement s_i. The optimal control theory is applied to find these values.

To solve this problem, all possible case must be analysed $u = \{-1, 1\}$. There are 4 possible cases for CSC paths ($u_0 = \{-1, 1\}$ and $u_1 = \{-1, 1\}$) and 2 possible cases for CCC paths ($u = \{-1, 1\}$).

B.2 CSC paths

The CSC path is completely determined by θ_1 of a segment phase. Therefore, the first step is to determine the value of this θ_1.

B.2.1 Determination of θ_1

For a segment phase, we have $\lambda_3(\theta_1) = 0$. Thus,

$$\tilde{z} = \frac{-\lambda_3(0)}{\lambda_1} + \frac{\lambda_2}{\lambda_1}\tilde{x} \text{ if } \lambda_1 \neq 0, \tag{B.10}$$

where $\tilde{z} = z - z_0$ et $\tilde{x} = x - x_0$.

The equation B.10 is in a form of linear function: $\tilde{z} = a + b\tilde{x}$. Thus,

$$\frac{\lambda_2}{\lambda_1} = \tan(\theta) \text{ if } \lambda_1 \neq 0, \tag{B.11}$$

$$\theta = \frac{\pi}{2} \text{ or} \frac{3\pi}{2} \text{ if not}, \tag{B.12}$$

where $\tan(\theta)$ is a slope of a segment.

λ can also be written as a function of θ.

$$\lambda_1 = k\cos(\theta), \tag{B.13}$$

$$\lambda_2 = k\sin(\theta), \tag{B.14}$$

where k is a constant.

By substituting (B.13) and (B.14) in (B.2 $\equiv 0$) during S phase ($\theta = \theta_1$ and $\lambda_3 = 0$), we have

$$0 \equiv 1 + k\cos(\theta_1 - \theta_1) \rightarrow k = -1. \tag{B.15}$$

Substituting also (B.13) and (B.14) in (B.2 $\equiv 0$) during C phase ($c_i = u_i|c_i|_{\max}$), we have $H = 1 - cos(\theta - \theta_1) + u_i\lambda_3|c_i|_{\max} \equiv 0$. Thus,

$$\lambda_3(\theta_f) = \frac{-u_1(1 - \cos(\theta_f - \theta_1))}{|c_1|_{\max}}, \tag{B.16}$$

$$\lambda_3(\theta_0) = \frac{-u_0(1 - \cos(\theta_1 - \theta_0))}{|c_0|_{\max}}. \tag{B.17}$$

Equations (B.13) and (B.14) are substituted in $\lambda_3(\theta_f) = \lambda_3(\theta_0) - \cos(\theta_1)\Delta z + \sin(\theta_1)\Delta x$. Thus,

$$0 = \frac{u_1(1-\cos(\theta_1-\theta_f))}{|c_1|_{\max}} - \frac{u_0(1-\cos(\theta_1-\theta_0))}{|c_0|_{\max}} + (\sin(\theta_1)\Delta x - \cos(\theta_1)\Delta z). \tag{B.18}$$

Then,

$$\begin{aligned} |c_1|_{\max}u_0 - |c_0|_{\max}u_1 = \cos(\theta_1)(-|c_0|_{\max}|c_1|_{\max}\Delta z + u_0|c_1|_{\max}\cos(\theta_0) - u_1|c_0|_{\max}\cos(\theta_f)) \\ + \sin(\theta_1)(|c_0|_{\max}|c_1|_{\max}\Delta x + u_0|c_1|_{\max}\sin(\theta_0) - u_1|c_0|_{\max}\sin(\theta_f)). \end{aligned} \tag{B.19}$$

Equation (B.19) can be reformulated as

$$A = \sqrt{A_1^2 + A_2^2}, \tag{B.20}$$

$$A_1 = -u_1|c_0|_{\max}\cos(\theta_f) + u_0|c_1|_{\max}\cos(\theta_0) - |c_0|_{\max}|c_1|_{\max}\Delta z = A\cos(\alpha), \tag{B.21}$$

$$A_2 = -u_1|c_0|_{\max}\sin(\theta_f) + u_0|c_1|_{\max}\sin(\theta_0) + |c_0|_{\max}|c_1|_{\max}\Delta x = A\sin(\alpha). \tag{B.22}$$

Thus,

$$\alpha = \arctan\left(\frac{A_2}{A_1}\right) \text{ if } A_1 \neq 0, \tag{B.23}$$

$$\alpha = \frac{\pi}{2} \text{ or } \frac{3\pi}{2} \text{ if not.} \tag{B.24}$$

We can also write

$$|c_1|_{\max}u_0 - |c_0|_{\max}u_1 = A_1\cos(\theta_1) + A_2\sin(\theta_1) = A\sin(\theta_1 - \alpha). \tag{B.25}$$

Thus,

$$\theta_1 = \alpha + \arccos\left(\frac{|c_1|_{\max}u_0 - |c_0|_{\max}u_1}{A}\right) \text{ if } A \neq 0. \tag{B.26}$$

Equation (B.26) has at least one solution if $\left|\frac{|c_1|_{\max}u_0 - |c_0|_{\max}u_1}{A}\right| < 1$ if $A \neq 0$. If $A = 0$, it means that $|c_1|_{\max}u_0 = |c_0|_{\max}u_1$ after the equation (B.25). For this condition to be true, $u_0 = u_1$ and $|c_1|_{\max} = |c_0|_{\max} = |c|_{\max}$. While substituting this condition in equations (B.21) and (B.22), we have

$$|c|_{\max}\Delta z = \cos(\theta_0) - \cos(\theta_f), \tag{B.27}$$

$$|c|_{\max}\Delta x = \sin(\theta_f) - \sin(\theta_0). \tag{B.28}$$

Equations B.27 and B.28 are substituted in equation B.5 and B.6. Thus,

$$0 = \cos(\theta_1)\Delta s,$$

$$0 = \sin(\theta_1)\Delta s.$$

That means $\Delta s = 0$ or $\cos(\theta_1) = \sin(\theta_1) = 0$. However, cos and sin cannot be equal to zero at the same time. Thus, $\Delta s = 0$. With $u_0 = u_1$, $|c_1|_{\max} = |c_0|_{\max} = |c|_{\max}$ and $\Delta s = 0$, the trajectory is in a form of a arc (C) which is a degenerated form of CSC.

$$\theta_1 = \alpha + \arccos\left(\frac{|c_1|_{\max}u_0 - |c_0|_{\max}u_1}{A}\right) \text{ if } \left|\frac{|c_1|_{\max}u_0 - |c_0|_{\max}u_1}{A}\right| < 1 \text{ et } A \neq 0, \quad \text{(B.29)}$$

$$\text{degenerated form (C) if not.} \quad \text{(B.30)}$$

B.2.2 Determination of intermediate length s_i

θ_1 is Substituted in (B.9). Thus,

$$s_1 = \frac{(\theta_1 - \theta_0)}{u_0|c_0|_{\max}}, \quad \text{(B.31)}$$

$$s_{2f} = \frac{(\theta_f - \theta_1)}{u_1|c_1|_{\max}}, \quad \text{(B.32)}$$

where s_1 is a length of curvature of the first circular movement and s_{2f} is a length of curvature of the second circular movement. Sincethe lengths of both circular movement (C) are known, the length of the linear movement (S) can be calculated by the following equations.

$$x_{12} = \Delta x - \frac{u_1}{|c_1|_{\max}}(\sin(\theta_f) - \sin(\theta_1)) - \frac{u_0}{|c_0|_{\max}}(\sin(\theta_1) - \sin(\theta_0)), \quad \text{(B.33)}$$

$$z_{12} = \Delta z - \frac{u_1}{|c_1|_{\max}}(\cos(\theta_1) - \cos(\theta_f)) - \frac{u_0}{|c_0|_{\max}}(\cos(\theta_0) - \cos(\theta_1)). \quad \text{(B.34)}$$

Thus,

$$s_{12} = \frac{z_{12}}{\sin(\theta_1)} \text{ if } \cos(\theta_1) \rightarrow 0, \quad \text{(B.35)}$$

$$s_{12} = \frac{x_{12}}{\cos(\theta_1)} \text{ if not,} \quad \text{(B.36)}$$

and

$$s_2 = s_1 + s_{12}, \quad \text{(B.37)}$$

$$s_f = s_1 + s_{12} + s_{2f}, \quad \text{(B.38)}$$

where s_2 is a summary length of the first two phases and s_f is a total length of the trajectory.

B.3 CCC paths

The CCC paths do not only depend on θ_1 but also θ_2. These equations can be developed in the same way as the CSC paths.

B.3.1 Determination of θ_1 and θ_2

Equations (B.7) and (B.8) can be reformulated as

$$A = \sqrt{A_1^2 + A_2^2},$$
$$A_1 = \frac{u\Delta x|c_0|_{\max}|c_1|_{\max}|c_2|_{\max} - |c_0|_{\max}|c_1|_{\max}\sin(\theta_f) + |c_1|_{\max}|c_2|_{\max}\sin(\theta_0)}{|c_0|_{\max}|c_2|_{\max} + |c_0|_{\max}|c_1|_{\max}} = A\sin(\alpha),$$
$$A_2 = \frac{u\Delta z|c_0|_{\max}|c_1|_{\max}|c_2|_{\max} - |c_1|_{\max}|c_2|_{\max}\cos(\theta_0) + |c_0|_{\max}|c_1|_{\max}\cos(\theta_f)}{|c_0|_{\max}|c_2|_{\max} + |c_0|_{\max}|c_1|_{\max}} = A\cos(\alpha),$$
$$B_1 = C\sin(\theta_1) - A_1,$$
$$B_2 = C\cos(\theta_1) + A_2,$$

where $C = \frac{|c_1|_{\max}|c_2|_{\max}+|c_0|_{\max}|c_2|_{\max}}{|c_0|_{\max}|c_2|_{\max}+|c_0|_{\max}|c_1|_{\max}} > 0$.

With the help of trigonometric, the equations (B.7) and (B.8) can be rewritten as

$$\sin^2(\theta_2) = B_1^2, \tag{B.39}$$
$$\cos^2(\theta_2) = B_2^2. \tag{B.40}$$

θ_2 can be found by verifying the conditions of B_1 and B_2:

- **if** $B_1 < 1$: $\theta_2 = \arcsin(B_1)$
- **if** $B_2 < 1$: $\theta_2 = \arccos(B_1)$
- **if** $B_2 \geqslant 1$ **and** $B_2 \geqslant 1$: no solution

Then, α can be found by solving A_1 and A_2.

$$\alpha = \arctan\left(\frac{A_1}{A_2}\right) \text{ if } A_2 \neq 0, \tag{B.41}$$
$$\alpha = \frac{\pi}{2} \text{ or } \frac{3\pi}{2} \text{ if not.} \tag{B.42}$$

Equations (B.39) and (B.40) can also be rewritten as

$$\begin{aligned} 1 &= (C\sin(\theta_1) - A_1)^2 + (C\cos(\theta_1) + A_2)^2, \\ &= C^2 - 2A_1C\sin(\theta_1) + 2A_2C\cos(\theta_1) + A_1^2 + A_2^2, \\ \frac{1 - C^2 - A^2}{2AC} &= \cos(\alpha)\cos(\theta_1) - \sin(\alpha)\sin(\theta_1), \\ &= \cos(\theta_1 + \alpha). \end{aligned}$$

Thus,

$$\theta_1 = \arccos\left(\frac{1 - C^2 - A^2}{2AC}\right) - \alpha \text{ if } 0 < |1 - C^2 - A^2| < |2AC| \text{ and } |A| \neq 0, \tag{B.43}$$
$$\text{no solution if not.} \tag{B.44}$$

B.3.2 Determination of intermediate length s_i

θ_1 and θ_2 are substituted in equation (B.9). Thus,

$$s_1 = u\frac{\theta_1 - \theta_0}{|c_0|_{\max}}, \tag{B.45}$$

$$s_{12} = -u\frac{\theta_2 - \theta_1}{|c_1|_{\max}}, \tag{B.46}$$

$$s_{2f} = u\frac{\theta_f - \theta_2}{|c_2|_{\max}}. \tag{B.47}$$

Thus,

$$s_2 = s_1 + s_{12} = u(|c_0|_{\max}(\theta_1 - \theta_0) - |c_1|_{\max}(\theta_2 - \theta_1)), \tag{B.48}$$

$$s_f = s_2 + s_{2f} = u(|c_0|_{\max}(\theta_1 - \theta_0) - |c_1|_{\max}(\theta_2 - \theta_1) + |c_2|_{\max}(\theta_f - \theta_2)). \tag{B.49}$$

B.4 CS paths

B.4.1 Determination of θ_1

All the required conditions are the same as the calculation of the CSC paths except that $\lambda_3(\theta_f) = 0$ because there is no change of θ after the second phase (linear movement). Thus, we can simplify the equation (B.19) as

$$-u_0(1 - \cos(\theta_1 - \theta_0)) + |c_0|_{\max}(\sin(\theta_1)\Delta x - \cos(\theta_1)\Delta z) = 0. \tag{B.50}$$

Suppose that

$$A = \sqrt{A_1^2 + A_2^2}, \tag{B.51}$$

$$A_1 = u_0\cos(\theta_0) - |c_0|_{\max}\Delta z = A\cos(\alpha), \tag{B.52}$$

$$A_2 = u_0\sin(\theta_0) + |c_0|_{\max}\Delta x = A\sin(\alpha). \tag{B.53}$$

Then,

$$\alpha = \arctan\left(\frac{A_2}{A_1}\right) \text{ if } A_1 \neq 0, \tag{B.54}$$

$$\alpha = \frac{\pi}{2} \text{ or } \frac{3\pi}{2} \text{ if not.} \tag{B.55}$$

Equation (B.50) can also be written as

$$u_0 = A_1\cos(\theta_1) + A_2\sin(\theta_1). \tag{B.56}$$

Thus, θ_1 can be deduced as

$$\theta_1 = \alpha + \arccos\left(\frac{u_0}{A}\right) \text{ if } \left|\frac{u_0}{A}\right| < 1 \text{ and } A \neq 0, \tag{B.57}$$

$$\text{no movement if } A = 0, \tag{B.58}$$

$$\text{no solution if } \left|\frac{u_0}{A}\right| > 1. \tag{B.59}$$

The justification for the equation (B.58) is that, if $A = 0 \rightarrow A_1 = A_2 = 0$, then $u_0 = 0$ according to the equation (B.56) and $\Delta z = \Delta x = 0$ after the equations (B.52) and (B.53). There is no movement as a consequence.

B.4.2 Determination of intermediate length s_i

θ_1 is substituted in (B.9). Thus,

$$s_1 = u_0 \frac{\theta_1 - \theta_0}{|c_0|_{\max}}, \tag{B.60}$$

where s_1 is a length between of curvature of the first and only circular movement. Since we know the length of the circular movement (C), the length of the linear movement (S) can be calculated by the following equations.

$$x_{12} = \Delta x - \frac{u_0}{|c_0|_{\max}}(\sin(\theta_1) - \sin(\theta_0)), \tag{B.61}$$

$$z_{12} = \Delta z - \frac{u_0}{|c_0|_{\max}}(\cos(\theta_0) - \cos(\theta_1)). \tag{B.62}$$

Thus,

$$s_{12} = \frac{z_{12}}{\sin(\theta_1)} \text{ if } \cos(\theta_1) \rightarrow 0, \tag{B.63}$$

$$s_{12} = \frac{x_{12}}{\cos(\theta_1)} \text{ if not.} \tag{B.64}$$

and

$$s_2 = s_1 + s_{12}. \tag{B.65}$$

where s_2 is a total length of the trajectory.

C Usage of APF as a preprocessing method of exploration space

Artificial potential field is one of the interesting path planning algorithms. It is a reactive method that guides the vehicle toward the destination while avoiding obstacles. By assuming that the APF can find trajectories to a destination. Here, author explain the usage of the APF as a preprocessing method of exploration space and how to calculate the artificial potential functions adapted to the vehicle system.

C.1 Brief description of APF

The idea of the APF is taken from nature. For instance, a charged particle navigating a magnetic field, or a small ball rolling in a hill. The idea is that depending on the strength of the field, or the slope of the hill, the particle, or the ball can arrive to the source of the field, the magnet, or the valley. Based on this idea, the APF is first introduced [Kha85].

In APF approach, the same effect as in nature can be simulated by creating an APF that will attract the vehicle to the goal. This field is called an attractive filed $\varphi_{\text{goal}}(\mathbf{x})$. The potential field is defined across the entire free space $\mathbb{X}_{\text{free}}$, and in each time step, the potential field is simulated at the vehicle position, and the induced force by this field is calculated. Then, the vehicle moves according to this force.

In environment cluttered by obstacles, another behavior can also be defined. The simplest way to avoid the obstacles is to generated a repulsive field $\varphi_{\text{obs}}(\mathbf{x})$ around it. Then, if the vehicle approaches the obstacle, a repulsive force will act on it. This results in the vehicle is pushed away from the obstacle. Moreover, the initial position of the vehicle can

act as an obstacle since the vehicle has to move away from it. Thus, a potential field $\varphi_{\text{init}}(\mathbf{x})$ is created.

These two behaviors, seeking and avoiding specific locations, can be combined. An particular artificial potential field $\varphi_{\text{art}}(\mathbf{x})$ can be obtained as

$$\varphi_{\text{art}}(\mathbf{x}) = \varphi_{\text{goal}}(\mathbf{x}) + \Sigma\varphi_{\text{obs}}(\mathbf{x}). \tag{C.1}$$

In case of no obstacle, we obtain

$$\varphi_{\text{art}}(\mathbf{x}) = \varphi_{\text{goal}}(\mathbf{x}) + \varphi_{\text{init}}(\mathbf{x}). \tag{C.2}$$

In the next section, the preprocessing of exploration space using the APF based on the equation (C.2) is presented.

C.2 Preprocessing of exploration space using the APF

The main problem of this method is how to choose and define a type and a nature of an artificial potential field while respecting the system dynamics and constraints. In this section, some mathematical forms of the APF are analyzed.

C.2.1 APF in quadratic form $\varphi_i = K_i r_i^n$

This form of potential field is mostly used in mobile robotics. It is often used for both attractive field and repulsive field around a point (ex: destination, etc.) or round object (ex: obstacle).

Suppose that the relation between r_i and (x_i, z_i) can be written as

$$\begin{aligned} x - x_i &= r_i \cos\theta_i \\ z - z_i &= r_i \sin\theta_i \end{aligned} \tag{C.3}$$

where θ_i is the angle between the axis x and r_i. Thus, we have

$$\begin{aligned} \varphi_x &= \frac{\partial}{\partial x}\varphi = nK_i \cos\theta_i r_i^{n-1} \\ \varphi_z &= \frac{\partial}{\partial z}\varphi = nK_i \sin\theta_i r_i^{n-1} \end{aligned} \tag{C.4}$$

If the equation (C.2) is considered, then, we have the ratio between the gain of the attractive field K_2 and the gain of the repulsive field K_1 as

$$\beta = \frac{K_2}{K_1} = \frac{r_f \kappa_0}{2} \frac{1}{\sin(\theta_0 + \theta_f)}. \tag{C.5}$$

This form of potential field can be divided into 3 cases:

1. $n = 0$: it leads to no force due to the potential field at all *ie.* $\frac{\partial}{\partial x_i}\varphi = 0$. Thus, this case is not interested here.

2. $n > 0$: we can clearly see that the equations ((C.4)) depends on r_i^{n-1}. The force due to the potential field is constant if $n = 1$ and increases with respect to r_i if $n > 1$. Moreover, for $n \in (0, 1)$, the force decreases with respect to r_i.

3. $n < 0$: in this case, the force due to the potential field decreases with respect to r_i.

By considering the nature of vehicle movements, case 3 is the most interesting since the forces are strong near the center of potential fields and weak far from it. Thus, the potential field in the exploration space can be expressed as

$$\varphi = K_1 r_0^n - K_2 r_f^n, \quad \text{for} \quad n < 0 \tag{C.6}$$

where r_0 is the distance from the initial origin (x_0, z_0) and r_f is the distance from the goal origin (x_f, z_f).

C.2.1.1 Calculation of the ratio $n = K_2/K_1$

By using equation (C.6) and (C.3), the partial derivatives of φ can be express as

$$\varphi_x = \frac{\partial}{\partial x}\varphi = nK_1 \cos\theta_0 r_0^{n-1} - nK_2 \cos\theta_f r_f^{n-1}, \tag{C.7}$$

$$\varphi_z = \frac{\partial}{\partial z}\varphi = nK_1 \sin\theta_0 r_0^{n-1} - nK_2 \sin\theta_f r_f^{n-1}, \tag{C.8}$$

$$\varphi_{xx} = \frac{\partial}{\partial x}\varphi_x = nK_1 r_0^{n-2}(1 + (n-2)\cos^2\theta_0) - nK_2 r_f^{n-2}(1 + (n-2)\cos^2\theta_f), \tag{C.9}$$

$$\varphi_{zz} = \frac{\partial}{\partial z}\varphi_z = nK_1 r_0^{n-2}(1 + (n-2)\sin^2\theta_0) - nK_2 r_f^{n-2}(1 + (n-2)\sin^2\theta_f), \tag{C.10}$$

$$\varphi_{xz} = \frac{\partial}{\partial x}\varphi_z = n(n-2)K_1 \cos\theta_0 \sin\theta_0 r_0^{n-2} - n(n-2)K_2 \cos\theta_f \sin\theta_f r_f^{n-2} = \varphi_{zx}. \tag{C.11}$$

The stream line curvature of this potential function can be found by using the formulation

$$\kappa = \frac{(\varphi_x^2 - \varphi_z^2)\varphi_{xz} + \varphi_x\varphi_z(\varphi_{zz} - \varphi_{xx})}{(\varphi_x^2 + \varphi_z^2)^{3/2}}. \tag{C.12}$$

Thus,

$$\kappa = \frac{Ar_0^{2n-3}r_f^{n-1} + Br_0^{2n-2}r_f^{n-2} + C2r_0^{n-2}r_f^{2n-2} + Dr_0^{n-1}r_f^{2n-3}}{n^3(K_1^2 r_0^{2n-2} - 2K_1K_2\cos(\theta_0 + \theta_f)r_0^{n-1}r_f^{n-1} + K_2^2 r_f^{2n-2})^{3/2}}, \tag{C.13}$$

with

$$A = n^3(n-2)K_1^2K_2(\cos(2\theta_0)\sin(\theta_0 + \theta_f) - \sin(2\theta_0)\cos(\theta_0 - \theta_f)),$$

$$B = \frac{n^3(n-2)K_1^2K_2}{2}\sin(2(\theta_0 - \theta_f)),$$

$$C = \frac{n^3(n-2)K_1K_2^2}{2}\sin(2(\theta_0 - \theta_f)),$$

$$D = n^3(n-2)K_1K_2^2(\sin(2\theta_f)\cos(\theta_0 - \theta_f) - \cos(2\theta_f)\sin(\theta_0 + \theta_f)).$$

With equation (C.13), we can deduce that $\kappa = 0$ if $r_0 = 0$ or $r_f = 0$. Its means that the value of β cannot be reduced at the center of potential fields. Thus, this type of potential field is not adaptable to our problem where the curvature is definitely known at $r_0 = 0$ and $r_f = 0$. An APF in another mathematical form is then proposed.

C.2.2 APF in logarithm form $\varphi_i = K_i \ln r_i$

C.2.2.1 Calculation of the ratio $\beta = K_2/K_1$

A potential field in the exploration space can be expressed by

$$\varphi = K_1 \ln r_0 - K_2 \ln r_f, \tag{C.14}$$

where K_1, K_2 are gains of potential fields, r_0 is the distance from the origin (x_0, z_0) and r_f is the distance from the origin (x_f, z_f).

By using equation (C.14) the partial derivatives of φ can be express as

$$\varphi_x = \frac{\partial}{\partial x}\varphi = \frac{K_1 \cos\theta_0}{r_0} - \frac{K_2 \cos\theta_f}{r_f} \tag{C.15}$$

$$\varphi_z = \frac{\partial}{\partial z}\varphi = \frac{K_1 \sin\theta_0}{r_0} - \frac{K_2 \sin\theta_f}{r_f} \tag{C.16}$$

$$\varphi_{xx} = \frac{\partial}{\partial x}\varphi_x = \frac{K_1}{r_0^2}(1 - 2\cos^2\theta_0) - \frac{K_2}{r_f^2}(1 - 2\cos^2\theta_f) \tag{C.17}$$

$$\varphi_{zz} = \frac{\partial}{\partial z}\varphi_z = \frac{K_1}{r_0^2}(1 - 2\sin^2\theta_0) - \frac{K_2}{r_f^2}(1 - 2\sin^2\theta_f) \tag{C.18}$$

$$\varphi_{xz} = \frac{\partial}{\partial x}\varphi_z = -\frac{K_1 \sin(2\theta_0)}{r_0^2} + \frac{K_2 \sin(2\theta_f)}{r_f^2} = \varphi_{zx} \tag{C.19}$$

By using equation (C.12), the stream line curvature of this potential function is

$$\kappa = \frac{A r_f^{-1} r_0^{-3} + B r_f^{-2} r_0^{-2} + C r_f^{-3} r_0^{-1}}{(K1^2 r_0^{-2} - 2K_1 K_2 \cos(\theta_0 - \theta_f) r_f^{-1} r_0^{-1} + K_2^2 r_f^{-2})^{3/2}}, \tag{C.20}$$

with

$$\begin{aligned} A &= 2K_1^2 K_2 \sin(\theta_0 - \theta_f), \\ B &= K_1 K_2 (K_1 + K_2) \sin(2\theta_f - 2\theta_0), \\ C &= 2K_1 K_2^2 \sin(\theta_0 - \theta_f). \end{aligned}$$

The path curvature is known at the origin (x_f, z_f), $\kappa = \kappa_f$,where $r_f = 0$, $\theta_0 \approx \arctan((z_f - z_i)/(x_f - x_i))$ and θ_f is the decided angle of attack at (x_f, z_f). Therefore, (C.20) can be rewritten as

$$\kappa_f = \frac{A}{K_2^3} = \frac{2K_1}{r_0 K_2} \sin(\theta_0 - \theta_f). \tag{C.21}$$

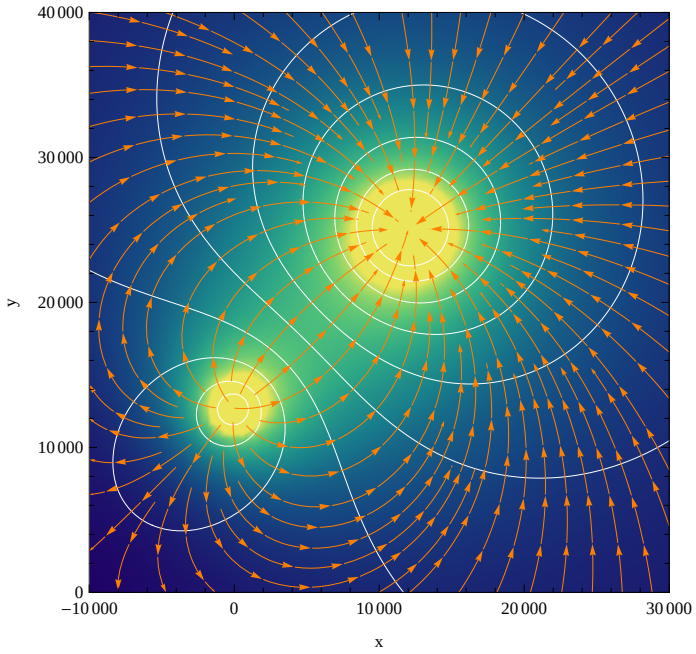


Figure C.1: Potential field respecting the decided curvature κ_f at (x_f, z_f)

Thus, the ratio $\beta = K_2/K_1$ can be found as

$$\beta = \frac{K_2}{K_1} = \frac{2}{r_0 \kappa_f} \sin(\theta_0 - \theta_f). \tag{C.22}$$

Figure C.1 shows the potential field which respects the decided curvature $\kappa_f = 4.4742 \times 10^{-5}$ at $(x_f, z_f) = (12000, 25000)$. The numerical value of n given by (C.20) is equal to 1.862.

The same calculation can be applied at the other origin (x_0, z_0), where $r_0 = 0$, $\theta_0 \approx \arctan((z_i - z_f)/(x_i - x_f))$ and θ_0 is the angle of attack at (x_0, z_0). We have

$$\kappa_0 = \frac{C}{K_1^3} = \frac{2K_2}{r_f K_1} \sin(\theta_0 - \theta_f). \tag{C.23}$$

Thus,

$$\beta = \frac{K_2}{K_1} = \frac{r_f \kappa_0}{2} \frac{1}{\sin(\theta_0 + \theta_f)} \tag{C.24}$$

Figure C.2 shows the potential field which respects the decided curvature $\kappa_0 = 3.1225 \times 10^{-4}$ at $(x_0, z_0) = (0, 12705)$. The numerical value of n given by (C.20) is equal to 2.3627.

C.2.2.2 Trajectory calculation

Since all the unknown variables of the potential function are found. Equations (C.15) and (C.16) can be rewritten in (x, z) coordinate as:

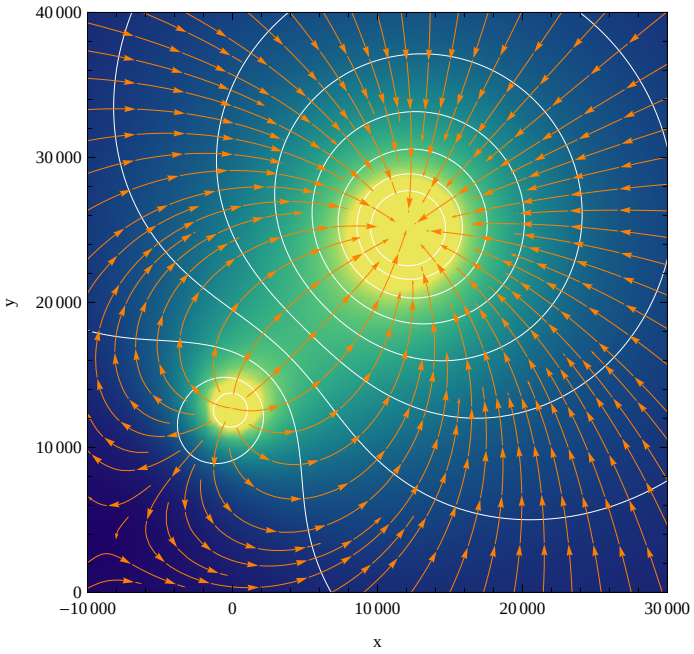


Figure C.2: Potential field respecting the decided curvature κ_0 at (x_0, z_0)

$$\varphi_x = \frac{2K_1(x - x_0)}{(x - x_0)^2 + (z - z_0)^2} - \frac{2nK_1(x - x_f)}{(x - x_f)^2 + (z - z_f)^2}, \tag{C.25}$$

$$\varphi_z = \frac{2K_1(z - z_0)}{(x - x_0)^2 + (z - z_0)^2} - \frac{2nK_1(z - z_f)}{(x - x_f)^2 + (z - z_f)^2}, \tag{C.26}$$

where (φ_x, φ_z) represents the potential forces in (x, z) axis due to the potential function φ. Thus, we have

$$\frac{dy}{dx} = \frac{\varphi_z}{\varphi_x} = \frac{(z - z_0)((x - x_f)^2 + (z - z_f)^2) - n(z - z_f)((x - x_0)^2 + (z - z_0)^2)}{(x - x_0)((x - x_f)^2 + (z - z_f)^2) - n(x - x_f)((x - x_0)^2 + (z - z_0)^2)}. \tag{C.27}$$

By solving (C.27), we have

$$n \arctan((z - z_f)/(x - x_f)) + \arctan((x - x_0)/(z - z_0)) = \text{Constant} = C_1. \tag{C.28}$$

C_1 can be found by replacing (x, z) by a point where we want to find a pass-by-point trajectory.

C.2.2.3 Idea for implementation

The potential fields found in the previous section respect only the path curvature at one origin of the potential field (either (x_0, z_0) or (x_f, z_f). In order to respect the curvatures at

both origins, the inverse distance weighting is used. Thus, the total APF can be obtained as

$$\varphi(x,z) = \frac{\varphi_0(r_0)r_f + \varphi_f(r_f)r_0}{r_0 + r_f},$$

where r_i is the distance between (x,z) and (x_i,z_i), $\varphi_i(r)$ is the potential field which respects the curvature at the origin (x_i,z_i) at distance r.

Note that this method requires a lot of parameter tuning. Moreover, there is no proof that the trajectory obtained by using this method respect all the path curvature along the trajectory since it is only proved that the path curvature at the origins are respected.

D Dubins' curves in a heterogeneous environment

The Dubins' curves are originally calculated by supposing that the curvature is constant along the curve; however, that is not the case of the missile who flies in the 2D vertical plane. The system considered is the same as the Dubins' car except that the curvature decreases exponentially with altitude. This is due to the fact that the air density decreases exponentially with altitude. In [HP13], it was shown that, analogously to Dubins' paths [Dub57][BCL91], shortest paths are a combination of curves of maximum curvature C and straight lines S, *i.e.* CSC or CCC path. In [SL01], it is proven that CSC path and not CCC path is the shortest path if two states are sufficiently far from each other. Thus, only CSC paths are considered, here, under the assumption that two vehicle states are sufficiently far from each other. This appendix is a theoretical recall of [HP13].

D.1 Dubins-like model

The calculation of the Dubins' paths is based on the simplified dynamics of aerial vehicles. It is modeled as:

$$\begin{aligned} x' &= \frac{dx}{ds} = \cos\theta, \\ z' &= \frac{dz}{ds} = \sin\theta, \\ \theta' &= \frac{d\theta}{ds} = c(z)u \ \text{ where } \ u \in [-1, 1], \end{aligned} \tag{D.1}$$

where θ is the orientation of the vehicle, $c(z)$ is the maximum path curvature depending on altitude z, and u is the control input.

Assuming that the vehicle velocity v is high enough in order to neglect the gravitational force during the flight. Considering a non-propulsive stage of the vehicle, the lift force f_L is the only aerodynamic force who contributes to maneuver the missile. It can be written as follow:

$$f_L = \frac{1}{2}\rho(z)SC_L v^2$$

where $\rho(z)$ is the air density depending on the current altitude z, S is the area of reference and C_L is the lift coefficient depending on the angle of attack. The maximum path curvature is characterized by lift force, *i.e.* $c(z) = \frac{1}{2m}\rho(z)SC_{L_{\max}}$. The air density can be expressed by:

$$\rho(z) = \rho_0 e^{-z/z_r}. \tag{D.2}$$

where ρ_0 is the air density at the standard atmosphere at sea level and z_r is the reference altitude. Then, the maximum path curvature can be expressed as:

$$c(z) = c_0 e^{-z/z_r}. \tag{D.3}$$

where c_0 is the maximum path curvature at the standard atmosphere at sea level.

D.2 Computation of the curve C in a plane

In order to derive curves of maximum curvature, the magnitude of the control input u in system (D.1) is set to a constant 1 or -1. By differentiating θ' with respect to s, we obtain

$$\theta'' = -\frac{\cos\phi}{z_r}\theta' \sin\theta. \tag{D.4}$$

Define $\zeta = \tan\left(\frac{\theta}{2}\right)$. After some straightforward trigonometry, we have

$$\cos^2\theta = \frac{1-\zeta^2}{1+\zeta^2}, \tag{D.5}$$

$$\theta' = 2\frac{\zeta'}{1+\zeta^2}. \tag{D.6}$$

By integrating equation (D.4) and applying some trigonometric techniques, we have

$$\theta' = \frac{\cos\phi}{z_r}\left(\frac{z_r}{\cos\phi}\theta'_0 - \cos\theta_0 + 1 - 2\cos^2\left(\frac{\theta}{2}\right)\right). \tag{D.7}$$

With equations (D.5), (D.6) and (D.7), we obtain

$$\begin{aligned} \zeta' &= A + B\zeta^2, \\ A &= \frac{\cos\phi}{2z_r}\left(\frac{z_r}{\cos\phi}\theta'_0 - \cos\theta_0 + 1\right), \\ B &= A - \frac{\cos\phi}{z_r}. \end{aligned} \tag{D.8}$$

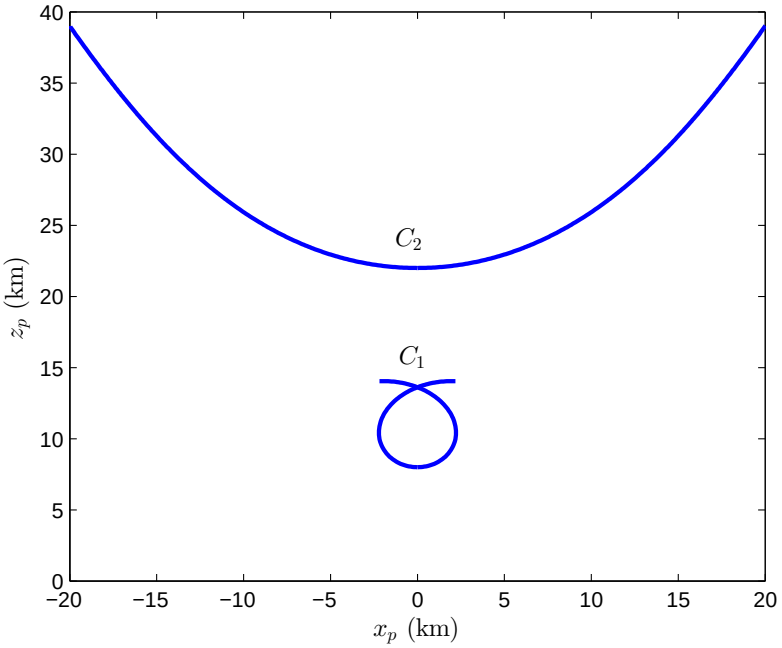


Figure D.1: Examples of arcs of maximum curvature

According to system (D.8), there are four types of curves depending on the values of A and B:

- C_1 curve if $AB > 0$,

$$\zeta_1(s) = \sqrt{\frac{A}{B}} \tan\left[A\sqrt{\frac{B}{A}}s + \arctan\left(\sqrt{\frac{B}{A}}\zeta_0\right)\right]. \tag{D.9}$$

  The C_1 curve is illustrated in figure D.1.

- C_2 curve if $AB < 0$,

$$\zeta_2(s) = \sqrt{\left|\frac{A}{B}\right|} \tanh\left[A\sqrt{\left|\frac{B}{A}\right|}s + \operatorname{arctanh}\left(\sqrt{\left|\frac{B}{A}\right|}\zeta_0\right)\right]. \tag{D.10}$$

  The C_2 curve is also illustrated in figure D.1. This curve has oblique asymptotes, *i.e.* $\zeta_2 \in \left[-\sqrt{\left|\frac{A}{B}\right|}, \sqrt{\left|\frac{A}{B}\right|}\right]$. This condition must be verified for both ζ_0 and $\zeta_2(s)$.

- C_3 curve if $A = 0$,

$$\frac{1}{\zeta_3(s)} = \frac{1}{\zeta_0} - Bs. \tag{D.11}$$

- C_4 curve if $B = 0$,

$$\zeta_4(s) = \zeta_0 + As. \tag{D.12}$$

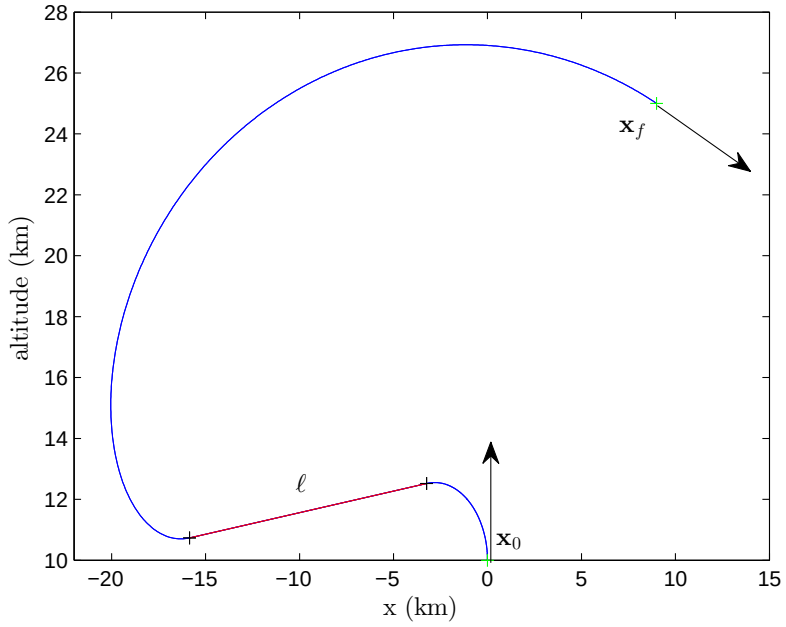


Figure D.2: Dubins' path in a heterogeneous environment

Remark 8 *C_3 and C_4 curves are the extremal cases of the first two curves. They are rarely obtained in reality. Thus, no illustration of these curves is presented in this paper.*

θ, x, and z can be derived as functions of $\zeta(s)$ as follows:

$$\begin{cases} \theta_p(\zeta) = 2\arctan\zeta + k(s)\pi, \\ x(\zeta) = \dfrac{z_r}{\cos\phi}(\theta(\zeta) - \theta_0) - \dfrac{z_r}{\cos\phi}(A+B)s, \\ z(\zeta) = -\dfrac{z_r}{\cos\phi}\ln\left(\dfrac{1+\zeta_0^2}{A+B\zeta_0^2}\dfrac{A+B\zeta^2}{1+\zeta^2}\right), \end{cases} \tag{D.13}$$

where $k(s)$ is an integer depending on the distance s. For C_1 curve, $k(s)$ is calculated as

$$k(s) = \left\lfloor sA\sqrt{\frac{B}{A}} \Big/ \left(u\frac{\pi}{2} - \arctan\left(\sqrt{\frac{B}{A}}\zeta_0\right)\right)\right\rfloor, \; u = \pm 1 \tag{D.14}$$

where $\lfloor\boldsymbol{.}\rfloor$ is a floor division. Otherwise, $k(s) = 0$ for C_2 curve. However, for the optimal solution $k(s)$ value is never greater than 1, *i.e.* $k(s) > 1$ means that the vehicle starts to turn in loop.

D.3 CSC path generation

Once two curves have been found, a solver is used to find a line segment $\boldsymbol{\ell}$ connecting both curves (see figure D.2) by verifying the objective function $F(\boldsymbol{\ell}) = \boldsymbol{\ell} - (\boldsymbol{\xi}_2 - \boldsymbol{\xi}_1) = \mathbf{0}$.

Remark 9 *With this methodology, the conditions* $\boldsymbol{\ell} \times \mathbf{v}_1 = \mathbf{0}$ *and* $\boldsymbol{\ell} \times \mathbf{v}_2 = \mathbf{0}$ *are automatically verified.*

There can exist four types of CSC paths where $u = \pm 1$ for both circular arcs. Figure D.2 shows the case that there exists only one of the four CSC paths. The solution can be found in the same way as the demonstration.

E Estimation of attacking velocity

Here, velocity estimation along the Dubins' paths is described. The Dubins' paths are combination of arcs of circle (C) and line segments (S). The velocity will be estimated step by step according to path type, *i.e.* C or S.

The dynamic model along the principal axis of the movement $\mathbf{e}_1^{\mathrm{v}}$ is described below

$$\dot{v} = \frac{T\cos\alpha}{m(t)} - \frac{1}{2m(t)}\rho(z)SC_{\mathrm{D}}(\alpha)v^2 \tag{E.1}$$

$$v\dot{\gamma} = \frac{T\sin\alpha}{m(t)} + \frac{1}{2m(t)}\rho(z)SC_{\mathrm{L}}(\alpha)v^2 \tag{E.2}$$

where $T = ||\mathbf{f}_{\mathrm{thrust}}||$ is the thrust force, v is the vehicle speed, $m(t) = m_0 - qt$ is the time-varying mass with the initial mass m_0 and the mass flow q, ρ is the air density which depends on the altitude h, S is the area of reference, and C_{D} and C_{L} are the aerodynamic coefficients which depend on the angle of attack α.

The estimation of the attacking velocity is done while neglecting the gravity. Since the vehicle can activate the propulsion or deactivate the propulsion, the estimation is calculated separately according to its propulsion stage for each type of trajectories.

E.1 Estimation of a linear trajectory

E.1.1 Non-propulsive stage

Since there is no propulsive force, the vehicle mass is constant. The, equation (E.1) can be rewritten as

$$\dot{v} = -\frac{1}{2m}\rho(z)SC_{\mathrm{D}}(\alpha)v^2. \tag{E.3}$$

Then, the estimation of the velocity is calculated based on the following hypothesis:

Hypothesis 3 *The drag coefficient $C_D = C_D(\alpha_0)$ is constant along the linear trajectory with $\alpha = \alpha_0 = 0$ because there is no control input.*

Moreover, the simplified environment model is used:

$$\rho(z) = \rho_0 e^{-z/z_r}, \tag{E.4}$$

where ρ_0 is the air density at the sea level and z_r is the reference altitude.

By developing equation (E.3), we have

$$\begin{aligned} -\int_{v_0}^{v_f} \frac{1}{v}dv &= \int_{t_0}^{t_f} \frac{1}{2m}\rho_0 e^{-z/z_r} SvC_{\mathrm{D}} v dt, \\ &= \int_{s_0}^{s_f} \frac{1}{2m}\rho_0 e^{-z/z_r} SvC_{\mathrm{D}} ds, \end{aligned} \tag{E.5}$$

where $ds = vdt$ is a curvature length, in this case, a linear curvature. Thus, the relation between h and s can be written as follow:

$$\sin\theta ds = dz, \tag{E.6}$$

where θ is the pitch angle.

By substituting (E.6) in (E.5), we have

$$-\int_{v_0}^{v_f} \frac{1}{v}dv = \frac{1}{2m\sin\theta} SvC_{\mathrm{D}} \int_{z_0}^{z_f} \rho_0 e^{-z/z_r} dz. \tag{E.7}$$

Thus,

$$v_f = v_0 e^{A(z)}, \tag{E.8}$$

where $A(z) = \frac{z_r \rho_0 SC_{\mathrm{D}}}{2m\sin\theta}(e^{-z_0/z_r} - e^{-z_f/z_r})$ for $\sin\theta \neq 0$. If $\sin\theta = 0$, the equation (E.5) is integrated directly with $\rho = \rho_{\mathrm{cst}} = \rho_0 e^{h_{\mathrm{i}}/H} = \text{constant}$. Thus,

$$v_f = v_0 e^{-\frac{1}{2m}\rho_{\mathrm{cst}} SC_{\mathrm{D}} \Delta s} \text{ if } \sin\theta = 0 \tag{E.9}$$

where Δs is a length of curvature.

E.1.2 Propulsion phase

In this case, the dynamics of the velocity is expressed as

$$\dot{v} = \frac{T}{m(t)} - \frac{1}{2m(t)}\rho(z)SC_{\mathrm{D}}v^2. \tag{E.10}$$

By substituting $\rho(z) = \rho_{\mathrm{av}} = \frac{\int_{z_0}^{z_f}\rho(z)dz}{z_f - z_0}$ into equation (E.10) and solving it, we have

$$v(t) = \sqrt{\frac{2T}{\rho_{\mathrm{av}}SC_{\mathrm{D}}}}\tanh\left(\sqrt{0.5T\rho_{\mathrm{av}}SC_{\mathrm{D}}}\left(A - \frac{\ln(m_0 - qt_0)}{q}\right)\right). \tag{E.11}$$

For the boundary condition $v(t=0) = v_0$, the constant A can be found:

$$A = \frac{\ln\left(\frac{1+v_0\sqrt{\rho_{\mathrm{av}}SC_{\mathrm{D}}/2T}}{1-v_0\sqrt{\rho_{\mathrm{av}}SC_{\mathrm{D}}/2T}}\right)}{\sqrt{2T\rho_{\mathrm{av}}SC_{\mathrm{D}}}} + \frac{\ln(m_0 - qt_f)}{q}. \tag{E.12}$$

By substituting (E.12) in (E.11), we have

$$\begin{aligned} v(t) &= \sqrt{\frac{2T}{\rho_{\mathrm{av}}SC_{\mathrm{D}}}}\tanh\left(\frac{1}{2}\ln\left(\frac{1+v_0\sqrt{\rho_{\mathrm{av}}SC_{\mathrm{D}}/2T}}{1-v_0\sqrt{\rho_{\mathrm{av}}SC_{\mathrm{D}}/2T}}\right) + \sqrt{\frac{T\rho_{\mathrm{av}}SC_{\mathrm{D}}}{2}}\frac{\ln\left(\frac{m_0-qt_f}{m_0-qt_0}\right)}{q}\right), \\ &= \sqrt{\frac{2T}{\rho_{\mathrm{av}}SC_{\mathrm{D}}}}\frac{\sqrt{\frac{1+v_0\sqrt{\rho_{\mathrm{av}}SC_{\mathrm{D}}/2T}}{1-v_0\sqrt{\rho_{\mathrm{av}}SC_{\mathrm{D}}/2T}}}\left(\frac{m_0-qt_f}{m_0-qt_0}\right)^{\sqrt{0.5T\rho_{\mathrm{av}}SC_{\mathrm{D}}}/q} - 1}{\sqrt{\frac{1+v_0\sqrt{\rho_{\mathrm{av}}SC_{\mathrm{D}}/2T}}{1-v_0\sqrt{\rho_{\mathrm{av}}SC_{\mathrm{D}}/2T}}}\left(\frac{m_0-qt_f}{m_0-qt_0}\right)^{\sqrt{0.5T\rho_{\mathrm{av}}SC_{\mathrm{D}}}/q} + 1}. \end{aligned} \tag{E.13}$$

E.2 Estimation of a circular trajectory

E.2.1 Non-propulsive phase

With constant mass, equation (E.2) can be rewritten as

$$\dot{\gamma} = \frac{1}{2m}\rho(z)SC_{\mathrm{L}}(\alpha)v, \tag{E.14}$$

where C_{L} is the lift coefficient.

We obtain an estimated final velocity by solving (E.3) and (E.14):

$$\dot{\gamma} = -\frac{C_{\mathrm{L}}(\alpha)}{C_{\mathrm{D}}(\alpha)}\frac{\dot{v}}{v} = -f\frac{\dot{v}}{v}, \tag{E.15}$$

where f is called the lift-to-drag ratio.

By integrating equation (E.15), we obtain

$$v_f = v_0 e^{-(\gamma_f - \gamma_0)/f}, \tag{E.16}$$

where γ_0 is the initial flight path angle and γ_f is the final flight path angle.

E.2.2 Propulsive phase

In this case we suppose that the propulsion effect only the dynamics of the velocity. Thus, the dynamic model can be rewritten as

$$\dot{v} = \frac{T}{m(t)} - \frac{1}{2m(t)}\rho(z)Sv^2C_{\mathrm{D}}(\alpha), \tag{E.17}$$

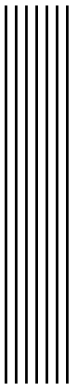
$$\dot{\gamma} = \frac{1}{2m(t)}\rho(z)SC_{\mathrm{L}}(\alpha)v. \tag{E.18}$$

By solving equations (E.17) and (E.18), we have

$$\dot{v} = -\frac{\dot{\gamma}}{f}v + \frac{T}{m_0 - qt}. \tag{E.19}$$

Integrate equation (E.19) by supposing that $\dot{\gamma} = \dot{\gamma}_{\mathrm{av}} = \text{constant}$, we obtain

$$v_f = v_0 - \frac{\dot{\gamma}_{\mathrm{av}}\Delta s}{f} + \frac{T}{q}\ln\left(\frac{m_0 - qt_f}{m_0 - qt_0}\right). \tag{E.20}$$



Bibliography

[ABSJM07] M. Abedi, H. Bolandi, F. F. Saberi, and M. R. Jahed-Motlagh. An adaptive rbf neural guidance law surface to air missile considering target and control loop uncertainties. *In Proceedings of the IEEE International Symposium on Industrial Electronics*, pages 257–262, 2007.

[AH83] J. R. Andrews and N. Hogan. Impedance control as a framework for implementing obstacle avoidance in a manipulator. *Control of Manufacturing Processes and Robotic Systems*, pages 243–251, 1983.

[AKH90] S. Akishita, S. Kawamura, and K. Hayashi. New navigation function utilizing hydrodynamic potential for mobile robot. *In Proceedings of the IEEE International Workshop on Intelligent Motion Control*, 2:413–417, 1990.

[BBT07] H. Bouadi, M. Bouchoucha, and M. Tadjine. Sliding mode control based on backstepping approach for an uav type-quadrotor. *International Journal of Mechanical, Aerospace, Industial and Mechatronics Engineering*, 1(2):60–65, 2007.

[BCL91] J. D. Boissonnat, A. Cérézo, and J. Leblond. Shortest paths of bounded curvature in the plane. Technical report, Institut National de Recherche en Informatique et en Automatique, 1991.

[Bec90] K. Becker. Closed-form solution of pure proportional navigation. *IEEE transaction on aerospace and electronic systems*, 26:526–533, 1990.

[Bel54] R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[Bet98] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[BHPL06] S. Bertrand, T. Hamel, and H. Piet-Lahanier. Performance improvement of an adaptive controller using model predictive control : Application to an uav model. *In Proceedings of the 4th IFAC Symposium Mechatronic Systems*, 2006.

[BL90] J. Barraquand and J.-C. Latombe. A monte-carlo algorithm for path planning with many degrees of freedom. *In Proceedings of the IEEE International Conference on Robotics and Automation*, 3:1712–1717, 1990.

[BL91] J. Barraquand and J.-C. Latombe. Robot motion planning : A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, 1991.

[BNS04] S. Bouabdallah, A. Noth, and R. Siegwart. Pid vs. lq control techniques applied to an indoor micro quadrotor. *Intelligent Robots and Systems*, 2004.

[Cox92] E. Cox. Fuzzy fundamentals. *IEEE Spectrum*, pages 58–61, 1992.

[CP05] S. Carprin and G. Pillonetto. Merging the adaptive random walks planner with the randomized potential field planner. *In Proceedings of the 5th International Workshop on Robot Motion and Control*, pages 151–156, 2005.

[CSW98] P. A. Creaser, B. A. Stacey, and B. A. White. Evolutionary generation of fuzzy guidance laws. *UKACC international conference on control '98*, 2:883–888, 1998.

[DDG04] G. M. Dimirovski, S. M. Deskovski, and Z. M. Gacovski. Classical and fuzzy-system guidance laws in homing missiles systems. *In Proceedings of the IEEE Aerospace Conference*, 5:3032–3047, 2004.

[Dic12] S. Dicheva. *Planification de mission pour un système de lancement aéroporté autonome*. PhD thesis, Université d'Évry-Val-d'Essonne, 2012.

[Dij59] E. W. Dijkstra. *A note on two problems in connection with graphs*, chapter 1, pages 269–271. 1959.

[Dij65] E. W. Dijkstra. *Solution of a problem in concurrent programming control*, volume 8, chapter 8(9), page 569. Comm. ACM, 1965.

[DJ10] T. Dierks and S. Jagannathan. Output feedback control of a quadrotor uav using neural networks. *IEEE Transactions on Neural Networks*, 21(1):50–66, 2010.

[DNKF10] K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, 39:533–579, 2010.

[DSF00] E. Devaud, H. Siguerdidjane, and S. Font. Some control strategies for a high-angle-of-attack missile autopilot. *Control Engineering Practice*, 8(8):885–892, 2000.

[Dub57] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal position and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[DVTK04] L. Doitsidis, K. P. Valavanis, N. C. Tsouveloudis, and M. Kontitsis. A framework for fuzzy logic based uav navigation and control. *In Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4041–4046, 2004.

[Efe11] M. O. Efe. Neural network assisted computationally simple pi $^{\lambda}$ d$^{\mu}$ control of a quadrotor uav. *IEEE Transactions on Industrial Informatics*, 7(2):354–361, 2011.

[EKR95] G. Elnagar, M. Kazemi, and M. Razzaghi. The pseudospectral legendre method for discretizing optimal control problems. *IEEE transactions on Automatic Control*, 40(10):1793–1796, 1995.

[ER98] G. Elnagar and M. Razzaghi. A collocation-type method for linear quadratic optimal control problems. *Optimal control applications and methods*, 18(3):227–235, 1998.

[Fal14] Paul Falstad. http://www.falstad.com/vector3d/directions.html, February 2014.

[FGQ12] L. De Filippis, G. Guglieri, and F. Quagliotti. Path planning strategies for uavs in 3d environments. *Journal of Intelligent & Robotic Sytems*, 65(1-4):247–264, 2012.

[FK11] I. F. Filippidis and K. J. Kyriakopoulos. Adjustable navigation functions for unknown sphere worlds. *IEEE Conference on Decision and Control and European Control Conference*, pages 4276–4281, 2011.

[FK12] I. F. Filippidis and K. J. Kyriakopoulos. Navigation functions for everywhere partially sufficiently curves worlds. *IEEE International Conference on Robotics and Automation*, pages 2115–2120, 2012.

[FKA12] I. F. Filippidis, K. J. Kyriakopoulos, and P. K. Artemiadis. Navigation functions learning from experiments: application to anthropomorphic grasping. *IEEE International Conference on Robotics and Automation*, pages 570–575, May 14-18 2012.

[GC95] P. G. Gonsalves and A. K. Caglayan. Fuzzy logic pid controller for missile terminal guidance. *In Proceedings of the IEEE Iinternational Symposium on Intelligent Control*, pages 377–382, 1995.

[GHM08] N. Guenard, T. Hamel, and R. MAhony. A practical visual servo control for an unmanned aerial vehicle. *IEEE Transactions on Robotics*, 24(2):331–340, 2008.

[God19] R. H. Goddard. A method of reaching extreme altitudes. In *Smithsonain miscellaneous collections*, volume 11. The smithsonain institution, 1919.

[GPM89] C. E. García, D. M. Prett, and M. Morari. Model predictive control: theory and practice - a survey. *Automatica*, 25(3):335–348, 1989.

[Gue76] M. Guelman. The closed-form solution of true proportional navigation. *IEEE Transactions on Aerospace and Electronic Systems*, AES-12:472–482, 1976.

[Hea52] E. Heap. Methodology of research into command-line-of-sight and homing guidance. *AGARD Lecture series no. 52 on guidance of tactical missiles*, May 1952.

[Hel11] T. Hellstrom. Robot navigation with potential fields. Technical report, Department of computing science Umea University, December 2011.

[HG10] S. Hota and D. Ghose. Optimal path planning for an aerial vehicle in 3d space. *In Proceedings of the IEEE Conference on Decision and Control*, pages 4902–4907, 2010.

[HLM97] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *In Proceedings of the IEEE International Conference on Robotics and Automation*, 3:2719–2726, 1997.

[HNR68] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, ssc-4:100–107, 1968.

[HP13] B. Hérissé and R. Pepy. Shortest paths for the dubins' vehicle in heterogeneous environments. *In Proceedings of the IEEE Conference on Decision and Control*, pages 4504–4509, 2013.

[HV13] C. Hajiyev and S. Y. Vural. Lqr controller with kalman estimator applied to uav longitudinal dynamics. *Positioning*, 4(1), 2013.

[JT08] D. Jung and P. Tsiotras. Bank-to-turn control for a small uav using backstepping and parameter adaptation. *International Federation of Automatic Control*, pages 4406–4411, 2008.

[Kel76] H. B. Keller. Numerical solution of two point boundary value problems. *SIAM*, 1976.

[KF10] S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. *In Proceedings of the IEEE Conference on Decision and Control*, pages 7681–7687, 2010.

[KF11] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30:846–894, 2011.

[KG11] B. Kada and Y. Ghazzawi. Robust pid controller design for an uav flight control system. *In Proceedings of the World Congress on Engineering and Computor Science*, 2, 2011.

[Kha85] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *In Proceedings of the IEEE International Conference on Robotics and Automation*, pages 500–505, 1985.

[KL00] J. J. Kuffner and S. M. LaValle. Rrt-connect : An efficient approach to single-query path planning. *In Proceedings of the IEEE International Conference on Robotics and Automation*, pages 995–1001, 2000.

[KL02] S. Koenig and M. Likhachev. D* lite. *American Association for Artificial Intelligence*, pages 476–483, 2002.

[Kok81] P. V. Kokotović. Subsystems, time scales and multimodeling. *Automatica*, 17(6):789–795, 1981.

[Kok92] P. V. Kokotović. The joy of feedback: nonlinear and adaptive. *IEEE Control Systems Magazine*, 12:7–17, 1992.

[Kor85] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

[KSLO96] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580, 1996.

[LaV98] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[LaV06] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[Lei99] D. J. Leith. Survey of gain-scheduling analysis & design. *International Journal of Control*, 73:1001–1025, 1999.

[Lin91] C. F. Lin. *Modern Navigation Guidance and Control Processing*. Prentice-Hall, Inc., 1991.

[LK99] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *In Proceedings of the IEEE International Conference on Robotics and Automation*, (5):473–479, 1999.

[LK01] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees : Progress and prospect. *Algorithmic and Computational Robotics : New Directions*, pages 293–308, 2001.

[LM99] C.-M. Lin and Y.-J. Mon. Fuzzy-logic-based guidance law design for missile systems. *In Proceedings of the IEEE International Conference on Control Applications*, 1:421–426, 1999.

[LY09] S.-Q. Li and L.-Y. Yuan. Design of fuzzy logic missile guidance law with minimal rule base. *In Proceedings of the IEEE Sixth International Conference on Fuzzy Systems and Knownledge Discovery*, pages 176–180, 2009.

[MAB98] E. Mazer, J. M. Ahuactzin, and P. Bessiere. The ariadne's clew algorithm. *Journal of Artifical Intelligence Research*, 9:295–316, 1998.

[MATB96] E. Mazer, J. M. Ahuactzin, E.-G. Talbi, and P. Bessiere. The ariadne's clew algorithm. 1996.

[MBB05] A. Mokhtari, A. Benallegue, and A. Belaidi. Polynomial linear quadratic gaussian and sliding mode observer for a quadrotor unmanned aerial vehicle. *Journal of Robotics and Mechatronics*, 17(4):483–495, 2005.

[MC66] S. A. Murtaugh and H. E. Criel. Fundamentals of proportional navigation. *IEEE Spectrum*, 3:75–85, 1966.

[MH04] R. Mahony and T. Hamel. Robust trajectory tracking for a scale model autonomous helicopter. *International Journal of Robust and Nonlinear Control*, 14:1035–1059, 2004.

[MP43] W. S. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of mathematical biophysics*, 5:115–133, 1943.

[MS10] F. L. L. Medeiros and J. D. S. Da Silva. A dijkstra algorithm for fixed-wing uav motion planning based on terrain elevation. *Advances in Artificial Intelligence*, pages 213–222, 2010.

[NDKF07] A. Nash, K. Daniel, S. Koenig, and A. Felner. Theta*: any-angle path planning on grids. *In Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1177–1183, 2007.

[NKT10] A. Nash, S. Koenig, and C. Tovey. Lazy theta*: any-angle path planning and path length analysis in 3d. *AAAI press*, 2010.

[PHB15a] P. Pharpatara, B. Hérissé, and Y. Bestaoui. 3d-shortest paths for a hypersonic glider in a heterogeneous environment. *In Proceedings of the IFAC Workshop on Advances Control and Navigation for Autonomous Aerospace Vehicles*, pages 186–191, 2015.

[PHB15b] P. Pharpatara, B. Hérissé, and Y. Bestaoui. 3d trajectory planning of aerial vehicles using rrt*. *IEEE Transaction on Control Systems Technology*, page submitted, 2015.

[PHPB13] P. Pharpatara, B. Hérissé, R. Pepy, and Y. Bestaoui. Samping-based path planning: a new tool for missile guidance. *In Proceedings of the IFAC Symposium on Automatic Control in Aerospace*, pages 131–136, 2013.

[PLM06] R. Pepy, A. Lambert, and H. Mounier. Reducing navigation errors by planning with realistic vehicle model. *In Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 300–307, 2006.

[Poh69] I. Pohl. Bi-directional and heuristic search in path problems. Technical report, Stanford linear accelerator center, May 1969.

[PPHB13] P. Pharpatara, R. Pepy, B. Hérissé, and Y. Bestaoui. Missile trajectory shaping using sampling-based path planning. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2533–2538, 2013.

[PY98] K. M. Passino and S. Yurkovich. *Fuzzy Control*. Addison-Wesley Longman, Inc., 1998.

[RA05] W. Ren and E. Atkins. Nonlinear trajectory tracking for fixed wing uavs via backstepping and parameter adaptation. *In Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, Aug 2005.

[Rac] S. Raczynski. Reachable sets for flight trajectories: an application of differential inclusions to flight maneuver simulation.

[Rao09] A. V. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135:497–528, 2009.

[RBD+10] A. V. Rao, D. A. Benson, C. Darby, C. Francolin M. A. Patterson, I. Sander, and G. T. Huntington. Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method. *ACM Transactions on Mathematical Software*, 37(2):22:1–22:39, 2010.

[RK91] E. Rimon and D. E. Koditschek. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. *Transactions of the American Mathematical Society*, 327:71–116, 1991.

[RK92] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transaction on Robotics and Automation*, 8(5):501–518, 1992.

[RN09] S. Russell and P. Norvig. *Artificial Intelligence A Modern Approach*. Upper Saddle River, New Jersey: Prentice Hall, 2009. Greedy-Best-First pp 92-93 A* pp 93-99.

[SB02] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer-Verlag, 2002.

[Sey93] H. Seywald. Trajectory optimization based on differential inclusion. Technical report, NASA, 1993.

[Sha07] M. Shanmugavel. *Path planning of multiple autonomous vehicles*. PhD thesis, Cranfield University, 2007.

[Shi08] B. M. Shippey. Trajectory optimization using collocation and evolutionary programming for constrained nonlinear dynamical systems. Master's thesis, The university of Texas at Arlington, 2008.

[SHSS07] D. M. Stipanovic, P. F. Hokayem, M. W. Spong, and D. D. Siljak. Cooperative avoidance control for multiagent systems. *Journal of Dynamic Systems, Measurement, and Control*, 129:699–707, 2007.

[Sio04] G. M. Siouris. *Missile guidance and control systems*. Springer-Verlag New York, Inc., 2004.

[SJFD08] B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake. Multiple uavs path planning algorithms: a comparative study. *Fuzzy Optimization and Decision Making*, 7(3):257–267, 2008.

[SL01] A. M. Shkel and V. Lumelsky. Classification of the dubins set. *Robotics and Autonomous Systems*, 34:179–202, 2001.

[SL03] G. Sanchez and J.-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Robotics Research in Advanced Robotics*, volume 6, pages 403–417. Springer Berlin Heidelberg, 2003.

[SOK84] V. R. Saksena, J. O'Reilly, and P. V. Kokotovic. Singular perturbations and time-scale methods in control theory: survey 1976-1983. *Automatica*, 20(3):273–293, 1984.

[ST02] E. J. Song and M. J. Tahk. Three-dimensional midcourse guidance using neural networks for interception of ballistic targets. *IEEE Transactions on Aerospace and Electronic Systems*, 38(2):404 –414, 2002.

[Ste93] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. Technical Report CMU-RI-TR-93-20, Robotics Institute, Pittsburgh, PA, August 1993.

[Sub11] S. Subchan. A direct multiple shooting method for missile trajectory optimization with the terminal bunt maneuver. *The Journal of Technology and Science*, 22(3):147–151, 2011.

[Sus95] H. J. Sussmann. Shortest 3-dimensional paths with prescribed curvature bound. *In Proceedings of the IEEE Conference on Decision and Control*, 4:3306–3312, 1995.

[Tar72] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 2:146–160, 1972.

[Tau02] D.-R. Taur. Midcourse trajectory optimization for a sam against high-speed target. *American Institute of Aeronautics and Astronautics*, 13:487–493, 2002.

[Tré12] E. Trélat. Optimal control and applications to aerospace: some results and challenges. *Journal of Optimization Theory and Applications*, 154(3):713–758, 2012.

[uVP09] D. Šišlák, P. Volf, and M. Pěchouček. Accelerated a* trajectory palnning: grid-based path planning comparison. *In Proceedings of the 19th International Conference on Automated Planning and Scheduling*, pages 76–83, 2009.

[Vor07] G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal fur die Reine und Angewandte Mathematik*, 133:97–178, 1907.

[WAS99] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space. *In Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

[Waw02] N. Wawresky. Évaluation et comparaison de lois de guidancge en présence de manœuvres de la cible. Master's thesis, École des Mines, 2002.

[WC14] C.-H. Wang and C.-Y. Chen. Intelligent missile guidance by using adaptive recurrent neural networks. *In Proceedings of the IEEE International Conference on Networking, Sensing and Control*, pages 559–564, 2014.

[WHD10] Y. Wei, M. Hou, and G.-R. Duan. Adaptive multiple sliding surface control for integrated missile guidance and autopilot with terminal angular constraint. *In Proceedings of the Chinese Control Conference*, pages 2162–2166, 2010.

[Whi] D. Whitley. A genetic algorithm tutorial.

[WZS07] R. Wang, Z. Zhou, and Y. Shen. Flying-wing uav landing control and simulation based on mixed h_2/h_∞. *In Proceedings of the IEEE International Conference on Mechatronics and Automation*, pages 1523–1528, 2007.

[XY12] X. Xing and D. Yuan. Quantitative feedback theory and application in uav flight control. Technical report, Automatic Flight Control Systems - Latest Developments, Dr. Thomas Lombaerts (Ed.), 2012.

[YA11] I. Younas and A. Aqeel. A genetic algorithm for mid-air target interception. *International journal of computor applications*, 14(1):38–42, 2011.

[YC92] P.J. Yuan and J.-S. Chern. Ideal proportional navigation. *Journal of Guidance, Control, and Dynamics*, 15(5):1161–1165, 1992.

[YS12] K. Yang and S. Sukkariech. Model predictive unified planning and control of rotary-wing unmanned aerial vehicle. *In Proceedings of the 12th International Conference on Control, Automation and Systems*, page 19741979, 2012.

[YY87] C.D. Yang and F.-B. Yeh. The closed-form solution of generalized proportional navigation. *Journal of Guidance, Control, and Dynamics*, 10(2):216–218, 1987.

[Zar94] P. Zarchan. *Tactical and Strategic Missile Guidance*, volume 157 of *Progress in Astronautics and Aeronautics*. America Institute of Aeronautics and Astronautics, Inc. (AIAA), second edition, 1994.

[Zho02] R. Zhou. Design of closed loop optimal guidance law using neural networks. *Chinese journal of aeronautics*, 15(2):98–102, 2002.

[Zin90] A. S. I. Zinober. Deterministic control of uncertain systems. *Peter Peregrinus*, 1990.

[ZZZL11] G. Zhang, M.-B. Zhu, Z.-B. Zhao, and X.-P. Li. Trajectory optimization for missile-borne sar imaging phase via gauss pseudospectral method. *In Proceedings of the IEEE CIE International Conference on Radar*, pages 867–870, 2011.