uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Cédric Cocaud

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Mechanical Engineering)

GRADE / DEGREE

Department of Mechanical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Autonomous Tasks Allocation and Path Generation of UAV's

TITRE DE LA THÈSE / TITLE OF THESIS

Dr. Amor Jnifene

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Dr. A. Fahim

Dr. W. Gueaieb

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Autonomous Tasks Allocation and Path Generation of UAV's

Cédric Cocaud

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of APPLIED SCIENCE

In Mechanical Engineering
Ottawa-Carleton Institute for Mechanical and Aerospace Engineering

University of Ottawa
Ottawa, Ontario, Canada
July 2006

Your file Votre référence
ISBN: 978-0-494-34061-5
Our file Notre référence
ISBN: 978-0-494-34061-5

Abstract

The feasibility and survivability of Unmanned Air Vehicles (UAV's) in field applications have been demonstrated during many of the recent conflicts. UAV's have the potential to reduce costs and personnel while performing certain missions such as surveillance and reconnaissance with higher efficiency and lower risks. Since it is now recognized that a single UAV cannot achieve all tasks required during a mission, the option of using fleets of UAV's with complementary capabilities is now explored. However, controlling and coordinating such fleets will be almost impossible when considering the overwhelming number of degrees of freedom and their associated constraints, if individual UAV do not possess a high level of autonomy.

Addressing the need for intelligent controllers, this work presents a new approach for on-line trajectory planning of a single UAV. The approach uses a genetic algorithm and integrates constraints of different types including specific times of arrival, vectors of approach close to target points, multiple objectives, weather conditions and behavioural guidelines with respect to speed, altitude and obstacle distance. The UAV's controller is divided into three modules. The first one is a trajectory generator that uses an analytical model to simulate the dynamics of UAV's. It is used to compute new trajectories in 3D as well as to assess the feasibility of existing ones. The second module is a knowledge base that uses a hybrid octree structure to represent terrain in 3D. The octree compresses digital elevation data by setting the resolution of the UAV's internal map to different levels based on the homogeneity of the terrain. The third module is a Genetic Algorithm that finds and optimizes candidate trajectories which fulfill all objectives of a mission while taking in account a variable number of constraints with varying preponderance.

Simulations have shown that the intelligent controller could find viable trajectories within 10 to 20 seconds for missions with a single objective, demonstrating the feasibility of using this control scheme for real-time path planning for UAV's. Dynamically feasible trajectories satisfying all constraints could also be found for missions with up to three objectives. However, real time capabilities were severely reduced for such complex missions, the average computing time varying from 200 to 450 seconds. In all cases, all resulting trajectories were dynamically feasible, and could be readily used by the autopilot.

Acknowledgement

I wish to express my deep appreciation to all those who helped me in completing this research.

I am particularly grateful to my supervisor Doctor Amor Jnifene who supported me through out my Master degree as well as the end of my bachelor at the University of Ottawa. His expertise and guidance were key to my success during all these years.

I would also like to thank Doctor Bumsoo Kim from the department of Defence Research and Development Canada, who gave me the opportunity to work on a concrete application related to the fascinating field of Artificial Intelligence.

Special thanks to Sandra Podhajsky who courageously revised this thesis several times.

Table of Contents

List of Figures

List of Tables

List of Symbols

ASO	Altitude, Speed and Obstacle distance
CCW	Counter clockwise
CS	Configuration section (see section 5.1.3.2. step (2))
CW	Clockwise
DEM	Data Elevation Map (see Chapter 4)
DTED	Digital Terrain Elevation Data (see Chapter 4).
EAS	Equivalent air speed.
ETA	Estimated Time of Arrival.
GA	Genetic Algorithm (see section 2.2.5. and Chapter 6).
MB	Mega Byte, i.e. 1048576 Bytes; or 1024 KiloBytes.
MOGA	Multi-Objective Genetic Algorithm (see Chapter 6).
$\mathbf{PT_{en}}$	The entry point of the second sections of a trajectory's segment (see section 5.1.3.2.).
$\mathbf{PT_{ex}}$	The exit point of the third sections of a trajectory's segment (see section 5.1.3.2.).
SM	Standard Map, i.e. a standard 3D map divided into non-overlapping cells of equal size.
TAP	Task Allocation Problem (see section 2.1. and Chapter 3)
TAPr	Task Allocation Process, i.e. the steps required to find the TAP's solution.
TG	Trajectory Generator (see Chapter 5).
ToA	Time of Arrival.
TPD	Temperature, Pressure and Density, i.e. fundamental parameters used to describe atmospheric conditions when computing aircraft trajectories
UAV	Unmanned Air Vehicle.
VoA	Vector of Approach.
$\cdot$	Dot product
$\hat{\bullet}$	An operator calculating the angle θ from a reference vector V_r to another vector V_f measured counter clockwise. The convention for this operator is to place the reference vector as the first argument, i.e. $\theta_{r \to f} = V_r \hat{\bullet} V_f$

List of Key Terms

Chromosome: One possible path linking all transitional and objective waypoints (only used in the context of the Genetic Algorithm).

Descriptor: One or a set of three polynomial equations describing a parameter influencing the UAV's flight. These equations are valid for a specific volume in space which is defined using global coordinates (see section 4.3.1.).

Objective Waypoint: A waypoint defining a position through which a UAV has to pass.

Path: An order sequence of waypoints connecting a starting position to a final one. This definition is extended to "an ordered sequence of spatial coordinates" in section chapter 2 in order to accommodate the various interpretations of this term in the literature.

Profile: A function mapping one of the physical attribute of a trajectory (i.e. time, position, or velocity) to a parametric representation of a path.

Section: A subdivision of a segment describing a parametric curve in 3D space.

Segment: A pair of the waypoints within a chromosome, i.e. a pair of waypoints specified in the path provided by the GA.

System: The intelligent controller developed for this thesis. Also referred to as *controller*.

Trajectory: A set of parameters defining the velocity and position as a function of time between two or more waypoints.

Transitional Waypoint: An intermediary waypoint through which the UAV does not necessarily have to pass.

Waypoint: A set of three coordinates representing a position in space.

Chapter 1: Introduction

1.1. The rationale for UAV's

The feasibility and survivability of Unmanned Air Vehicles (UAVs) in field applications have been demonstrated during several conflicts during the past decade, including the Gulf War, Bosnia, Afghanistan, and the more recent Iraq situation. First considered as a curiosity, UAV technologies have progressively drawn attention up to the point at which it was recommended by the Defence Capability Initiative (DCI) as a high priority requirement for surveillance and reconnaissance (Washington summit, April 20th 1999), as well as NATO's Air Force Armament Group (NAFAG) during the Conference of National Armament Directors (CNAD) where Group VII was put in place to provide a forum for the exchange of national UAV concepts and programmes (*Kreienbaum 2000*). Based on warfare development of the past thirty years, international and national military organizations are changing their objectives from global war to a range of tactical situations including humanitarian relief, peace keeping and regional conflicts where UAVs would provide key functions. Integration schemes are already being developed to determine the allocation of UAVs at divisional, Corps and brigade - regimental - levels (see *Dossier (2) 1999*). The increasing interest in UAVs is obvious for both European and North-American countries when a total investment of 6 billion US dollars for the USA, and an annual budget of 2.9 million US dollars for the EU (see *News 2002*). More than 35 organizations in 14 countries are involved in the production of UAVs, including organizations such as NASA, Boeing, Dassault and British Aerospace (see *Dossier (1) 1999*).

1.1.1. Why developing UAV's ?

UAV's have the potential to reduce cost and personnel while increasing the efficiency and reducing the risks of certain missions (*Kreienbaum 2000*). The single largest cost for operating aircraft is by far qualified personnel. An air-force pilot requires years and hundreds of thousands of dollars to train. Moreover, their lives are at high risks when performing missions such as reconnaissance, surveillance and communication relay over hostile areas. UAVs offer the possibility to avoid endangering lives, while being operated by personnel that requires far less training. Cost reduction could be achieved through

reduced peace-time utilization (training can be achieved without actually flying the aircraft), reduced training time (performed though a simple program update), and less operating personnel (based on increased UAV autonomy). In addition, UAVs constitute a more politically acceptable military system, reducing the risk of loss or capture of personnel.

Table 1.1: *Summary of UAV types and their designated missions*

Military Missions	MALE & HALE	Tactical UAV	VTOL	UCAV	LTA	Micro UAV	LALE
Aerial mine detection		•	•				
Attack				•		•	
Communication monitoring		•	•		•		
Over-the-hill reconnaissance		•	•				
Reconnaissance		•	•				
Suppression of enemy air defence				•			
Surveillance	•		•		•		
Target acquisition				•			
Urban warfare		•	•	•			
Civilian Missions							
Advertising					•		
Agriculture			•				
Communication relay			•		•		
Crop inspection					•		
Filming					•		
Fishery information					•		•
Forest fire control					•		
Forestry inspection					•		
Mapping					•		
Meteorological monitoring					•		•
Power line inspection					•		
Supply transport			•				

MALE & HALE: Medium & High Altitude Long Endurance

Tactical UAVs: short range action UAVs to be integrated at the regimental level

VTOL: Vertical take-off and landing

UCAV: Unmanned combat air vehicle

LTA: Lighter-than-air UAV (i.e. aerostat)

Micro-UAV: air-vehicle that could be contain within a 16 cm (6 inches) square box

LALE: Low altitude long endurance

UAVs also offer the advantages inherent to any robotic system, which consists of efficiently performing long, dull and possibly dangerous missions. Limiting factors inherent to human operated vehicles (such as fatigue of the crew and the limited food & water supplies) could be avoided with the development of long-range and/or long endurance UAVs (*Blyenburgh 1999*).

UAVs have promoted the development of advanced systems and technologies such as robust local communication systems (*Kong et al. 2002* and *News 2002*) and automated agricultural surveillance (*Herwitz et al. 2004* and *Blyenburgh 1999*). Table 1.1 shows a summary of the different types of UAV and their designated mission types.

The challenges that UAVs are currently facing can be divided into three categories, infrastructure, ethics and technology. Infrastructure covers the problem of operating UAV's in civil airspace, which implies that UAV's will have to be able to cope with numerous air traffic control regulations, as well as to insure safe and reliable operation satisfying air regulation standards (*Allouche 2000*). It also covers the problem of compatibility between communication protocols of UAV's, ground control and manned air-traffic.

Ethical considerations must also be addressed regarding the human acceptance for autonomous systems. Pure military effectiveness dictates higher level of autonomy. Pursuing this direction may be questionable when considering that these highly autonomous air vehicles may be armed with potentially lethal equipment. One reasonable guideline would be to always keep humans in the loop for all critical decisions.

Although infrastructure and ethics are raising concerns that still need to be addressed, the most constraining problem remains the limitations of current technologies. Guidance and navigation controllers need to be faster and more robust. Communication data links need to be more reliable and secure. True all-weather sensors are yet to be developed while integration of training capabilities remains a research topic. Based on *Dossier (2) 1999*, the future of UAV's depends for a large part on the development of high quality sensors, low price and low weight components.

1.1.2. The need for intelligent autonomous UAV's

Based on *Allouche 2000*, 20% of errors made during a flight are human errors. The same percentage seems to apply for manned aircrafts and UAV operations. Current systems require a minimum of two ground control operators for a single UAV. The first one is responsible for controlling the aircraft to the desired flight pattern while the second one takes care of operating the vehicle's payload as well as recording and analyzing upcoming data. This configuration increases the risks of coordination errors between operators' commands. On the other hand, it has now been recognized that it is impossible to produce a single UAV that can fulfill all roles for a designated mission (*Blyenburgh 1999*). Thus, the new direction being pursued is the development of fleets which integrate UAV's with different equipment and different capabilities. Based on the current level of autonomy of these systems, operators will most likely be unable to cope with the overwhelming number of degrees of freedom that must be controlled in real-time, and the constraints associated with controlling individual UAV's and coordinating between them to achieve a given set of objectives for a particular mission. Thus, the future of UAVs requires intelligent controllers demonstrating a very high level of autonomy. The fundamental problem that this new generation of controllers has to address is referred to in computer science as the Task Allocation Problem (TAP), which consists of determining the order in which a given set of objectives must be completed as well as the best pathway to reach each of them.

1.2. Objective of this work

Attempting to address the need of more advanced autonomous controllers, the work presented here aims to establish a control scheme that will autonomously solve the task allocation problem (TAP) in real-time, and could easily be used for on-line coordination of a fleet of UAV's. The system described in the following sections provides the means to deal with a wide variety of missions, constraints and types of UAV.

1.3. Scope

This work is limited to the Task Allocation aspect of the UAV control system. The corresponding modules of the overall architecture of a UAV shown in Figure 1.1 are the *Knowledge Base* and the *Global Planner*.

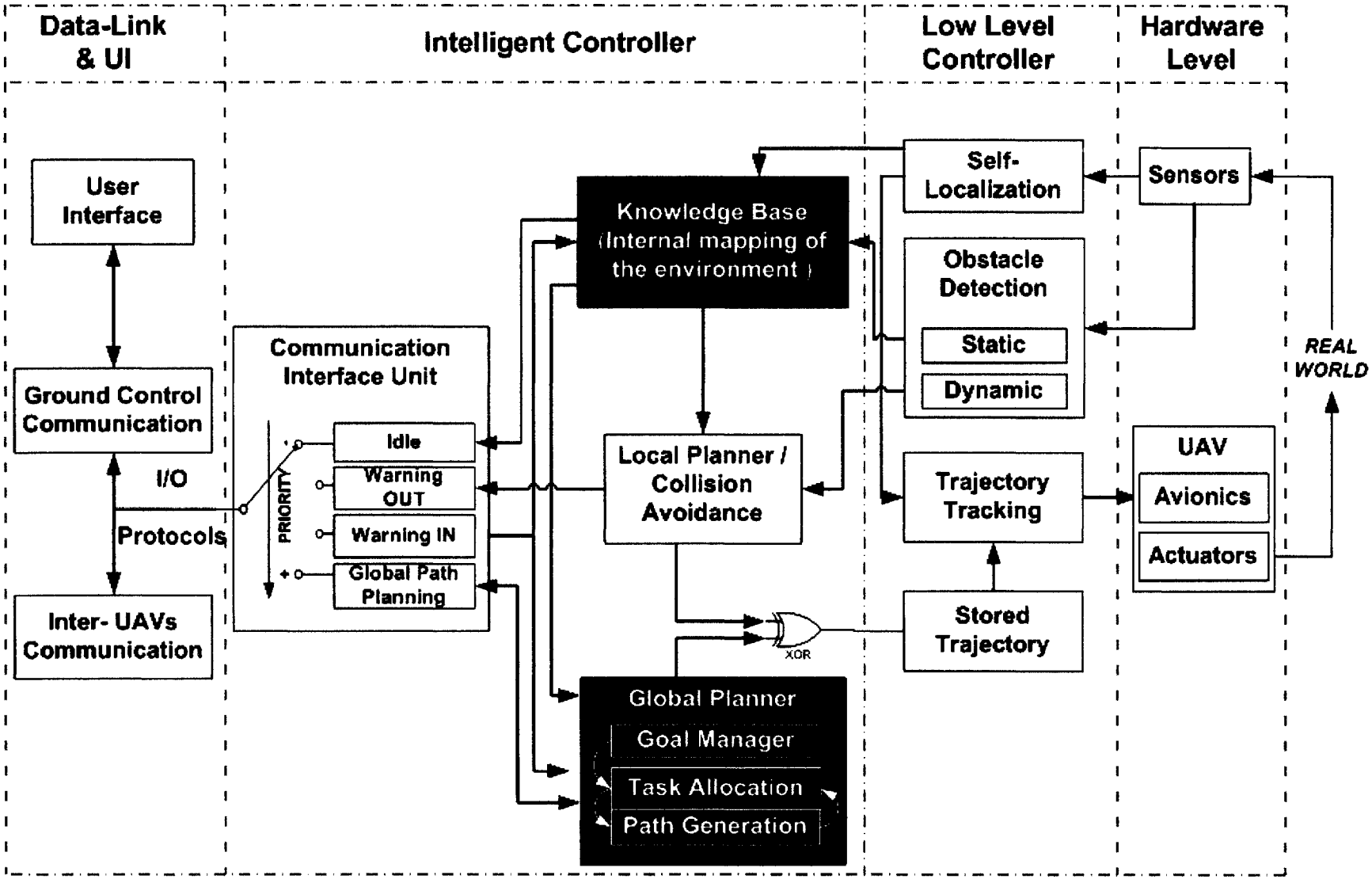


Figure 1.1: *Overall UAV architecture*

The system described in this thesis is only responsible for assigning goals and tasks, specified as waypoints in the mission profile, to each UAV in the fleet. The system is designed to provide an optimized trajectory for a single UAV, taking into account its dynamics and the order in which objectives are executed. On-line re-planning and collision avoidance with static obstacles are also parts of the proposed system.

Real-time is defined here as the time available between two maneuvers. It typically ranges from a few seconds to a few minutes depending on the type of mission and UAV.

The Local Planner in Figure 1.1 which is responsible for reactive collision avoidance, communication systems (i.e. Data Link and Communication Unit), low-level controller and the physical hardware of the UAV are outside the scope of the present work.

1.4. Problem definition & approach

The Task Allocation Problem consists of five steps:

1. Establish the resource available for a mission,
2. Define the list of objectives and constraints,
3. Set the order in which objectives must be completed,
4. Compute the best waypoint sequence that completes all objectives, and
5. Generate the trajectories associated with each waypoint sequence.

The standard procedure to solve the TAP consists of computing a trajectory such that all the objectives of a mission are achieved in some optimal way. The overall level of optimization, or "cost", of a trajectory is determined by the amount of resources required to complete it. Trajectories are optimized by reorganizing the sequence in which the mission's objectives are completed, and by changing intermediate waypoints between each objective.

The system proposed herein provides the means to generate dynamically feasible trajectories based on individual UAV characteristics, while avoiding all static objects and "no fly zones". It optimizes the sequence in which objectives are completed by taking into account a series of basic and user-defined criteria, as well as three parameters related to the desired flying altitude, speed and obstacle proximity.

The approach followed in this work consists of developing a portable software package that can generate valid solutions in a very short period of time while providing

increasing level of optimization for increasing time intervals. In order to provide for both flexibility and efficiency, this software package is meant to take full advantage of parallel processing hardware without depending on it in the eventuality where such hardware is unavailable. According to these guidelines, the software is mainly procedural, aiming for time performance rather than code reusability.

1.5. Thesis outline

The first chapter of this thesis explains the rationale for the use of UAV's as well as the difficulties that needs to be overcome in their future development.

Chapter 2 presents a literature review summarizing the state of the art in the field of advanced controllers and artificial intelligence for autonomous vehicles.

Chapter 3 is an overview of the intelligent controller. It explains the different parts of the system as well as its design approach.

Chapters 4 to 6 describe separately the theory of each of the three modules constituting the system.

Chapter 4 describes the *Knowledge Base* which is implemented as a multi-layer hybrid octree representing the UAV environment. It compresses the information contained in a standard Digital Elevation Map (DEM) by locally modifying the map's resolution and adding information on atmospheric conditions.

Chapter 5 is description of the *Trajectory Generator* which creates trajectories linking a start point to a variable number of waypoints defined as mission objectives. It uses an analytical model of aircraft dynamics to assess the trajectories' feasibility.

Chapter 6 describes the mechanism of the *Multi-Objective Genetic Algorithm* (MOGA) which finds and optimizes candidate trajectories fulfilling all mission objectives while taking into account a variable number of constraints with varying preponderance.

Chapter 7 shows the simulation results assessing the performances of each of the three modules outlined in the previous chapters. It also presents the overall performance and limitations of the system when attempting to solve the TAP for missions ranging from one to three objectives.

Chapter 8 concludes this thesis by summarizing the main achievements and making suggestions on future works.

Chapter 2: Literature Review

Current UAV's operations are partially automated, enabling aircrafts to fly autonomously along a direct path between two waypoints using a low-level controller (auto-pilot) that simply compensates for the heading and position error. Such controllers have first been developed for tactical airborne missile systems, ranging from the most simplistic LOS (Line-Of-Site) guidance and PNG (Proportional Navigation Guidance) systems, which modify the attitude based on heading error, to the more complex ones including "velocity compensated PNG", "pursuit plus PNG" and "dynamic lead guidance" which can cope with manoeuvring targets (*Naeem et al. 2003*). The Apollo program in the 1960s demonstrated the successful adaptation of these schemes to a complex NGC (Navigation, Guidance and Control) system (*Apollo Shuttle Systems 1998*). Recent guidance systems have incorporated more sophisticated control schemes such as Fuzzy Logic, Fuzzy Genetic Algorithms, and H_∞ based guidance law (*Naeem et al. 2003*). Some of these guidance systems have been adapted to respond to the specific requirements of autonomous vehicles such as UAVs and Autonomous Underwater Vehicles (AUVs). The main control schemes pursued up to now have been:

- *Waypoints Guidance by LOS or PNG* which is the equivalent of PD and PID controllers.
- *Vision based guidance* using different stereo-camera systems or combinations of laser and cameras.
- *Lyapunov-based guidance* controllers using stable control laws that minimizes the error between current and desired position and orientation (or attitude).
- *Guidance with chemical signals*: controller following a steepest descent algorithm toward higher chemical concentrations.
- *Magnetic and electromagnetic guidance* which is the equivalent to guidance with chemical signals (*Naeem et al. 2003*) but using sensors detecting magnetic and electromagnetic fields.
- *Advanced non-linear controllers* as in *Choi and Yuh (1996)* that cope with dynamic changes of the system.

Most of the progresses achieved toward more intelligent UAV systems have been oriented toward the development of control algorithms for cohesive group formations; these algorithms ranged from simple PID control based on relative positional error of different UAV referee in the formation, to sliding mode control (*Singh, Pachter, et al. 2000*) and other nonlinear adaptive control schemes (*Singh, Zhang, et al. 2003*; and *Singh, Chandler, et al. 2000*). Some work dealing with optimal trajectory generation has been done (*Singh and Fuller 2001*) but literature addressing problems specific to UAV's remain very sparse compared to the one available for wheeled robots, rovers, or autonomous vehicles (none-UAV) in general.

It is only during the past five years that "intelligent" controls for UAV's have started appearing as a distinct domain in the scientific literature. The development of these intelligent control schemes can be summarized in two stages. The first one aims to provide UAV's with some reasoning capabilities for a given operational goal (i.e. a global mission statement), allowing them to make decisions without human intervention (*Barbier and Chanthery 2004*). This would give great flexibility to UAV's, allowing them to adapt by replanning their actions in response to the changing environment during an ongoing mission (*Lucas et al. 2003*). The second stage aims to raise the level of autonomy once more by letting a fleet of UAV's coordinate itself without human guidance. This problem is generally referred as the Multi-Tasks Allocation Problem (MTAP). The simpler problem of letting a single UAV coordinate its own course of action is referred analogously as the TAP, which is essentially what the first generation of intelligent control is doing with varying level of success. The following section discusses the Task Allocation Problem and the different algorithms that have been developed to solve it.

2.1. Task allocation problem (TAP)

Before proceeding with the description of the TAP, it is necessary to define two terms that will be used extensively throughout this thesis: *path* and *trajectory*. The term *trajectory* specifically refers to a description of the position and velocity of a vehicle for a specific time interval. The term *path* only refers to the sequence of spatial coordinates that a vehicle has to traverse. Due to the different interpretations of this term in the literature, "path" will be used in its broadest sense in this chapter. A formal and much

more rigorous definition is given in the glossary, and applies for all other sections of this thesis.

This problem consists of five steps (*Tierno 2001*; *Richards et al. 2002*; *Rabbath et al. 2004*):

1. Establishing the list of resources available for a mission, i.e. number of UAV, onboard equipment available for each UAV, etc.,
2. Defining the list of objectives to complete, and possibly assigning priorities to the different objectives,
3. Allocating resources (i.e. UAVs) to each objective, with the possibility of assigning more than one objective for each resource,
4. Computing the best sequence of objectives,
5. Generating the trajectories implementing the action plan obtained from the steps above.

More precisely, an action plan consists of a sequence of waypoints linked by trajectories defining the position and velocity of each UAV at any time t during a mission. Generally, an objective can be defined as a specific location or area (i.e. one waypoint or a set of waypoints) where an action has to be performed by a UAV using onboard equipment. The TAP is not concerned with the details of each action to be performed. The main purpose of the TAP is to determine the optimal plan required to achieve all of these actions in the most efficient manner. In order to have consistent solutions for the TAP, long-duration actions (such as circling continuously around an area) needs to be broken down into a series of sub-objectives (waypoints-action pairs), so that time and use of resources can be minimized.

The TAP is a very complex problem. From a computational point of view, the complexity of any given problem is typically measured based on the number of possible solutions that must be searched before finding a valid one. Problems of polynomial order (i.e. $O(N^X)$ with $x = 1,2,3\ldots$) can generally be solved for optimality using near-exhaustive search algorithms. If x is large, more time-efficient algorithm are usually available such that an optimal solution can still be found without exhaustive search. Problems of higher complexity for which no polynomial-time algorithms are available, can be defined as

non-deterministic problem (NP). The order of complexity of such problems are exponential and generally ranges from O(N!) to $O(X^N)$ with X an integer greater than 2.

Due to the complexity of NPs, optimality can no longer be guaranteed with available search algorithms. Assuming one UAV, a set of objectives with no constraints other than the completion of all objectives, and a series of costs associated with each path linking every pair of objectives, the TAP becomes an instance of the well known Traveling Salesman Problem (TSP) for which numerous algorithms have been derived (*Luger 2002*; and *Winston and Venkataramanan 2003*). The TSP is known to be a NP-complete problem and optimality cannot always be guaranteed for large problems of this class. The TAP for a fleet of identical UAV's, with a set of objectives that can be achieved in any sequence by any UAV, is equivalent to the Mutli-Traveling Salesman Problem (MTSP), which is a NP-complete problem of higher order than the TSP. Heterogeneous fleets of UAV's adds to the complexity of the problem by varying the cost of each path linking two objectives (i.e. the path cost will depend on the member of the fleet taking it). The full TAP problem is further complicated by the fact that a given UAV could be allowed to define a velocity-varying trajectory, which implies that the same path (connecting two objectives), followed by the same UAV can have different costs.

The addition of constraints such as timing requirements for certain objectives does not increase the size of the problem. It only reduces the number of solutions within the problem's search space. Possible constraints relevant to the UAV TAP are shown in Table 2.1 (*McLain et al. 2001*; *Bellingham et al. 2003*).

The main difficulty of the TAP comes from the strong coupling between task assignment and task planning (*Chandler et al. 2003*). The global optimal allocation of objectives between UAV's depends on the costs of the path followed by each UAV. These path costs depend on the sequence of objectives, and thus, depend on global allocation of these same objectives. The optimization of one problem depends on the optimization of the other, and vice versa. Iterative algorithms are unavoidable to solve such problem. Moreover, one must realize that at least two algorithmic module are required to solve the TAP. The first one must be able to find the optimal path between two waypoints, evaluating the cost of the resulting path, while the second one optimizes the combination of paths constituting the global action plan of the fleet.

Table 2.1: *Summary of constraints reported in the literature*

Type of Constraint	Description
UAV-specific objectives	Some objectives may be achieved only by a subset of the UAV fleet being equipped with the proper equipment.
Simultaneous TOA	An objective may require that a subset (or the entire fleet) of UAV's arrive at the same location simultaneously.
Objectives sequencing	Some objectives may have to be performed in a specific sequence, or at specific times.
Objectives scheduling	Some objectives may have to be performed at very specific times after the mission has begun.
Objectives selection	Certain situations may require choosing between two objectives of different value.
Collision avoidance	UAV paths must not cross at the same time.
Obstacle avoidance	UAV paths must never touch an obstacle (be it static or dynamic).
Strategic altitude	It may be desirable to minimize the altitude of UAV's in order to take advantage of covers, or to decrease their exposure to radar or other threats. Conversely, certain types of UAV's are meant to operate at very high altitude (e.g. for surveillance).

Another aspect of the TAP is the definition of optimality. For a single UAV, a solution can be described as optimal when it minimizes a cost function which typically combines cost related to time-of-arrival (TOA), threat exposure, fuel expenditure, etc. When a fleet of UAVs is considered, one must determine if it is more desirable to minimize the sum of all UAVs' costs, or the maximum cost among all members of the fleet -or a combination of the two- (*Brumitt et al. 2001*). Optimality becomes ever more difficult to define when taking into account the fact that predefined plans have a limited lifespan within the mission time frame due to the occurrence of events that could not be known *a priori*. This consideration brings up the concept of finite-time optimality, which

essentially suggests that the rapidity at which a solution can be found should be prioritized over the accuracy of the plan defined.

The last point to be mentioned is the replanning aspect of the TAP. Because of the changing environment in which UAV's operate (e.g. weather conditions, new threats appearing, unforseen obstacles), replanning may be required during a mission. Because of the significant computational burden inherent to solving the TAP, it is important to determine when replanning will occur (*Murphy et al. 1999*). Table 2.2 lists three different options that are generally considered.

Table 2.2: *Replanning schemes*

Replanning scheme	Description
Event-driven	Replanning is triggered by an event such as the occurrence of a new obstacle that blocks the path, or the arrival of a map-update message through the communication link (*Brumitt et al. 2002*).
Scheduled	Replanning is automatically done after a fixed amount of time or distance traveled
Error-driven	Replanning is triggered when the robot's internal representation and the sensor-based representation give rise to an error exceeding a certain threshold (e.g. computing the difference between the sensed and estimated position of pre-selected landmarks (*Le Fort-Piat et al. 1997*).

Chandler et al. 2003 have analyzed several algorithms based on feasibility, optimality, strength and type of coupling, required communications, and degree of decentralization. Mixed integer linear programming, auction based optimization, constraint satisfaction based feasibility enforcement, network flow, and heuristics were compared, concluding that decentralized solutions do not necessarily perform better than centralized solutions, unless communication costs are low. Some examples using these algorithms are described below.

2.1.1. Mixed-integer linear programming

Bellingham et al. 2003, investigated the use of mixed-integer linear programming to find the optimum solution of the simplest form of the TAP. A mission was defined as visiting a set of waypoints while avoiding "No Fly Zones". It is assumed that waypoints can be visited in any given order. The list of objectives to be completed is considered to be given, and resource allocation is assumed to have already been performed. In other words, groups of feasible objectives have already been attributed to teams of UAVs. The algorithm must be performed for each team separately and no tools are provided to check compatibility between their action plans. Optimization of the TAP is based on a cost function weighting the average completion time of the team and the maximum completion time among all members. The algorithm used to find the shortest path between two waypoints corresponded to the Floyd-Warshall All-Pairs Shortest Path algorithm which is based on visibility graphs between a point and the obstacles' vertices.

2.1.2. Decentralized model predictive control

Using decentralized model predictive control, *Tierno 2001* implemented an auction-based control scheme (also called Market-oriented programming) to proceed to the allocation of UAV resources to a given set of tasks. The allocation algorithm works by swapping one resource at a time between tasks such that the scoring function related to the probability of success is progressively maximized, while the scoring function for combat exposures is minimized.

2.1.3. D* algorithm

Working more specifically on autonomous vehicles (unmanned military Hummers), *Brumitt and Stentz 1998* developed a "mission statement grammar" in which complex statements describing multi-objectives multi-vehicles missions can be formulated and automatically decomposed into simpler statements describing vehicles-objectives associations. Using a "plannable grammar", the first step consists of translating the mission statement into atomic sub-statements which can be readily associated with four basic problems: (1) selecting minimum cost path; (2) adding minimum cost path; (3) TSP; (4) MTSP. The TSP and MTSP are solved using the Simulated Annealing algorithm described in *Winston 2003*, which consists of a randomized search based on an objective function. The function used by the authors mentioned above consisted in a weighted

average of the cost of each vehicle's path over the longest one. The algorithm used to search individual paths corresponded to a heuristic depth-first search called the D* Algorithm (*Stentz 1994*). This algorithm is a variation of the A* algorithm (*Luger 2002*). It decomposes the search space into a tree-graph in which the root is the starting point and the branches are all the possible directions that the vehicle can take. The goal is one of the leaves of the tree. The A* algorithm proceeds to a depth-first search where the node selected to deepen the search is the one with the lowest estimated cost based on the estimation function $f(x) = h(x) + g(x)$. The term $g(x)$ correspond to the cost of the path that separates node x from the root, while $h(x)$ is a heuristic function that estimates the distance between the node x and the goal. The difference between the A* and the D* algorithms is that the latter allows dynamic costs adjustments while searching for some near-optimal path. The overall control scheme developed in this work used an event-driven replanning scheme where allocation of objective waypoints and optimal path are recomputed altogether.

2.1.4. Voronoï diagram search

Focusing on a mission where a team of up to three UAV's must reach a target at the same time, *McLain et al. 2001* solved the TAP by performing exhaustive search distributed between UAV's. The search space was reduced by performing computations only for regions of overlapping Estimated-Time-Of-Arrival (ETA). The overall allocation of the number of UAV's for a given target (i.e. the objective) was performed by explicitly assessing the risks for a team of 1, 2 or 3 UAV's, selecting the team for which the risk was minimal. All vehicles' speeds were assumed constant along each path. Following to this work, *Beard et al. 2002* relaxed some of the initial assumptions by allowing velocity variations along a path, while integrating feasible trajectory generation using the dynamics of the UAV's. The path planner module used the Eppstein's k-best paths graph search using a Voronoï diagram constructed from known threat locations. The Voronoï diagram was constructed by linking all the points that were equidistant from all near-by threats. Each arc of the diagram was assigned an average threat cost (inversely proportional to the distance that separates it from the threat) and a length cost. The path planner performed the graph search using an objective function that weighted out the threat and length costs. The target manager (i.e. task allocation module) used an

acceptability function and rejectability functions that were computed based on path costs for each possible target / UAV allocation pair. These functions were designed to reflect four objectives: (1) minimizing group paths length; (2) minimizing group threat exposure; (3) maximizing the number of vehicles prosecuting each target; (4) maximizing the number of targets visited. Despite the fact that these functions can be evaluated in $O(k*N*M)$ time (with k the number of paths considered, N the number of possible targets and M the number of vehicles), the overall algorithm is still computationally intensive.

2.1.5. Genetic Algorithm

Avoiding linear programming optimization algorithms and explicit iterative allocation procedures, *Zheng et al. 2004* have developed a Genetic Algorithm to simultaneously find optimal paths and task allocation for a fleet of UAVs, using the same conditions as in *Bellingham et al. 2003* described previously. Each chromosome corresponded to a flight path composed of a variable number of path points (the genes) with predefined start and finish points. The initial population was generated randomly. The evaluation function to assess the quality of each candidate paths (i.e. individual) was a weighted sum of three factors: path length, average altitude (penalizing high altitude), and threat exposure. The dynamic of the vehicle was taken into consideration by imposing constraints on the maximum turning, climbing, and diving angles. The algorithm was set to manage dynamic environments using and continuously updating an Environment Information Look-Up table.

2.1.6. Limitations of current algorithms

All the works mentioned above have several important limitations when considering real-time real-world UAV applications. Most of them do not take the UAV dynamics in account or use very simplistic models. Most of the world representations used with the algorithms described above are relatively simplistic, and may not be sufficient when considering implementation in real-world. In all cases, only 2-D maps (i.e. a predefined area for a constant altitude) using binary space representations were considered. In such representations, the search space is composed of free or occupied (i.e. non-crossable) areas. Terrain elevation data are not considered. Internal representation of the world uses either explicit geometric descriptions of obstacles, or direct discretization of the area considered for the mission. Both of these techniques do not make an efficient use of

memory. On-line path correction is performed through complete replanning, including in most cases re-computation of all individual UAV paths. This approach may become prohibitive for frequent replanning. Finally, no schemes are implemented to cope with new information and uncertainties such as sensor noise, false a priori knowledge, etc.

2.2. Global path planners

The efficiency of the algorithms that solve the TAP strongly depends on the rapid generation of paths between given waypoints. The evaluation of one combination of optimal paths constitutes only one step on the task allocation level. But the generation of these paths alone constitutes an intensive computation of numerous possibilities for each UAV on the path planning level. Further processing time may even be required to ensure that resulting trajectories (i.e. the path's velocity and position profiles) are smooth and can easily be followed by a vehicle despite its inertial and dynamic characteristics. Although real-time smoothing algorithms have already been implemented, certain factors still limit their use in practical applications. This is the case for the work of *Anderson et al. 2005*, where the UAV dynamics had to be simplified to a first-order system, and the resulting path had to be associated to a constant velocity.

Time is an essential factor for path planning algorithms. Warfare history has shown that rigid predefined plans have very limited use. In realistic settings, the conditions in which a mission progresses change very rapidly. Any predefined plan becomes progressively out of phase with the evolving situation. The only remedy is adaptability, a key aspect in military operations. Therefore, the task allocation module is likely to be called on a frequent basis to replan the entire mission. Based on these considerations, rapid path generation is the corner stone of any controller aiming to solve the TAP in real-time.

Path planning has been extensively explored for autonomous vehicles and robotic robots during the past twenty years. An overview of the most common Path Planning techniques can be found in *Cameron 1994*. The techniques that have been developed are general and can be applied to a wide range of robots including UAV's. Path planning can be divided into two domains which are separate but complementary: Global Path Planning (GPP), and Local Path Planning (LPP). The first one is concerned with the generation of an optimal path based on a list of criteria which defines the cost between

two points. The second one covers all types of reactive controls designed to move along the goal heading, while primarily focusing on obstacle and collision avoidance. The latter usually assumes absolutely no a priori knowledge of the environment. Reactive controllers have a tendency to get easily trapped in dead-ends and other types of local minima which are generally avoided when using GPP. Moreover, reactive controller cannot generate near-optimal paths. For this reasons, path planning systems are generally implemented with both GPP and LPP (*Murphy et al. 1999*). Since the scope of this thesis is limited to GPP, LPP will not be covered in this chapter.

Approaches investigated for GPP can be divided into four categories: Roadmap and graph-search; Cell Decomposition; Potential Fields (PF); and Genetic algorithms. All these algorithms must use some form of real-world representation to be able to generate a global plan. Two main categories of world representations, also called *maps*, exist: *metric maps* and *topological maps* (*Prestes et al. 2002*). These two types of representations are discussed in the following section. The different categories of GPP are then described in the subsequent sections of this chapter.

2.2.1. World representations

Topological maps connect key places which have their equivalent in the real-world, with arcs that define some relations between them. However, the arcs do not necessarily implement physical relations such as real-world distances, and the arcs-nodes configuration doesn't necessarily correspond to the real-world configuration of these key places. Topological maps are generally implemented using undirected graph structures. Evaluation of path costs in this type of map is achieved by attributing costs to the arcs. These costs can be based on any kind of heuristics, from traveling time and distance, to threat exposure and fuel consumption.

Metric maps implement the physical configuration of key places of interest using a global frame of reference. Grid-based map are one type of metric map. They are constructed by using arrays of cells discretizing the search space. The real-world distance between two adjacent cells is constant throughout the structure, and is predetermined by the map resolution. Each cell of the array is usually attributed a binary value indicating if the cell is free or occupied (although more complex representations using Fuzzy Logic also exists – *Soltani et al. 2003*). In most cases, obstacles dimensions are increased by a

certain safety factor in the internal representation of the autonomous vehicle. Thus, the region of occupied cells is usually larger; preventing the autonomous vehicle from coming too close of obstacles. This modified representation of the world is called the configuration space, or C-space (*Brumitt et al. 2002*). The distance between two cells being already defined by the map resolution, evaluation of path costs can be done by directly summing up the number of cells visited along the path. If additional metrics need to be considered, costs based on heuristic functions must be attributed to each cell. The overall cost of a path would then correspond to the summation of the costs of all visited cells. As for topological maps, any kind of heuristics can be used.

2.2.2. Roadmap and graph-search

This approach is the most traditional one. The real-world space is first represented internally through a connected graph consisting of arcs and nodes. Depending on the level of complexity, this representation can either be a 2-D or a 3-D map of the real-space in which the autonomous vehicle will evolve. As shown in Figure 2.1, the nodes of the graph represent different geographical points of the real world, while arcs define the cost of moving from one node to the next. The arc costs are usually defined based on a cost function which combines a series of criteria ranging form the Euclidean distance, the city block distance (*Ratering and Gini 1993*), and safety considerations, to manoeuvrability, threat exposure, etc. A vector based cost function approach was investigated in *Refanidis and Vlahavas 2003* to offer a general technique of integrating a weighted hierarchy of user-defined criteria for any kind of search algorithm. Certain types of diagram inherently integrate some of the criteria mentioned above. When considering a path with minimum threat exposure, or maximum distance between the vehicle and obstacles, the Voronoï diagram becomes very useful. By definition, this diagram is constructed by connecting all the points that are equidistant form all near-by obstacles (or threats). Thus, the use of this type of diagram partially implements exposure minimization (*McLain et al. 2001; Beard 2002*). Another type of graph implementing path planning coordination for multiple robots was investigated by *Švestka and Overmars 1998*. The concept was to construct roadmaps for a single robot, and then to combine these roadmaps. The resulting structure was stored in a super-graph from which multi-robot coordinated paths could be retrieved. All these graph representations belong to the topological maps category.

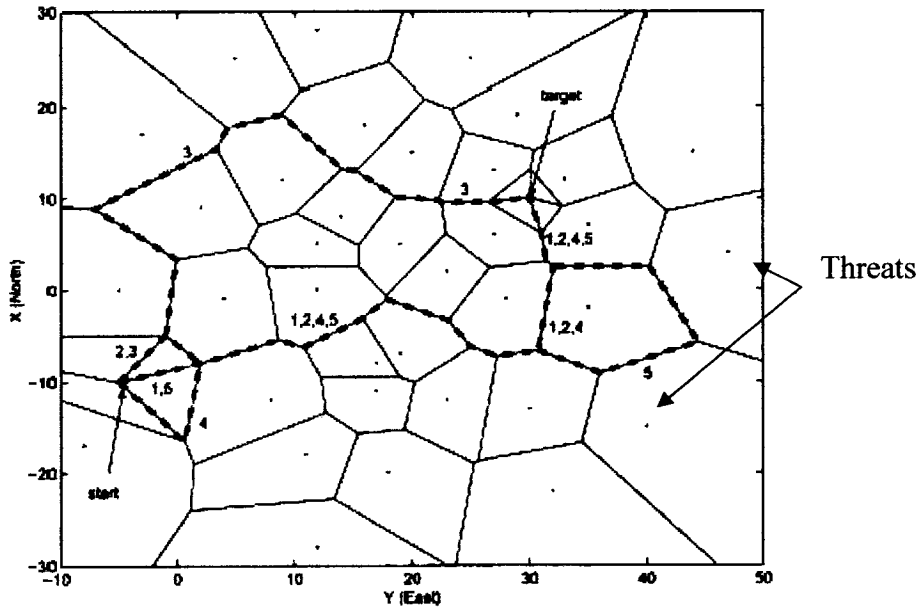


Figure 2.1: *Voronoï diagram*

The advantage of using graph structures is that generic search algorithms can be used with very few modifications. The most common algorithms used are the depth-first A* algorithm, the D* algorithm, and the Dijkstra algorithm (*Goodrich and Tamassia 2001*) which is essentially a breadth-first version of the A* algorithm. Other less common algorithms have also been used such as the Eppstein's k-best paths algorithm (*Beard 2002*). The main difference between the various works that used these algorithms is the type of heuristic (or metric) that they used to estimate the distance from one position to the goal position.

Al-Hasan and Vachtsevanos 2002 used A* with a route cost function based on safety, distance from start position, and manoeuvrability. *Le Fort-Piat et al. 1997* selected the same type of algorithm but based their heuristic evaluation function on the number of landmarks constituting each path. *Barbier and Chanthery 2004* have used the Dijkstra algorithm using a cost function computing the weighted sum of the route length and fuel consumption. Forbidden areas were treated as obstacles, giving them an infinite cost value. As a final example, *Podsedkowski et al. 2001* applied a D* algorithm based on the Cartesian distance and the shortest distance between the current and the goal nodes. The Cartesian distance was computed as the straight line between the two nodes, while the shortest distance estimated the path length that would pass around obstacles.

2.2.3. Cell decomposition

Cell decomposition techniques do not represent path-search algorithms. They implement alternative world representations that can minimize memory usage, and is compatible with a wide variety of path planning techniques including graph-search algorithms and potential fields. The concept of cell decomposition techniques is to vary the resolution of grid-based map such that high resolution grids are available in the vicinity of obstacles, while low resolution is used for large portions of free space. Since many cells are required to encode empty areas (i.e. free space) in standard grid-maps, using different resolutions will decrease the number of cells and thus, reduce the amount of memory required for the internal representation of the environment.

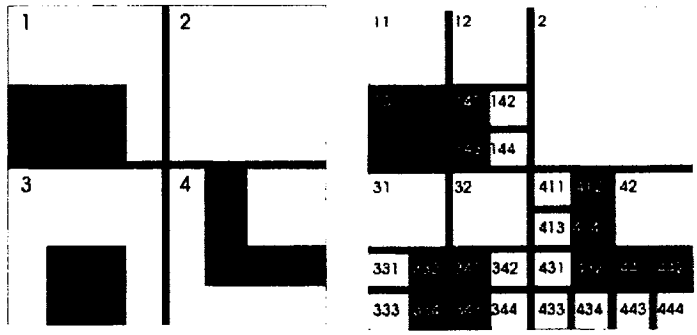


Figure 2.2: *(Left) First step of the cell decomposition process of a 2-D environment with three obstacles; (Right) Final decomposition (Williams and Jones 2001)*

There are three main approaches for cell decomposition: quadtree, octree, and framed quadtree. The first and third approaches can be used for 2-D maps while the second one is designed for 3-D environments. To the author's best knowledge, the implementation of framed octree have not been published yet. The framed quadtree technique could easily be adapted to octrees. All these approaches essentially work in the same way; they use a *tree* data-structure where *leaves* are cells of various sizes discretizing the search space. The general form of the algorithm implementing these three techniques is as follows (*Williams and Jones 2001*): The initial search-space is represented as one cell. If the cell is free of obstacles or completely occupied by one (i.e. the fraction of free space within the cell is negligible), the process is stopped and no further decomposition is required. The cell is attributed a binary value labelling it as free or occupied. This type of cell is referred as a leaf in the tree data-structure. If the cell contains a significant fraction of

free space, it is divided equally into four (4) cells for quadtrees, and eight (8) cells for octrees. These new cells are called the *children* of the initial cell, which in turn, is referred as the *parent*. Each new cell is recursively decomposed by applying the steps mentioned above, until no further decomposition is possible. Figure 2.2 shows the decomposition of a 2-D environment comprising three obstacles.

The decomposition of the search-space is stored in the tree structure shown in Figure 2.3. The white cells indicate free space while the black cells indicate an obstacle. The grey cells correspond to regions of the search space where both obstacles and free space exists. A grey cell is decomposed until its children are either black (occupied) or white (free space).

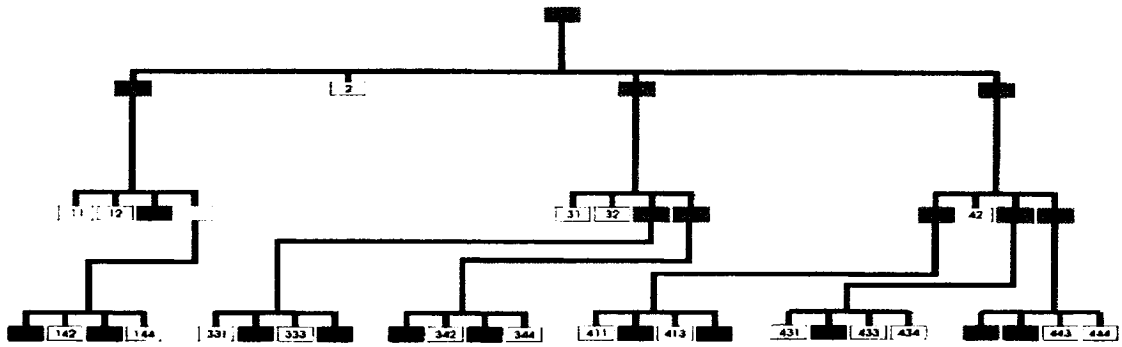


Figure 2.3: *Tree structure obtained for the decomposition (Williams and Jones 2001)*

Depending on the search algorithm, the tree-structure can be transformed into an undirected graph as shown in Figure 2.4. In order to do so, each cell is augmented by a series of pointers that list all its neighbouring cells among the leaves of the tree. The old tree structure can then be discarded to save additional memory and avoid redundant connections.

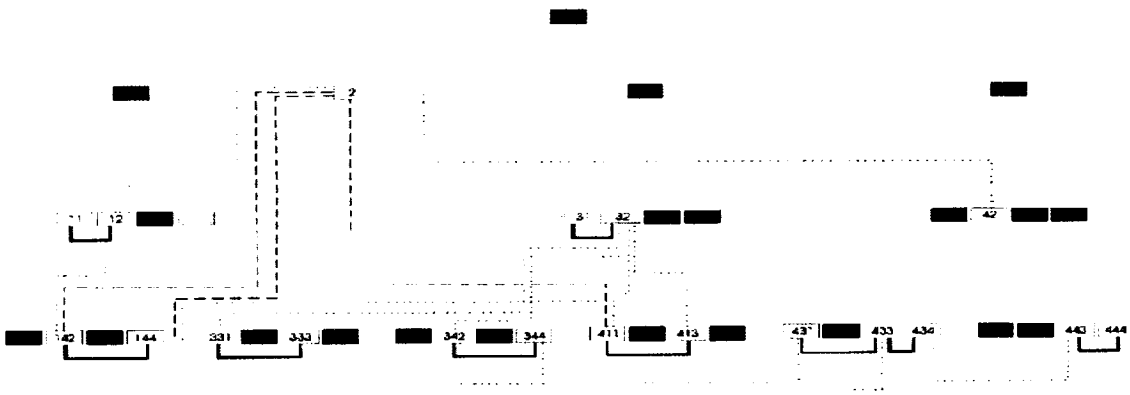


Figure 2.4: *Transform of a tree structure into an undirected graph (Williams and Jones 2001)*

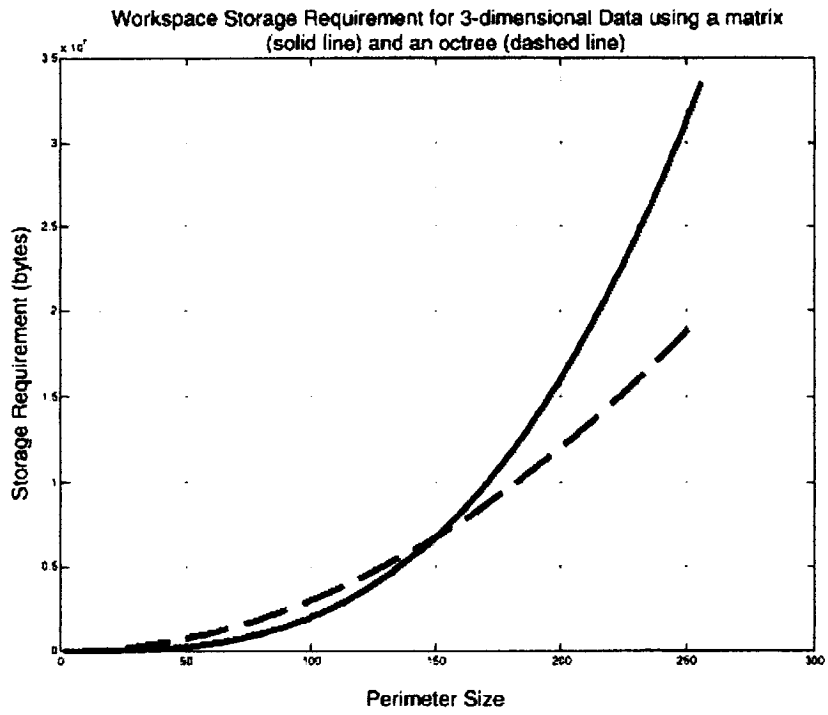
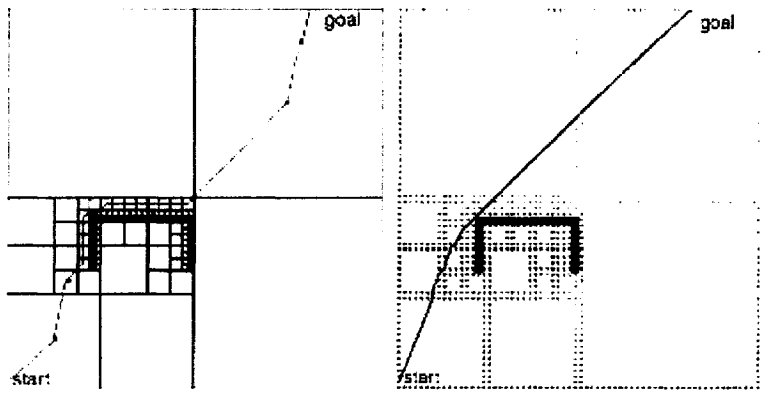


Figure 2.5: *Relationship between computer memory requirement and workspace size for matrix and octree representations (Williams and Jones 2001).*

Because of the overhead inherent to the tree-structure, standard grid-decomposition (i.e. maps of uniform resolution) actually takes less memory for grids of small sizes (e.g. 100 x 100 cells). As shown in Figure 2.5 But the gain becomes significant when the grid size exceeds 150x150 cells, or the environment has a low obstacle density.



(a) (b)

Figure 2.6: *(Left) Standard quadtree decomposition; (Right) framed quadtree decomposition (Yahja et al. 2000)*

Originally, path planning using quadtrees (or octrees) generated paths by finding the sequence of cells that led to the goal, and assembling a series of straight segments connecting all the cell centers of the sequence obtained. Unfortunately, the paths generated by these techniques contained unnecessary terms as shown in Figure 2.6.a, resulting in less optimal trajectories.

In an attempt to solve this problem, *Yahja et al. 2000* developed a variation of the quadtree algorithm called framed quadtree. The concept is to first perform a standard quadtree decomposition resulting into a hierarchy of quadtree regions. Subsequently, cells of the highest resolution are added around the perimeter of each quadtree region. This added series of cells circling around each quadtree region can be used to establish path segments that could cross these regions in any way, without necessarily passing through the center. For clarity, the added cells are referred as *frame-cells*, and the regular quadtree regions as regular *cells*. Figure 2.7 shows all possible path segments emerging from one frame-cell to another in the same cell, or in an adjacent one.

The cost of traveling through quadtree regions was no longer represented by a single value for each cell in framed-quadtrees. Instead, a function computed the cost by adding the static cost of the two frame-cells at each end of the segment, with the length of the segment. As a result, framed-quadtree offered more accurate cost estimations and shorter paths. The main drawback was that the structure could end-up requiring more memory than standard single-resolution grid decomposition techniques. This factor becomes problematic for environments with high obstacle density.

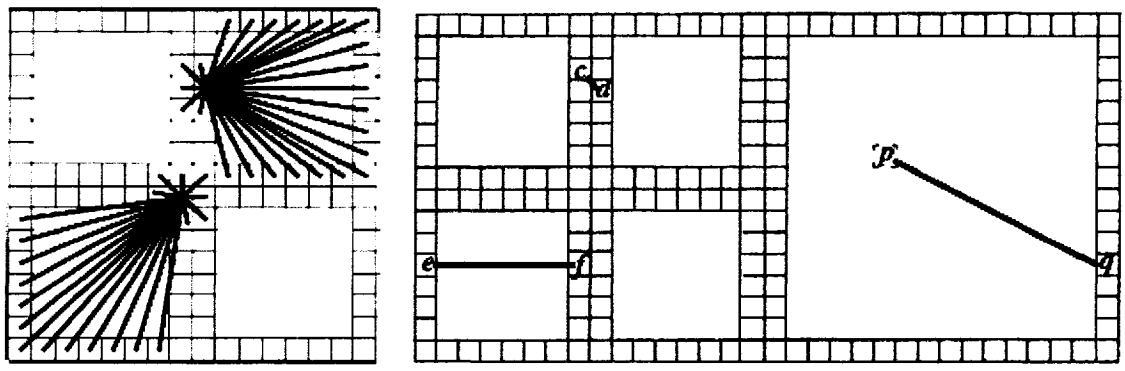


Figure 2.7: *Configuration of framed quadtree (Yahja et al. 2000)*

Another variation of the quadtree technique was developed by *Autere 2002*. Based on the concept of multi-resolution maps, the multi-size grids of quadtrees were replaced by a graph representation. Two types of edges were used, one connecting neighbouring key places at the same resolution, and another one connecting configurations of different resolutions. The main difference is that the decomposition of the search-space doesn't necessarily use the binary representation of free-cell and occupied cell. Moreover, each node can represent a configuration of any dimension. This work actually implemented this multi-resolution graph approach for a robot manipulator with 6 DOF (i.e. 6-D configurations). However, representation of physical environment is less intuitive using that technique compared to quadtree and octree representations.

2.2.4. Potential fields

Rather than finding individual paths for each possible starting position, researchers investigated methods to find families of paths that would reach the same goal through a wide range of itineraries. The main concept that emerged was the potential field (PF) paradigm. The idea of PFs is to attract the autonomous vehicle toward its goal regardless of its current position. In order to do so, the goal position is represented as a virtual positive charge, and the start position as a negative charge. Both of these charges are positioned on a map of the known environment (generally using a standard grid-map). The field generated by the two charges naturally flows from the positive charge (the start position) to the negative charge (the goal). Obstacle avoidance is achieved by positively charging the obstacles' contours, such that a repulsive force is created in their neighborhood. The force of the field is calculated at each cell of the grid-map. The robot can always find a path to the goal by following a simple steepest descent algorithm.

The simplicity and robustness of this method made it quite attractive in the field of autonomous robots. Early works of 1990s, integrated obstacle avoidance and global path planning by superimposing two fields; a global field applied to the entire map and defining the goal position; and a local one obtained by sensors feedback to push the robot away from obstacles. The rationale for such an approach was that local PFs could be generated very rapidly, making them a feasible option for real-time control. Thus, relatively simple schemes were used for PFs in order to minimize computational load. The most common one was the wave expansion algorithm (*Ratering and Gini 1993*),

which was used under various names throughout the literature (e.g. the "wavefront propagation" in *Murphy et al. 1999*; the "distance transform" in *Williams and Jones 2001*). This algorithm consists of assigning increasing cost values to the grid-map cells, such that the farther the cell is from the goal, the higher is its cost value. The algorithm proceeds by an iterative process which first assigns a zero value to the goal cell and then increments all adjacent cells to one. During each step of the process, every unassigned cell adjacent to one already having a cost value is given its neighbour's cost value incremented by one. As a result, the wave expansion PF gives an inclined surface with only one global minimum: the goal.

The main problem associated with the first generation of PFs is that they cannot avoid generating local minima where repulsive forces of surrounding obstacles balance out with attractive forces of the goal. Therefore, robots are likely to get trapped before reaching the global minimum (i.e. the goal). In an attempt to prevent such situations from happening when using the wave propagation algorithm, *Barraquand et al. 1992* proposed three techniques to escape local minima: random motion; gradient motion; and backtracking in case of dead-ends. All of these techniques were based on a graph connecting all the local minima of the PF. This approach is limited by the computational load for high DOFs, and the efficiency of the search algorithms used to find the local minima. On the other hand, *Murphy et al. 1999* avoided the problem by switching to a reactive controller (i.e. a local path planner) until the path is clear of obstacles.

Neural networks (NN) implementations were also developed to solve the local minima problem while speeding up the wave propagation process. *Kassim and Vijaya Kumar 1997* proposed a wave expansion neural network (WENN) that could be used to implement a series of PFs without spurious minima. The first one called the simple attractive potential field (SAPF) WENN did have only one minimum but didn't take into account the proximity of the obstacles (a similar implementation – facing the same problem – can be found in *Yang and Meng 2000*). The second one called the distance field (DF) WENN essentially constructed a Voronoï diagram with which standard graph-search algorithms had to be used to find a path. The third one called the boundary propagating WENN gave an alternative to the DF WENN for obtaining the connectivity of the Voronoï diagram (*Kassim and Vijaya Kumar 1999*). Although these techniques

have no restrictions on the number of dimensions of the search space, they are meant for non-holonomic vehicles (i.e. no constraints on the motion continuity) and they are still computationally intensive when considering the training of the NN and the fact that graph-search algorithms are still required to find a path.

Complete analytical solutions for the local minima problem were found by investigating a different category of fields involving continuous differential equations. The concept was to model the propagation process of a fluid flowing around obstacles from the goal to the starting point. Being continuous by nature, these equations had to be adapted to the discrete representation of the environment used with PFs. One of the first attempts was the "*diffusion method*" of *Schmidt and Azarm 1992* which used the differential equations of the diffusion process for a gas. The path between the goal and starting position could be found by following the concentration gradient. Later on, *Conolly and Grupen 1993* developed PFs based on harmonic functions which are solutions of the Laplace's Equation:

$$\nabla^2 p(\vec{r}) = \sum_i \frac{\partial^2 p(\vec{r})}{\partial^2 x_i^2} = 0 \qquad 2.1$$

Due to the properties of this solution, the path leading to the goal can be found from any starting point using a standard gradient descent algorithm. Although the absence of spurious minima was guaranteed, the first implementation of this technique was still computationally intensive, the PF having to be recomputed each time the environment changed. Discrete implementations of the Harmonic functions technique called *relaxation methods* were developed to reduce the computational load by calculating the value of the PF in a local and iterative manner (*Prestes et al. 2004*; *Prestes et al. 2001*). The two main relaxation methods are the *Gauss-Siedel update rule* and the *Successive-Over-Relaxation* rule. The first method was reported to outperform the second one (*Prestes et al. 2002*). Both methods work as discrete filters that compute a cell cost value based on the cost value of the surrounding cells. This process is similar to pixel-based filters for computer imaging.

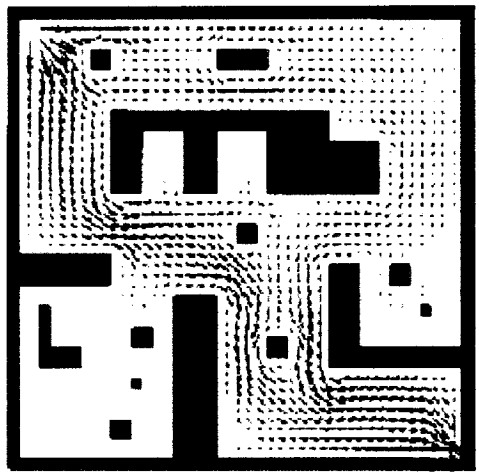


Figure 2.8: *Potential field representation using the Viscous Fluid method (Louste and Liegeois 2000)*

Following the work on the diffusion method, *Louste and Liegeois 2000* adapted the Navier-Stokes equations for an incompressible viscous fluid for a discrete map of a closed environment. Using the goal as an inlet, the starting point as an outlet, and the grid-map as the control volume, the algorithm computes currents of varying strength throughout the search-space. The novelty of this algorithm is that each cell is assigned a vector rather than a scalar cost value. Based on fluid dynamics, the fluid velocity on the obstacle boundaries is set to zero, and the pressure gradient is assumed normal to the obstacle boundaries. As a result, streamlines that are the farthest from obstacles will correspond to currents of higher velocities. The best paths can then be found by following these streamlines. As for the diffusion and the Harmonic functions methods, the viscous fluid method uses relaxation rules to compute the PF in a discrete and iterative manner that minimizes computational load. This PF has the advantage of generating smooth trajectories in terms of both velocity and acceleration without the use of additional smoothing algorithms.

Besides discrete PFs such as the wave propagation PF, and continuous PFs such as the Viscous Fluid PFs, Fuzzy Logic-based representations were also investigated to account for environment uncertainties. Since obstacles are unlikely to be all known a priori, sensor feedback is essential to update the environment representation during a mission. Because of the noise associated with these sensor readings, obstacle may be falsely identified. In order to avoid this situation, sensor fusion techniques may provide

information on the obstacle position and the levels of confidence associated with it. A standard grid-map using fuzzy values can then be used to capture the location of obstacles and the certitude of the obstacle being there (*Gundersen et al. 1996*). Membership functions can also be used to represent varying risk levels over a given area (*Soltani and Fernando 2004*).

2.2.5. Genetic algorithms

Optimization methods such as the Simplex algorithm constituted a major breakthrough in terms of finding the optimal solution for a set of linear constraints, given an objective function based on a linear combination of the control variables. However, non-linear problems cannot be directly solved using Linear Programming (LP) without being linearized in some way.

Genetic Algorithms (GA) belong to the randomized search category. Compared to the other search algorithms of the same category, they have the advantage of not requiring the gradient of the objective function, nor the constraint functions. Also, its highly randomized nature drastically reduces the chances of being trapped in local optimums. Moreover, the time complexity of GAs are usually much lower than conventional search algorithms such as A*, Dijkstra, or even the simplex algorithm when linear problems are considered.

Table 2.3 shows the time complexity of the main search algorithms considered in this literature review.

Table 2.3: *Time complexity of path planning algorithms*

Type of Algorithm	Resolution Time Complexity	Parameters
Simplex Algorithm (Linear Programming)[1]	$O\left(\frac{n!}{m!(n-m)!}\right)$	n: number of variables m: number of constraints
Network Simplex Algorithm (Linear Programming)[1]	$O\left(\frac{n!}{m!(n-m)!}\right)$ 100 ~ 200 x faster than standard Simplex Algo.	N: number of variables m: number of constraints
Interior point method (Linear Programming - Khachian and Karmarkar[1])	n^X	n: number of variables x: proportional to the number of constraints
A* algorithm[2]	$O(n * \log n) \sim O(n^2)$ Depends on the type of heuristics	n: size of search-space (e.g. number of cells in a grid-map)
D* algorithm	$O(b^* n * \log_2 n)$	n: size of search-space b : branching factor
Dijkstra algorithm	$O(n^2)$	n: size of search-space
Simulated annealing[1]	$O\left(\frac{T_0}{\delta} I * n^2\right)$	n : size of search-space T_0: initial temperature δ : temperature increment I : number of neighborhood evaluations
Wave expansion NN	$O(m^d)$	m: size of search-space d : number of DOF of robot
Diffusion method	O (T* n * r)	T: time step of diffusion process n : size of search-space r : number of iterations required by a relaxation method to obtain a steady state solution
Viscous Fluid Method	O (3* n * r)	n : size of search-space r : number of iterations required by a relaxation method to obtain a steady state solution
Genetic algorithm	O (c* p * g)	c : chromosomes size p : population size g : number of generations

[1]: see *Winston. et al. 2003;* [2]: see *Luger 2002;*

The other algorithms have been discussed and referenced in section 2.2.

Although these estimates of time complexity are somewhat general and can vary depending on the problem and the algorithm implementation, works found in the literature seem to agree that GA outperforms most search techniques (*Soltani et al. 2002; Nearchou 1999*), or gives equivalent results for the worst case (*Louste and Liegeois 2000*). Table 2.4 shows a performance comparison between the A*, Dijkstra and GAs for optimal path planning of a single vehicle.

Table 2.4: *Actual execution times for two grid sizes for Dijkstra, A* and GA (from Soltani, et al. 2002)*

Grid Size	**Actual Execution Time (s)**			
	Dijkstra	*A* with straight-line heuristic*	*A* with cumulative line-nodes*	*Genetic Algorithm*
40	7	7	22	5
80	120	120	456	51

Developed in the 1970s by John Holland, GAs aim to emulate the biological evolutionary process taking place in nature. This process can be applied by analogy to various domains, which makes GA's a very flexible technique for general problem solving. GAs can be divided into five elements (*Nearchou 1999*):

- *Representation mechanism*: Each potential solution must be encoded into some numerical format supporting the modification and selection operations inherent to GAs. This generally implies fragmenting the solution into sub-units representing genes that can be interchanged between chromosomes (i.e. individual solution). Originally, chromosomes were set to have a predefined constant number of genes, however variable-size chromosomes have shown better performances for certain types of problems.
- *Initial population*: A set of potential solutions must be generated before the GA can actually start. The convergence speed of the algorithm as well as its capacity to find the global optimum strongly depends on the initial population. Population diversity (through random individual generation) reduces the chance of converging to a local optimum, while specialization of the population in the neighborhood of a partially known solution will increase the convergence rate.

- *Evaluation mechanism*: Individual solutions must be classified according to their fitness level. The fitness function is based on predetermined criteria which depend on the problem specificities. It represents the objective function that must be optimized.
- *Operators*: GA's proceed by iteratively applying a series of genetic operators to the population of potential solutions. Each operator emulates some evolutionary process occurring in nature. The most common operators are:
    - Crossover: Two parent chromosomes are randomly divided into a predefined number of parts. Each part of the first chromosome is recombined with parts of the other parent chromosome. The newly generated chromosomes are called the offspring. The parent chromosomes remain unaffected by the operation. Variations of the crossover operation include single-point separation, two-point separation, cyclic crossover and uniform crossover.
    - Mutation: One gene is randomly altered in a chromosome. Mutation is equivalent to a change from '0' to '1' for binary genes (i.e. one gene is one bit). The mechanism selecting the chromosome to be altered is probabilistic. Although the probability generally remains low for all chromosome, it is usually related to the chromosomes fitness score.
    - Insertion: a new gene is randomly inserted within a chromosome according to the same probabilistic selection mechanism as the mutation operators.
    - Deletion: a gene is randomly deleted from a chromosome based on the same probabilistic selection mechanism.
    - Swap: two chromosomes are divided in two, each half being interchanged with the other chromosome's halves. As for the mutation, deletion and insertion operators, swap is executed based on a relatively low probability which is defined a priori according to some custom function.
    - Selection: This operation simulates the "survival-of-the-fittest" phenomenon occurring in natural evolution. After all genetic operations have been carried out, the GA proceeds to a selection process which determines the individuals that will constitute the next generation's population. The basic approach is to eliminate a fixed number of the weakest individuals, corresponding to the number of new individuals generated by the crossover operation. The

alternative is to assign a probability to each individual, which is inversely proportional to their fitness level. A fixed percentage of the population is then eliminated based on this probability distribution. This technique called the *roulette wheel*, (also referenced as *Monte Carlo*, or *fitness proportionate selection*) is based on the possibility of having weaker individuals, which may possess genes that would contribute to more optimal solutions (*Luger 2002*).

Additional operators have been used for the special case of the path planning problem. Implementations of GAs for such problems generally involve a representation were genes are waypoints, and chromosomes are series of genes defining a potential solution path. *Zheng et al. 2004* used the following genetic operators in addition to the ones specified above:

- Smooth mutator: Sharp turns between two points along the path are removed by inserting two new points and deleting the one making the sharp corner.
- Fixed vector mutator: In the case where an vector of approach is specified between the goal and the waypoint preceding it, this genetic operator modifies the latter to satisfy the constraint.

- *Control parameters*: They include population size, maximum number of generations (i.e. time limit), type of crossover, selection process, fitness function, and the fixed probabilities associated with the genetic operators described above. The combination of the control parameters will ultimately define the performance of the GA. Unfortunately, there are no formal techniques for the optimization of these parameters. Tuning is generally based on some heuristic rules.

Several researches have successfully used GAs to solve the path planning problem. *Soltani et al. 2002* used a standard grid-map with a fuzzy potential field to evaluate the cost value of each cell along a potential path. Each path was represented as a chromosome which was encoded as a string of integers designating a series of cells within the grid-map. The fitness function weighted the candidates' path length and exposure to hazardous areas (using fuzzy values). Selection was based on the roulette wheel method. *Tian and Collins 2004* applied GAs to find a path for a robot manipulator moving between obstacles. The selection method and the chromosome representation

were the same as for *Soltani et al. 2002*, excepted that potential paths (chromosomes) had a fixed number of points (genes) defined by 12-bit binary numbers. The genes' integer value corresponded to parametric values of a points on hyper-surfaces defined within the search space. The algorithm found sets of points in real-space rather than C-space. Trajectories were obtained by using a Hermite Cubic Polynomial to fit a smooth curve through all the points constituting the path considered.

Rather than finding a sequence of points, *Nearchou 1999* developed a GA to find the best sequence of actions required to reach the goal. Designed for wheeled robots moving in a grid-map representation of the world, the possible actions were to move forward or turn left by increment of one cell. These two actions were implemented as a binary numbers , '1' for move forward and '0' for move left, which encoded chromosomes of variable length. Fitness was evaluated based on path feasibility, positional error at the goal defined by the difference between the position reached by the candidate path and the desired goal position, and the length of the path. Selection was based the roulette wheel method. *Smierzchalski (1999)* implemented a GA to solve the problem of collision avoidance of ships in navigation traffic areas. Based on the assumption that the speed and heading of approaching targets were constant, the algorithm aimed to find a series of points defining a path which would avoid collisions, together with the ship's speed required for each section of that path. The algorithm integrated dynamic constraints based on the speed, course and bearing of the moving targets in order to keep a safe approach distance. The speed component of the trajectory used fuzzy logic to define its amplitude. Fitness of the potential candidates was determined based on path length, trajectory smoothness, and the time needed for covering the trajectory. The time limit in on-line mode was set to 56 seconds of CPU time.

As a last example, *Zheng et al. 2004* developed a GA to solve the Task Allocation Problem for a fleet of UAVs given the constraint that all vehicles had to have the same estimated time of arrival (ETA). Global planning was performed using Digital Elevation Data in a 2-D grid-map representation of the search space. Dynamic elements were integrated using an Environment Information Look-Up table which was continuously updated during a mission. Individual path planning was implemented using variable length chromosomes representing paths with each gene being equal to the real-world

coordinate of a waypoint. Fitness was evaluated based on path length, threat exposure, and altitude. Constraints included minimum and maximum path length, flying height, turning angle, climbing or driving angle, ETA, approach angle to the goal, and collision avoidance. Optimal path could be found in approximately 60 generations, while feasible paths could be generated in less than 1 second.

Chapter 3: Controller Overview

This chapter will describe the fundamental elements of the intelligent controller's implementation. Since the TAP can take many forms depending on the type and number of constraints considered, its basic definition is given as a reference at the beginning of section 3.1. The rest of this section presents the constraints considered in this work as a mean to describe the fundamental capabilities of the controller, and to help the reader understand the effect of each constraint on the TAP. Section 3.2. gives the definition of a series of concepts which are essential to understand the controller's internal representation. Section 3.3. introduces the structure of the software implementing the intelligent controller. Finally, section 3.4. presents the top-level algorithm defining the procedure followed by the controller in order to solve the TAP.

Note that the controller is referred interchangeably as *the system* and *the controller* throughout this thesis.

3.1. The TAP and its constraints

For this work, the basic form of the TAP will consist of finding the optimal objectives distribution among a fleet of UAVs with all objectives being achievable by any UAVs of the fleet. Each objective consists of a single waypoint. The mission length is unspecified, as well as the order in which these objectives must be achieved. The only external elements to be considered are static obstacles and atmospheric conditions based on the International Standard Atmosphere model. The wind is assumed negligible. Many constraints can be added to the basic form of the TAP. In an attempt to provide a flexible framework which can integrate most constraints seen in the literature and their possible variations, six general categories have been considered. They are described in Table 3.1. Conformity of the trajectories to the UAV dynamics, as well as collision avoidance between UAVs and static obstacles, are both considered to be implicit requirements of the basic form of the TAP. Therefore, they are not considered as constraints, and thus, do not appear in the list shown above.

Table 3.1: *TAP Constraints*

#	Constraint Type	Description
1	Continuous actions	An objective may consist of continuously performing some action such as surveying a specified area.
2	Specific Sequence	A series of objectives may have to be performed in a specific order.
3	Specific Scheduling	Certain objectives may have to be performed at specific time.
4	Tactical options	A specific approach vector may be specified for an objective
5	Simultaneous TOA (time of arrival)	An objective may require that all assigned UAVs arrive at same point in space and time.
6	Strategic options	Certain flight attributes may be specified. The main ones are: - Desired Altitude: high -detectable/ low -less detectable - Desired mission length: fast -hazardous / slow –safe - Desired obstacle proximity: close -covered / far -exposed

Constraints of types (1) and (2) represent hard constraints which directly impact the bounds of the search space. For instance, the Task Allocation Process (TAPr) will not have to optimize the sequence of objectives assigned to a UAV if this sequence has already been specified in the mission statement. In that case, only the optimal path linking this sequence of objectives needs to be found. Thus, constraints of types (1) and (2) actually reduce the size of the basic TAP's search space.

Table 3.2: *The Altitude, Speed and Obstacle distance (ASO) Strategy*

ASO Strategies		
Altitude	**Speed (Mission length)**	**Obstacle Distance**
Minimal low detection risk / hazardous distance from ground	*Slow:* near stall speed	*Minimal* full cover mode / hazardous distance from obstacles
Low partially covered	*Endurance* maximize flight time	*Medium* partially exposed
Medium partially exposed	*Range* maximize flight range	*Large* fully exposed / safe distance from obstacles
High high detection risk / safe distance from ground	*Fast* near maximum speed	-
Ceiling maximum altitude of the vehicle	*Escape* maximum speed	-

On the other hand, constraint types (3) to (5) will only reduce the number of valid solution within the search space. Constraint type (6) does not influence the search space or the solution space. It simply changes the position of the optimum within the solution space by altering the costs associated with the criteria defining the optimality of a path. As mentioned above, three physical parameters are considered in this case, namely altitude, speed and obstacle distance, as shown in Table 3.2. Since the speed, obstacle distance and maximum altitude may change depending on the atmospheric conditions and/or the characteristics of the aircraft, a more instinctive approach is provided to the operator. The minimum and maximum values of each physical parameter are automatically calculated by the system so that the user only has to specify a relative value between those extremes. This approach is referred as the ASO strategy in the next chapters of this thesis. The description of each parameter is given in Table 3.2 and their physical meaning is given in Table 3.3.

Table 3.3: *Physical values associated with the ASO strategy*

Altitude		**Speed (Mission length)**		**Obstacle Distance**	
Minimal : 0 m		*Slow*:	$SF*V_{stall}$	*Minimal*:	$SF*L_{max}$
Low :	1/5 of max alt.	*Endurance*:	$V_{endurance}$	*Medium*:	4*ErrGPS
Medium : 1/2 of max alt.		*Range*:	V_{range}	*Large*:	8*ErrGPS
High :	4/5 of max alt.	*Fast*:	$9/10 * V_{max}$		-
Ceiling:	SF * max alt.	*Escape*:	V_{max}		-

In order to use the ASO strategy, the operator simply needs to specify a rating from one to ten for each parameter, 1 representing the "minimal" level (or "slow" level for speed) and 10 representing "High" Altitude, "Escape" speed, and "Large" obstacle distance. These ratings are directly converted into numerical values by calculating a standard weighted average based on the values defined in Table 3.3, as shown in Appendix-B for mathematical descriptions of V_{stall}, $V_{endurance}$, V_{range} and V_{max}. The numeric values represent the desired altitude, speed and obstacle distance to be kept during the mission. The system will use these values as guidelines to generate trajectories, and as reference to compute the deviation of these trajectories from the optimum. Deviations may occur when the trajectory generator has to adjust some of its parameters to avoid obstacle or satisfy other constraints.

3.2. Controller's representation of the contextual elements of a mission

The contextual elements of a mission define how concepts such as *objective* and *resource* are implemented and understood by the system. Interpretation of such concepts is not unique, and some may be more adequate than others depending on the type of application considered. Focusing on flexibility, the following definitions aim to provide a general framework which can adapt to a wide range of missions involving aerial vehicles. The following entities define the basic data structure on which the TAPr is based:

- A *Mission* is defined as a set of objectives, a set of constraints, and a resource.
- A *Resource* is a UAV characterized by
  - Dynamic characteristics.
  - Onboard equipment defining which type of action can be performed.
  - Autonomy: Fuel capacity [kg] and Specific Fuel Consumption (SFC) [kg/N hr].
- An *Objective* has the following characteristics:
  - The action to be performed.
  - A type: Simultaneous; Sequence; or Single.
  - A sequence of one or more waypoints, referred as "*goals*" (or sub-objectives), which corresponds to physical locations in a global reference frame.
  - A group-identification number indicating to which sequence the objectives belongs. This number is set to zero if the objective is not of the *Simultaneous* or *Sequence* type.
  - Specified time of arrival (ToA); set to zero if not required.
  - Specified vector of approach (VoA); set to the null vector if not required.
- An *Action* has the following characteristics:
  - Type: basic; combined; or special
  - Completion time
  - Fuel consumed
  - Cargo mass change
  - Required equipment type(s)

Regardless of its number of goals, an *objective* that is unconstrained by any other objectives is considered as "Single". The fact that it may have several waypoints representing goals only implies that the action to be carried on for this objective must be performed continuously for all goals. A typical "continuous action" would consist of carrying a surveillance mission between a set of particular waypoints. The "Sequence" type represents an objective that must be executed in a specific order with respect to some other objectives. An additional characteristic is then used to specify the list of objectives associated to the corresponding sequence. The ToA is used to specify the order in which those objectives should be performed. The "Simultaneous" type represents an objective that must be reached at the same time by several UAV's. The system is dealing with that constraint on an individual basis for each UAV, using the ToA value to compute the deviation between the desired time of arrival and the actual one predicted by a given trajectory. It is must be mentioned that if an objective needs to be performed more than once during a mission, different instances of the same objective must be created.

An *action* is defined as the utilization of some onboard piece(s) of equipment. It is assumed that an action requires all the attention of the UAV controller, and thus, cannot be performed in parallel with some other action. This assumption is required to provide a consistent definition of an objective. Otherwise, nothing would prohibit a UAV from attempting to perform two objectives at the same time. This would result in scheduling problem as well as contradictory path plans. Additional elements would then need to be added to the data structure defined above, making it less efficient in terms of flexibility and computational speed.

At the present stage, *actions* and *equipment types* are only implemented as a "hook" for future system development. The current version of the program treats these variables only as optional information for which memory has been allocated, but no functions have been written. However, this information will be essential when a global planner working at the fleet level will be implemented.

The concepts mentioned above in this section correspond to the high level understanding of the system. The following definitions give the basic blocks supporting the implementation of these contextual elements:

- An *Objective waypoint* is a set of three coordinates representing a position in space through which a UAV has to pass. Objective waypoints are specified by the operator of the system in the mission statement.
- A *Transitional waypoint* is an intermediary waypoint through which the UAV does not necessarily have to pass. Transitional waypoints are generated by the GA and the operator has no control over them.
- A *Path* consists of a sequence of objective and transitional waypoints connecting a starting position to a final one. Each sequential pair of waypoints is called a segment.
- A *Trajectory* is an instance of a path for which position, and velocity profiles are defined as functions of time. Sharp corners occurring at the intersection of pairs of segments are replaced by smooth curvilinear profiles taking into account the aerodynamics of each UAV.
- A *Profile* is a function mapping a physical attribute (i.e. time, position, or velocity) to a parametric representation of a path. As such, the profiles that fully defines a trajectory can be though of as a set of mapping functions of the form: {Time(v); Position(v); Velocity(v); | $v \in [0\sim1]$}.

Now that the fundamental elements defining the system's conceptual understanding of a mission have been presented, section 3.3. briefly introduces its structure and the design strategy behind it. Section 3.4. shows the top-level algorithm defining the procedure followed by the controller in order to find a solution for the TAP.

3.3. The controller's hierarchical structure

The approach followed in this work consists of developing a portable software package that can generate valid solutions in a short period of time while providing increasing level of optimization for increasing time intervals. In the attempt of providing both flexibility and efficiency, this software package is meant to take full advantage of parallel processing hardware without depending on it in the eventuality where such hardware is unavailable. According to these guidelines, the software is coded in C++ but

uses only C routines. The program is mainly procedural and was designed for time performance rather than code reusability.

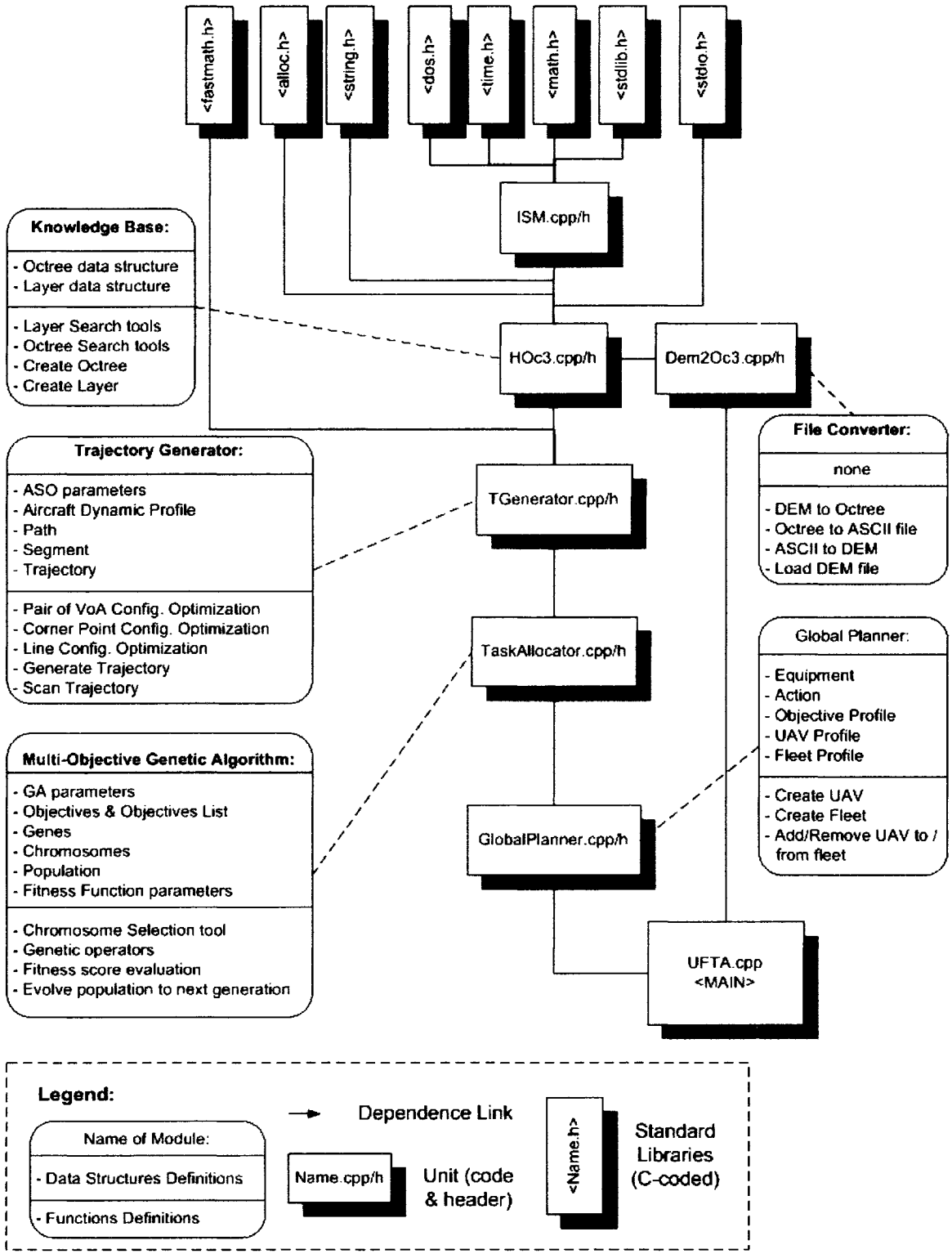


Figure 3.1: *System architecture*

The software's modules are regrouped under different units (i.e. pair of header (.h) and code (.cpp). files). The files structure aims to implement a modular architecture with common interfaces. Figure 3.1 shows the files' hierarchy with the corresponding modules that each unit implements.

The software's structure has been designed to integrate a top level planner that would operate at the fleet level. Several data structure definitions referring to fleets of UAV's can be found in the code. Since the scope of this work was limited to a single UAV, the global planner's interface between the UAV level and the fleet level is left for future development. This is the reason why the user's module (i.e. the UFTA.cpp file) interfaces with both, the Global Planner and the Task Allocator modules as defined in Figure 1.1. Also, complete integration of the dynamic layers still needs to be done by interfacing a "variables identification" module with the "Fitness score evaluation" functions in the MOGA module. Note that dynamic layers using the same variables as the wind and TPD layers can readily be used in the current version of the system.

3.4. User level algorithm

The user interface being out of the scope of this work, the modules constituting the controller consist of a library of functions that can be integrated to set up a mission, and find the solution of the corresponding TAP. The flow chart shown in Figure 3.2 represents the top level algorithm at which all modules are integrated to form what has been referred to so far as the *system*. The performance and limitations of that system are discussed in Chapter 7: .

The procedure followed by the controller to solve the TAP was implemented in 7 steps. The first one requires the user to specify the desired ASO strategy, as defined in section 3.1. The second step is required to define the type of UAV that will be used for the mission. All aerodynamic characteristics must be specified as well as the fuel tank capacity and the engine(s) maximum thrust. The third step consists of defining the physical environment in which the UAV will fly. This environment is made of two elements, the atmospheric conditions and the terrain relief. All information are internalized in a hybrid octree knowledge base which will be described in Chapter 4: . The fourth step is used to define the mission profile which specifies different destinations as well as the constraints imposed by the mission on the UAV's flight plan. Step five

initializes the genetic algorithm that will search for a valid solution of the TAP, or in other word, a valid flight plan. Step six corresponds to an iterative process where previously tested candidate flight plans are reused to find better ones. This process of reusing previous solutions to find new ones is referred as the *evolution* process. The eighth and final step consists of stopping the evolution process after a predefined period of time, and to select the best solution found so far.

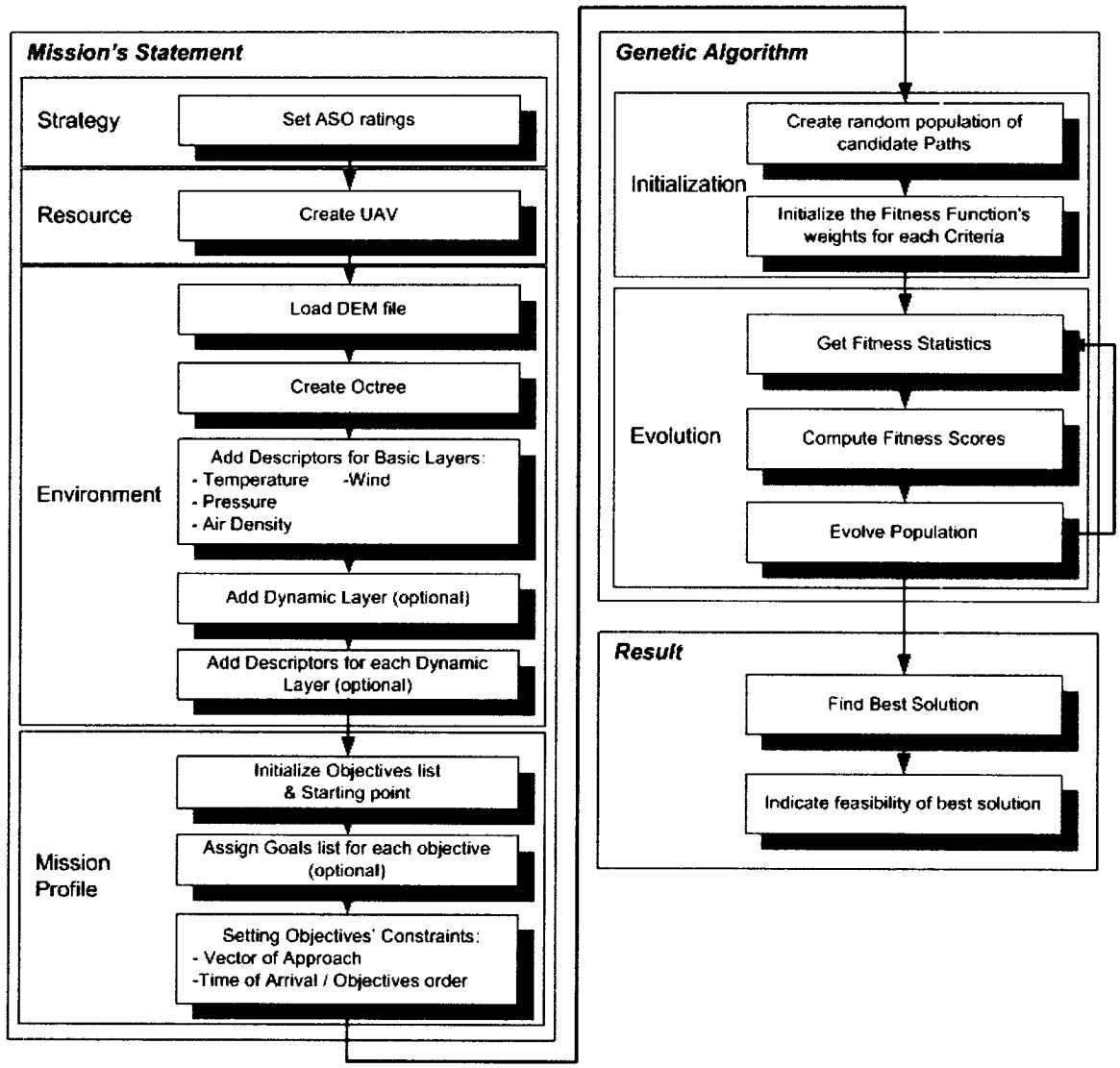


Figure 3.2: *System's algorithm at user level*

Now that the structure and logic of the intelligent controller have been presented, the following chapters give a description of its architecture. The three modules that will be described are shown in Figure 1.1, and correspond to the *Hybrid Octree Knowledge Base*, the *Trajectory Generator* and the *Multi-Objectives Genetic Algorithm.*

Chapter 4: Hybrid Octree Knowledge Base

This module is used to provide a common internal representation of the world to UAV's. New information can be added and shared on-line using their data link module. The information compiled in the knowledge base determines to a large extent the cost of every path evaluated for the TAP. Thus, this world representation is the foundation on which the GA can assess the optimality of candidate trajectories aiming to solve the TAP. Since the type of information required to assess the fitness of a path may vary depending on the mission and the environment, a *multi-layer metric map* was implemented to provide the UAV's with a common internal world representation. The map is built based on Digital Terrain Elevation Data (DTED) of the regions in which the UAV may fly for a given mission. Also referred as Data Elevation Maps (DEM) in the literature, the DTED can be constructed using satellite photos or other topographical data of the region. The space described by the DTED is then converted to a 3D array of discrete cubic volumes referred to as cells. The resolution of the internal map will determine the level of discretization of the real-world, and thus, the size of the volume represented by every cell. The term "metric map" simply represents the fact that the global coordinates of the space represented by each cell can be found directly from the position of that cell in the array.

4.1. The hybrid octree concept

As described earlier in section 2.2.3. , the octree concept is to start with a 3D map and divide it equally into eight cells. Each cell is then checked for obstacles (i.e. terrain elements). If a cell contains only an empty space, it is set as "free" and is no more sub-divided. On the other hand, if a cell's space is completely filled by an obstacle, it is set as an "obstacle". Like free cells, an obstacle cell stops being sub-divided once it has been set as such. If a cell is only partially occupied by an obstacle, it is set as a "grey" cell which is to be sub-divided in the next iteration of this discretization process. Once the maximum number of sub-divisions (i.e. max octree depth) is reached, the process stops and all remaining grey cells are set as "obstacle" cells. This process is illustrated in Figure 4.1.

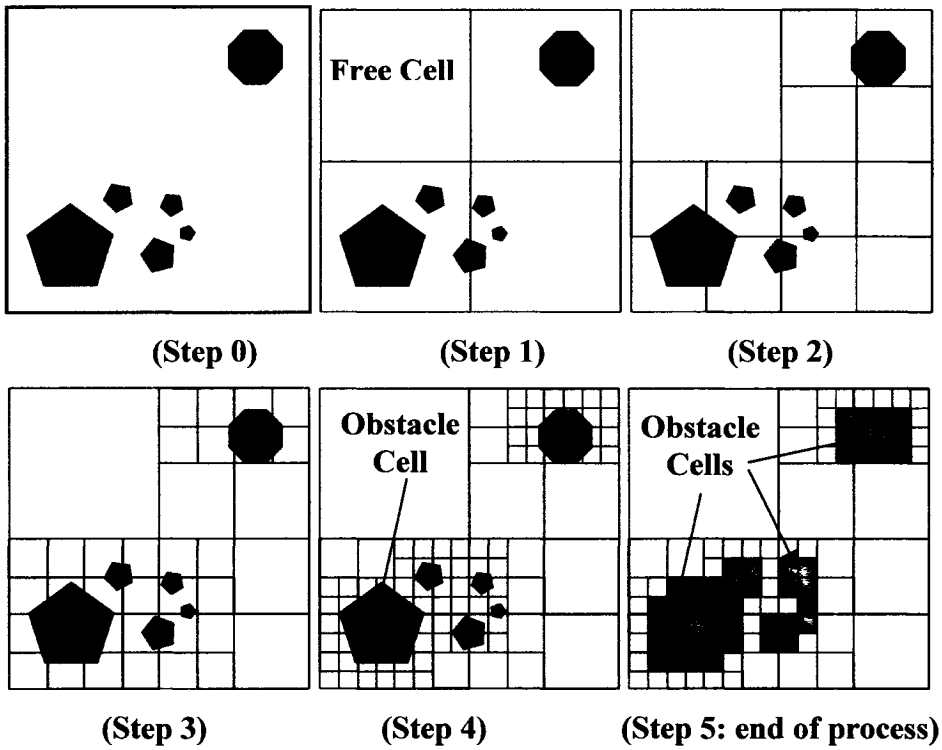


Figure 4.1: *Discretization process of an octree structure in 2D*

Each step is added to a structure referred in computer science as a *tree*. The root of that tree corresponds to one cell encompassing the entire map, and is associated to a depth set to zero by convention. Every division or "step" as shown in Figure 4.1, represent one level in the tree. The state of each cell for each level is stored in the tree. Figure 4.2 represents the data structure of an octree mapping a 3D area. The black boxes represent obstacle cells, white boxes represent free cells, and grey boxes are for grey cells. This figure clearly shows how the subdivision process expands grey cells while stopping further sub-divisions for free and obstacle cells.

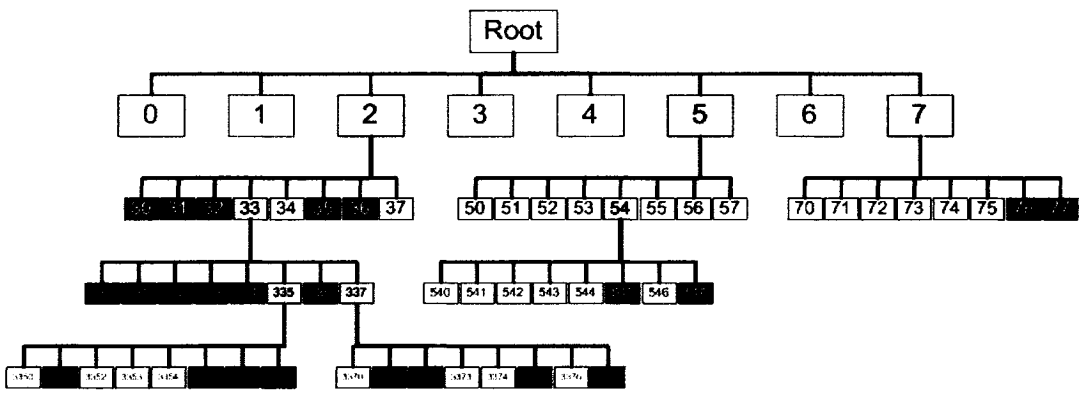


Figure 4.2: *Representation of the data structure of an octree*

The grey cells constituting the overhead of the octree are essential to reconstruct or update the map in case new information needs to be integrated to the structure (e.g. adding newly detected obstacles on the map). For that reason, grey cells are kept even after the end of the discretization process. This is one of the differences with the standard approach described in the literature and discussed in section 2.2.3. Keeping the overhead of the tree also ensures that any cell can be accessed through a search algorithm with a time order less or equal than the maximum depth of the octree.

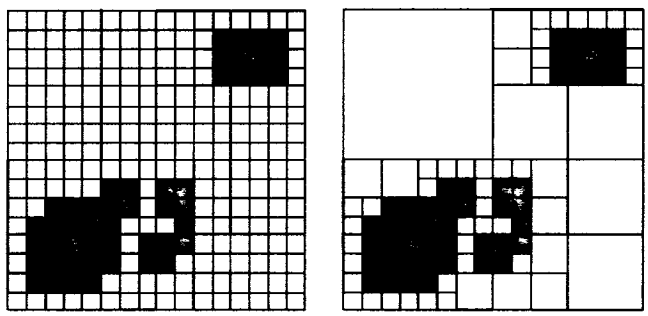


Figure 4.3: *Comparison between the standard discretization process (left) and the octree (right)*

The advantage of octree structures is that they can represent large open areas with much less cells than in a standard discretization process would do, the latter subdividing the entire map into a number of cells corresponding to the highest resolution required. Figure 4.3 shows the difference between the two discretization processes. The inherent flaw of the octree approach is the over-discretization of spaces near obstacles when the algorithm is run until a complete binary representation of the world is obtained. That is, if the entire space is allowed to be divided into "free" or "occupied" cells without limiting the depth at which the algorithm must stop.

4.2. Standard map cells

In order to cope with the over-discretization problem of octrees near obstacles, this work uses a hybrid approach which combines standard discrete maps and an octree map. The octree discretization process is run until a predefined resolution is reached, limiting the depth of the octree and leaving some cells partially occupied by obstacles (i.e. grey cells). The space represented by these remaining grey cells is then discretized using a standard map (SM) with a constant resolution (i.e. cell size). In order to fully take

advantage of this approach, the resolution of the standard map has to be higher than the maximum resolution associated with the octree. If the SM resolution is higher by a factor of n compared to the octree's maximum resolution, the algorithm avoids the computation of $2^{n/2}$ cells. This fact holds under the assumption that the octree discretization would ultimately result in the same number of cells as the SM discretization, which is a valid assumption for spaces near obstacles. Thus, the hybrid approach is the most efficient one, taking advantage of the two alternatives for open spaces and near-obstacle spaces.

The standard map structure is much simpler than the octree. It is composed of only *free* and *obstacle* cells, and does not take in account the extent at which a cell is occupied by obstacles. As soon as some portion of the cell contains an obstacle, the entire cell is tagged as an obstacle cell. The resolution of all standard maps contained within the octree doesn't have to be the same. The size of the octree cell and the number of grid cells within the SM will implicitly determine its resolution. For practical reasons, the physical space represented by one SM grid cell should not be smaller than the space required by the vehicle to manoeuvre. Higher resolutions, and thus, smaller grid cells, will only increase the memory requirement and processing time while providing no additional information on whether the vehicle can manoeuvre between obstacles for a given trajectory.

In order to facilitate the use of SM for the operator, areas in the non-discretized map can be pre-selected by entering two sets of physical coordinates defining the ends of a virtual box which encompasses this area as illustrated in Figure 4.4.

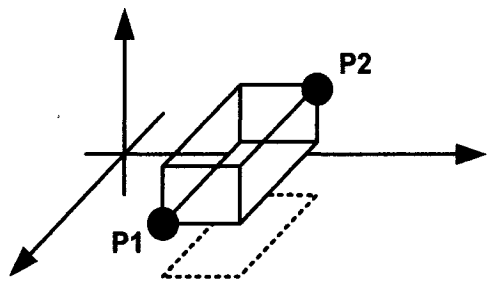


Figure 4.4: *Identifying an SM area based on two set of physical coordinates*

Once specified, these two points are processed by the system and the cells included within that area are automatically linked to a standard map. The discretization process of the SM area is performed independently of the rest of the octree structure.

4.3. Multi-layers map

Based on the mission profile, a list of all factors relevant to the path planning module is established. A layer is created for each of them which results in a 3×N-dimensional map for N factors considered. These factors include the atmospheric temperature, pressure and air density for the basic layers which will be described in section 4.3.1. The dynamic layers may include additional user-defined factors (e.g. radar exposure). Each layer is evaluated locally at the cell's level. The purpose of these layers is to provide an evaluation tool that can assess the suitability of a path based on a series of pre-specified criteria. The number of layers can be expanded or decreased as desired. This requires the explicit intervention of the human operator supervising the UAVs.

In its most basic form, the system's map of the environment consists of two layers; the atmospheric condition layer and the static obstacle layer. These two layers are required before any path can be generated. A *No Fly Zone* layer is usually used in most missions as a third layer but the information provided by this layer is not mandatory for the system to work. While the atmospheric and static obstacle layers are fundamental, the *No Fly Zone* layer can be added (or removed) dynamically during a mission. For this reason, layers are classified as either *basic*, or *dynamic*. All dynamic layers use the same data structure, providing a generic way of representing a wide range of situations.

4.3.1. Basic layers

The system has five build-in basic layers, four for the atmospheric conditions and one for static obstacles. The first three layers represent environmental conditions by using sets of polynomial equations which are valid only within a limited area (referred to as area of influence). These sets of polynomials are called *descriptors* and must be entered manually by the operators. As for standard maps, the operators simply has to specify the area for which each descriptor is valid and the system will automatically allocate them to the proper cells. Descriptors with overlapping areas are dealt with by computing a weighted average of their output based on their relative volume within the cell of interest (e.g. a cell crossed by a UAV's trajectory). Figure 4.5 represents two overlapping descriptors with different areas of influence for a given cell. Their relative weights shown in the figure corresponds to the area they cover within that cell.

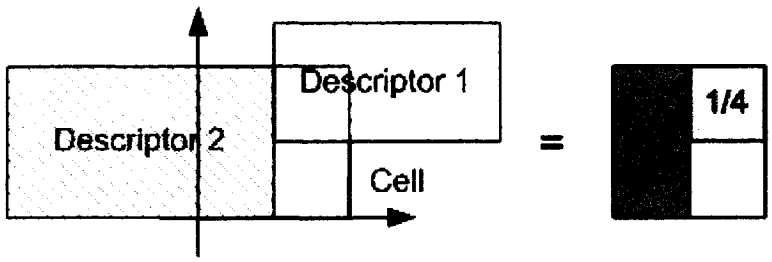


Figure 4.5: *Relative weights of overlapping descriptors with one cell*

The *atmospheric conditions* layers encode two aspects of the environment:

- *Wind speed and direction*

  They are described as *vector fields* based on a combination of three polynomials in X, Y and Z. If the wind varies a lot in direction and magnitude within the space of a cell, the polynomials may go up to the 6^{th} degree to approximate accurately the changes occurring within that space. On the other hand, a vector of constant direction and magnitude (i.e. zero-degree polynomials) may be sufficient for cells representing small areas on the map.

- Air pressure, temperature, and density (TPD)

  In contrast with the "wind layer" which uses sets of three polynomial equations to describe vector fields, the TPD layers uses only one polynomial with the altitude as its unique variable. In order to avoid confusion with vector fields, those single variable polynomials are referred as *density functions*. As for the wind layer, cells representing large spaces may require a polynomial going up to the 6^{th} degree, while cells of smaller size may only require a constant value.

In order to optimize memory usage, all polynomial equations used in the octree are stored only once. Cells simply include a reference to the proper equations rather than the equations themselves. In order to illustrate the concept of the descriptors which hold these polynomial equations, Figure 4.6 shows the full sets of characteristics required to set such descriptor for a layer using vector fields (the wind layer in this case). A descriptor using only a density function (e.g. the TPD layers) has the same characteristics as shown above, except that it uses only one set of coefficients rather than three.

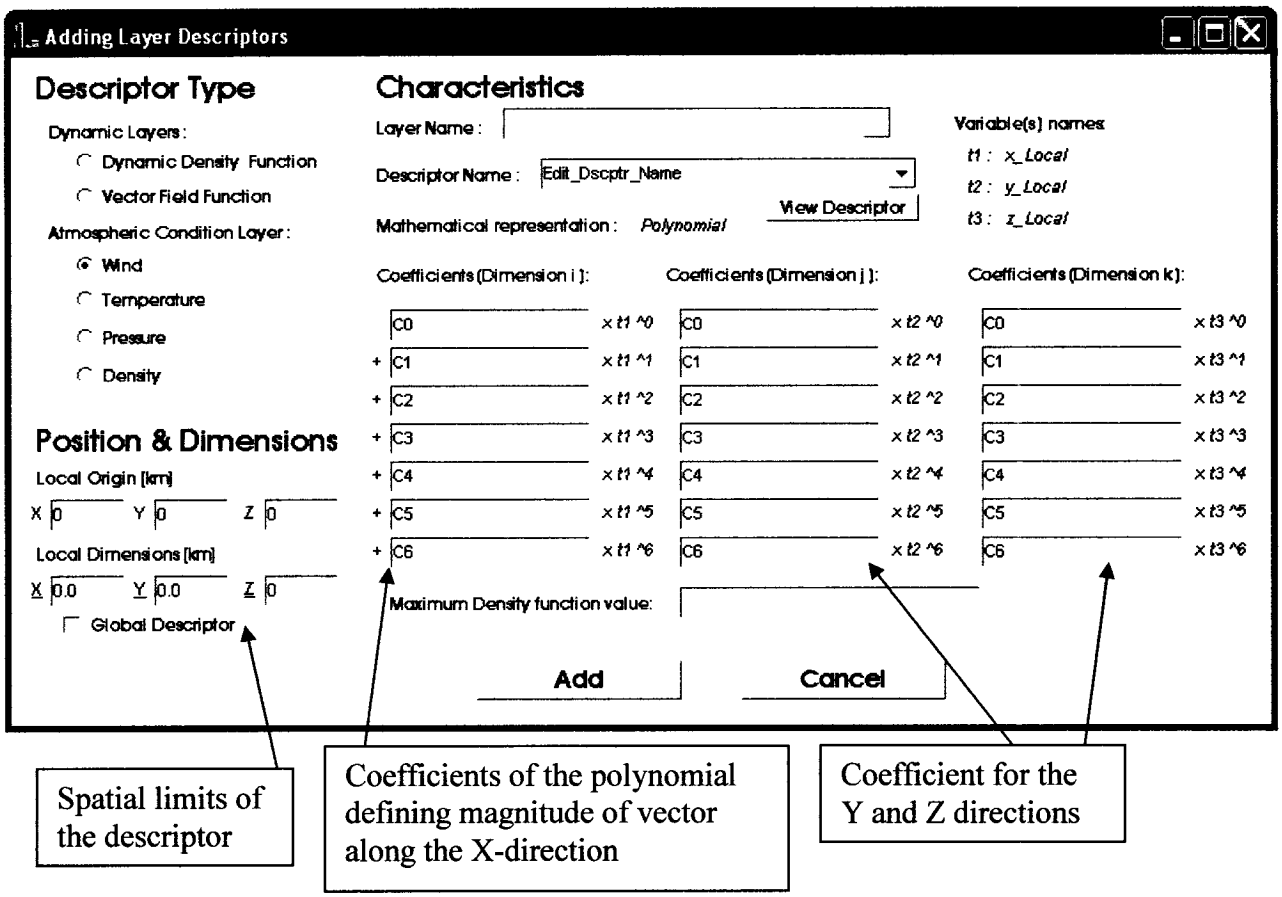


Figure 4.6: *Characteristics of a "Vector Field" descriptor*

The *static obstacle* layer integrates all non-moving obstacles such as terrain, buildings, towers, etc. Its purpose is to encode the Digital Terrain Elevation Data (DTED) of the regions. The system uses the DEM file format as its standard DTED (*Gesch D. and Greenlee S., 1997*). The basic data structure for each cell has the following parameters:

- The global or local position of the cell's origin in space. This position is set as the physical location of the South West corner point of lower altitude, in GPS coordinates (i.e. longitude, latitude and altitude) according to the WGS84 standard (*EUROCONTROL and IfEN, 1998*).
- The depth of the cell in the octree: Combined with the size of the overall map, this parameter is sufficient to completely define the size of each cell's sides:

$$S_{Ci} = N_i / 2^h \qquad 4.1$$

with N_i, the total side length of the map along the 'i' direction (i.e. x, y or z).

h, the depth in the tree with h=0 at the root.

$S_{C\,i}$, the side length of the cell along the 'i' direction.

- The cell type: free, obstacle, grey or SM.
- A "parent" pointer that roots back to the cell of greater dimension encompassing the current cell.

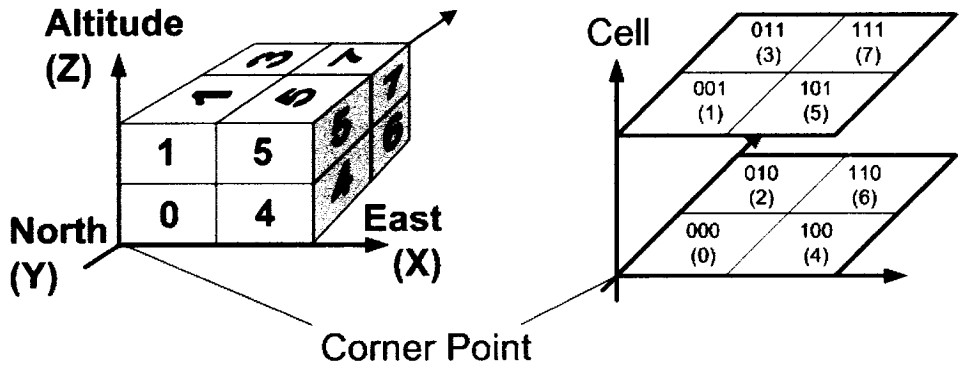


Figure 4.7: *Global position reference for an octree cell and its sub-cells*

- Eight child pointers that link the current cell to the 8 sub-cells equally dividing its volume. The indexing standard of the sub-cells is shown in Figure 4.7. The following conditions apply to the cell and its children:
  i. If the cell corresponds to a free or obstacle cell, the children are set to zero. In that case, the cell corresponds to a leaf cell.
  ii. If the cell corresponds to a SM cell, the first pointer is used as a link to the standard map, while the others are set to zero.
  iii. If the cell is a node cell, the 8 pointers are used to link the sub-cells dividing its volume.

4.3.2. Dynamic layers

A dynamic layer can represent regions of space that can either be favourable or detrimental to the UAV performance in achieving its mission. These zones can be used to represent various situations ranging from risky military zone, to safer regions where the UAV can take advantage of its environment for protection or detection avoidance.

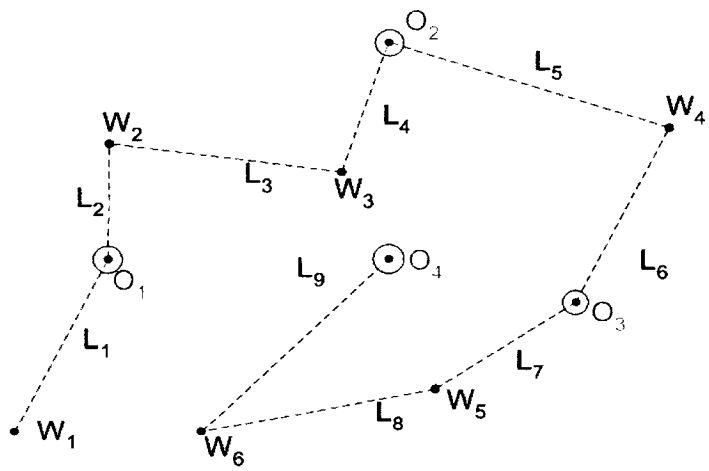
In mathematical terms, a dynamic layer implements attractors or "repulsors" in the search space. Attractors are created by adding a negative weight in the evaluation function that decreases the cost of a path as it passes close to the designated regions. In a similar way, a positive weight can be added in the evaluation function to increase the cost of a path, making it less attractive as a candidate solution. For the sake of generality, cost

increasing regions are referred to as negative attractors while cost decreasing regions are referred to as positive attractors. The term "attractor" simply represents the fact that a region exerts some influence on the path of the UAV under some arbitrary criteria implemented by a human operator. As for the atmospheric condition layers, attractor can be implemented with descriptors which use vector fields or density functions. The operator can add manually a maximum of 5 dynamic layers, for which a maximum of 100 descriptors can be set. These limits can be tuned within the system, but increasing them unnecessarily also increases the memory requirements and could decrease the system performance. In contrast with the TPD layers and the wind layer, all dynamic layers must use normalized function(s); that is, functions with an output ranging from 0 to 1. Thus, polynomial functions considered in dynamic layer descriptors must be bounded by a maximum value, which is used to divide the magnitude of these functions output. The field containing this value is shown in Figure 4.6. in front of the "maximum density function value" label (although this field is deactivated in the example shown above, because the wind layer is not a dynamic layer).

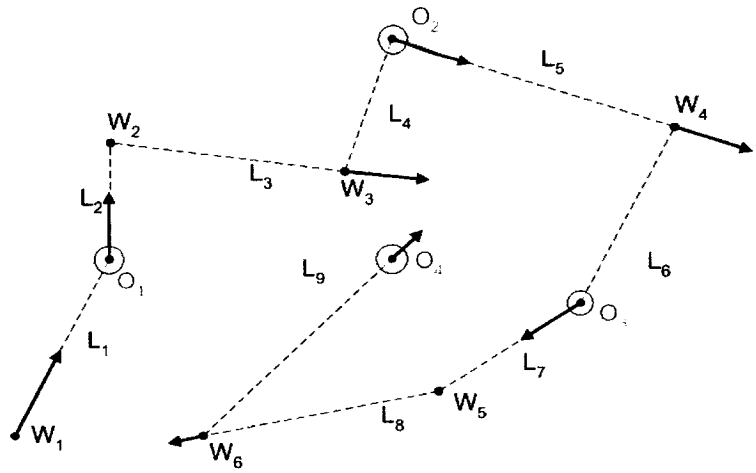
Chapter 5: Trajectory Generator

The purpose of the trajectory generator (TG) is to convert a path, which consists of a simple sequence of waypoints, into a trajectory including the position, velocity and time profiles, according to the dynamic characteristics of each aircraft. The approach pursued in this work proceeds in three steps. Firstly, the skeleton of the trajectory is constructed by linking all waypoints with straight segments, respecting the order in which they are encountered in the path. Secondly, vectors of approach (VoA) are assigned to objective waypoints and their preceding transitional waypoint if any. Thirdly, the straight segments are replaced by 3D curvilinear trajectories satisfying the aircraft dynamics. Additional waypoints are inserted in that process in order to have a complete parametric representation of the trajectory. The third steps compute the load factor, speed, and geometry of the trajectory between each pair of waypoints. It also assesses the dynamic feasibility of these segments. Figure 5.1 schematically summarizes this process.

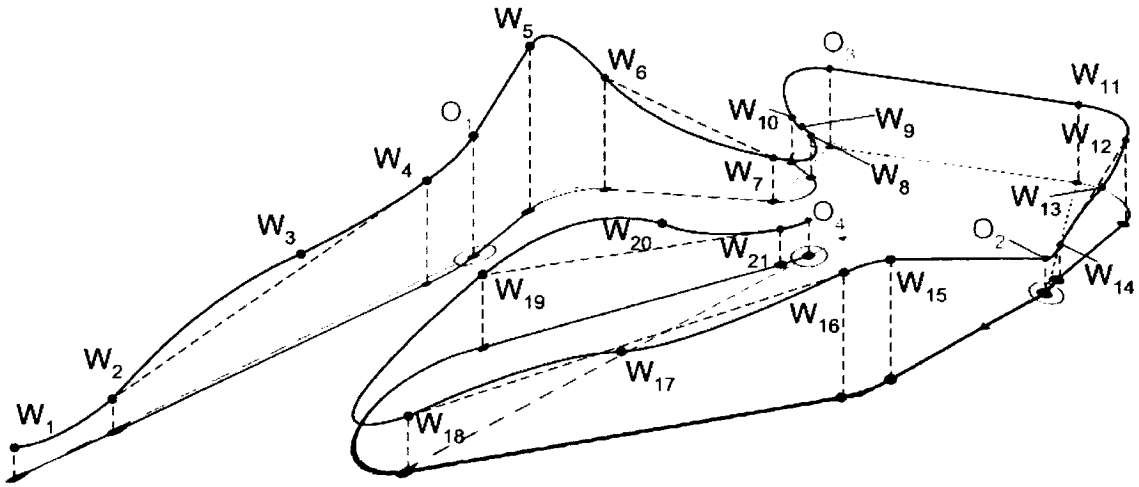
Rather than using complex dynamic models which would result in a lengthy optimization process, compact algebraic solutions were developed to face every feasible configuration of waypoints. This approach attempts to minimize the amount of data required as well as the processing time, in order to boost the performances of the GA to near-real time.



(a) Step1 : A path composed of transitional waypoints (Wi) and objective waypoints (Oi)



(b) Step 2: Augmented path integrating vectors of approach



(c) Step 3: Fully defined 3D parametric representation of a trajectory

Figure 5.1: *The TG: transforming a path into a 3D parametric trajectory*

These algebraic solutions are based on the well defined *performance equations* of subsonic aircrafts, as well as on purely geometric constructs such as circles, helix, lines, etc. They guarantee an optimal trajectory in terms of speed and distance traveled. However, optimization of more complex factors such as aircraft loading conditions and fuel consumption would actually require the use of more complete and computationally intensive dynamic models. Because the gains of such approach are largely outweighed by the decrease in GA processing-time, the optimization of such factors is left for future development of this system. This chapter is divided into two parts. The first one introduces the basic tools required to construct a trajectory, while the second one presents the detailed algorithm of the Trajectory Generator (TG).

5.1. Building blocks of a trajectory

In order to clearly understand the different levels at which the TG algorithm operates, the components constituting a trajectory must be presented first. Section 5.1.1 presents the mechanism that interfaces the output of the GA with the input of the TG. Section 5.1.2. presents the basic blocks used to construct each segment of the trajectory. Section 5.1.3. describes how these blocks are assembled together to create segments with continuous curvature at their junctions.

5.1.1. Augmented Path: the skeleton of a trajectory

The output of the GA is an ordered sequence of transitional and objective waypoints. Each pair of waypoints specified in the path provided by the GA is called a *segment*. The first step of the TG is to convert the path given by the GA into a pattern that is used as a first approximation of the trajectory's geometry. This is achieved by linking all contiguous pairs of waypoints by lines as shown in Figure 5.1 (a).

In the general case, constraints on objective's VoA are not specified. Thus, the second step consists of assigning VoA to all objective waypoints and all transitional waypoints preceding one. This process ensures that all objective waypoints are traversed by the vehicle without making excessive turns or other manoeuvres. If specific vectors of approach have already been specified for some objectives based on type (5) constraints, these vectors are kept unmodified. For all others, the following rules are applied:

- All transitional waypoints preceding an objective waypoint are assigned a VoA in the direction of the straight segment *preceding* them (e.g. W_3, W_4 and W_6 in Figure 5.1 (b)).
- All objective waypoints are assigned a VoA set in the direction of the straight segment *following* them (e.g. O_1 to O_3 in Figure 5.1 (b)).
- The starting waypoint is assigned a VoA set in the direction of the initial direction of the vehicle's velocity. If this is unknown, the VoA is assumed collinear to the first segment (e.g. W_1 in Figure 5.1 (b)).
- The objective at the end of a path is assigned a VoA set in the direction of the straight segment *preceding* it (e.g. O_4 in Figure 5.1 (b)).

Figure 5.1 part (b) shows the result of this process in a situation where no constraints have been specified and where the direction of the initial velocity vector is unknown (as shown in part (a)). The result is referred as the *augmented path* and provides the initial framework required to compute the geometry of the final trajectory.

5.1.2. Segment: a block of sections

The augmented path gives to the TG a series of segments, each consisting of a straight line linking two waypoints possibly associated with VoA's. Because sharp turns are dynamically unfeasible for aircrafts, the sequence of straight lines must be replaced by a series of smooth curves satisfying all specified VoA's. The approach is to construct the geometry of the trajectory iteratively, defining it segment by segment. The procedure used to construct the augmented path ensures continuity between each segment (the problem of segments continuity is addressed in the section 5.1.3.).

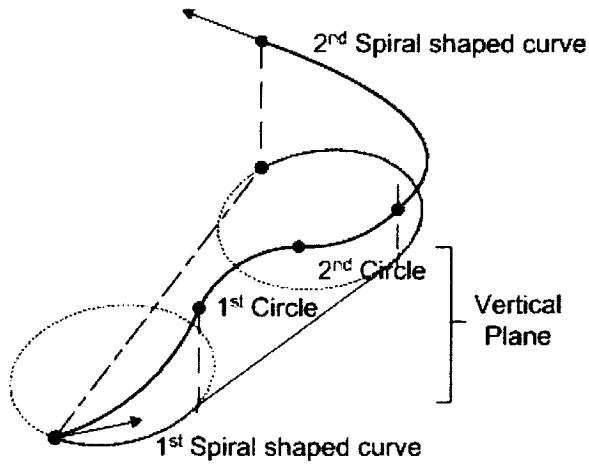


Figure 5.2: *Set of curves for shortest way between two points in 3D*

At the segment level, the problem comes down to finding the minimum set of curves that link one waypoint to another, while ensuring that the end point of these curves are tangent to the waypoints' VoA. As reported in *Anderson et al. 2005*, the shortest way linking two points with curvature constraints consists of two circles and a straight line in 2D space. For this work, the path in 3D is build of two spiral shaped curves, and two circles in the vertical plane as illustrated in Figure 5.2.

Each curve composing part of a segment is referred in this work as a *section.* A segment contained in 2D will consists of three sections. This situation can represent an

aircraft keeping a levelled flight in the horizontal plane. A segment in 3D will consists of four sections in the general case. Assuming that one of the VoA is collinear to the vertical plane shown in Figure 5.2, the segment will be reduced to three sections, only one spiral shaped curve being necessary. Using standard algebraic equations, a trajectory can be established between any two points using only five types of section. The geometry and underlying aircraft dynamics of these sections are now presented. The standard Cartesian axes are used, where the X-Y plane representing the horizontal plane, and the Z axes relating to the altitude.

5.1.2.1. Horizontal turn section

This section describes a turn in the X-Y plane based on a circular trajectory with a constant speed and bank angle. The general configuration is shown in Figure 5.3. Any point along this section can be found by the following set of parametric equations:

$$\begin{aligned} S_X &= R_H \cos(v^*\Delta\gamma_r + \gamma_{r,0}) + C_X \\ S_Y &= R_H \sin(v^*\Delta\gamma_r + \gamma_{r,0}) + C_Y \\ S_Z &= C_Z \end{aligned} \qquad 5.1$$

with R_H the horizontal radius of the turn

$C(X,Y,Z)$, the center of the turn in global coordinates

$\Delta\gamma_r$, the angle between the initial and final velocity vectors of the section

$\gamma_{r,0}$, the initial angle at which the aircraft initiates the turn in terms of the global Cartesian coordinates of the map.

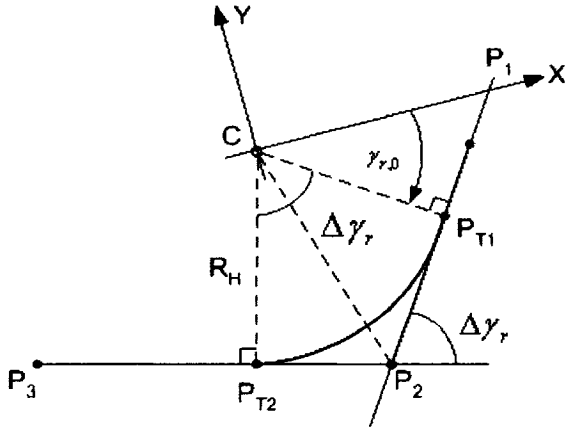


Figure 5.3: *General configuration for a turn inscribed in the horizontal plane*

The variable v allows to describe the whole set of points defining the curve S in three dimensions based on a single parameter. This variable may vary from 0 to 1, the value 0 giving the first point of the curve and 1 giving the last one.

The constants R_H, $\Delta\gamma$, $\gamma_{r,0}$ and (C_X, C_Y, C_Z) can be found based on geometric relations as well as standard aircraft dynamic equations. The equations giving these constants are included in Appendix-D.

5.1.2.2. Vertical turn section

This section describes a turn in the vertical plane (i.e. a partial loop) based on a circular trajectory with a near-constant speed and a zero bank angle. Figure 5.4 shows the vertical turn defined within a plane Z-Γ, where Γ is the horizontal axis and the Z-axis is the vertical one defining the altitude of the aircraft.

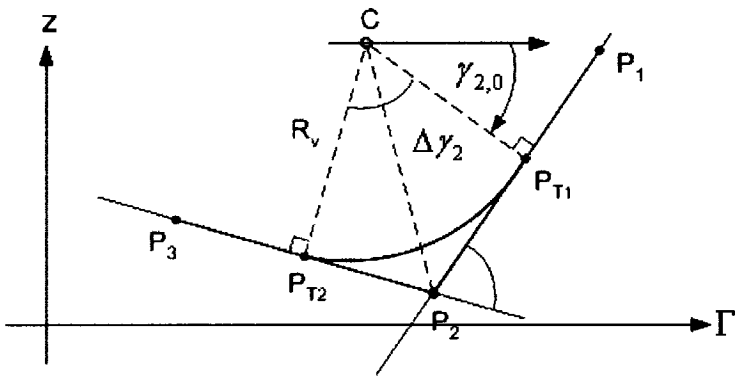


Figure 5.4: *Configuration of a turn in a vertical plane*

The equations describing any point $S(\Gamma, Z)$ on this type of section can be computed by the following set of parametric equations:

$$\begin{aligned} S_\Gamma &= R_V \cos(v \cdot \Delta\gamma_2 + \gamma_{2,0}) + C_\Gamma \\ S_Z &= R_V \sin(v \cdot \Delta\gamma_2 + \gamma_{2,0}) + C_Z \end{aligned} \qquad 5.2$$

with R_V, the vertical radius of the turn

$C(\Gamma, Z)$, the center of the turn

$\Delta\gamma_2$, the angle between the initial and final velocity vectors of the section

$\gamma_{2,0}$, the initial angle at which the aircraft initiates the loop with respect to the horizontal axis Γ

The constants R_V, $\Delta\gamma_2$, $\gamma_{2,0}$ and $C(\Gamma, Z)$ can be found based on geometric relations as well as standard aircraft dynamic equations presented in Appendix-D.

A full thee-dimensional representation of this section is shown in Figure 5.5 where the axis Z- Γ are the local reference frame of the section while *X,Y,Z,* define the global reference frame.

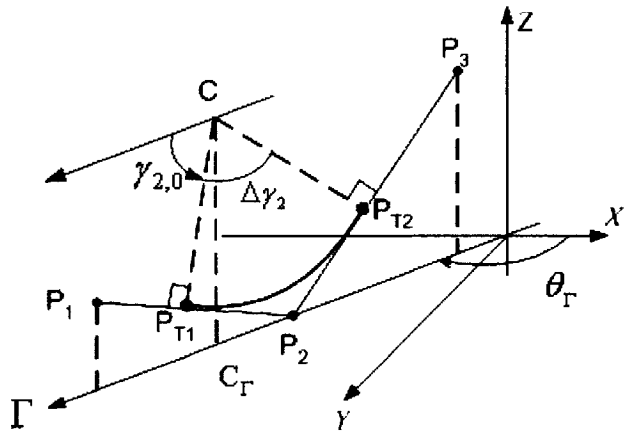


Figure 5.5: *Vertical turn configuration in 3-D*

Any points in the Z- Γ plane can be given with respect to the global reference frame *X,Y,Z* according to the following equations:

$$\begin{aligned} P_X &= P_\Gamma \cos\theta_\Gamma \\ P_X &= P_\Gamma \sin\theta_\Gamma \\ P_Z &= P_Z \end{aligned} \qquad 5.3$$

The angle θ_Γ is given by the following equation:

$$\theta_\Gamma = \tan^{-1}\left(\frac{P_{3Y} - P_{1Y}}{P_{3X} - P_{1X}}\right) \qquad 5.4$$

5.1.2.3. Log section

This section combines a turn in the vertical and horizontal planes. It is similar to a helix with a varying pitch. The horizontal component is based on a fixed radius of curvature R_H, which result in the same x-y parametric equations as for the horizontal turn section. The vertical component is based on a vertical loop with a constant rate of change for the pitch of the aircraft. As given in equation 5. 5, the load factors associated with the segment's horizontal and vertical components are chosen such that the norm of these two factors is less than the maximum load factor of the aircraft. It is assumed that decomposing the load factor as such ensures that the total load factor of the maneuver falls within the aircraft's maneuver envelope (i.e. within the allowable load range).

$$n_{max} \geq (n_H^2 + n_V^2)^{1/2} \qquad 5.5$$

with n_V, the load factor associated with the vertical component of the turn

n_H, the load factor associated with the horizontal component of the turn

Any point along this section can be found by the following set of parametric equations:

$$\begin{aligned} S_X &= R_H \cos(v \cdot \Delta\gamma_r + \gamma_{r,0}) + C_X \\ S_Y &= R_H \sin(v \cdot \Delta\gamma_r + \gamma_{r,0}) + C_Y \\ S_Z &= \frac{R_H \Delta\gamma_r}{\Delta\gamma_2} \ln\left(\frac{\cos(\gamma_{2,0})}{\cos(v\Delta\gamma_2 + \gamma_{2,0})}\right) + Z_0 \end{aligned} \qquad 5.6$$

As described in greater details in Appendix-D, the equation giving the S_Z is based on two assumptions; the rate of change of the pitch is assumed constant, and the pitch angle is assumed to be bounded between $[-\pi/2; \pi/2]$.

The variables R_H, $\Delta\gamma_r$, $\gamma_{r,0}$ and C are calculated based on the equations for a standard horizontal turn with a load factor set to n_H. The variable $\gamma_{2,0}$ is set to the pitch angle of the aircraft at the beginning of the log section. If there is a section preceding it, the pitch angle is set equal to the pitch angle of the last point of that section. If the *log* segment is the first one of the path, the pitch is calculated based on the initial velocity vector of the aircraft. Z_0 is found in the same way, using either the last point of the preceding section, or the initial position of the aircraft.

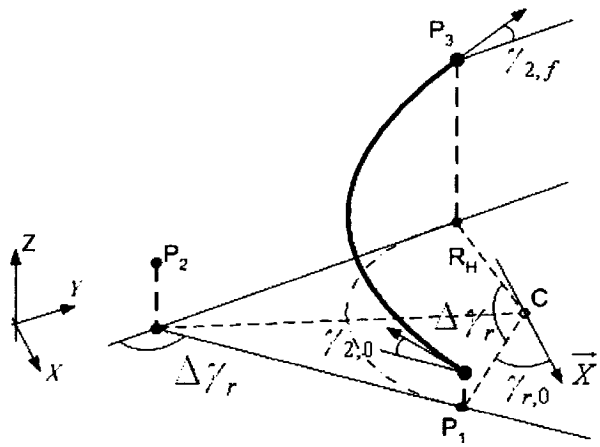


Figure 5.6: *Typical configuration of a Log section*

The final pitch angle depends on the variables R_H and $\Delta\gamma_r$, which define the horizontal component of the turn. Thus, the total variation in pitch possible along a log section is directly dependent on the total angular variation of the aircraft in the horizontal plane.

5.1.2.4. Helix section

This section illustrated in Figure 5.7, is a special case of a log section where the pitch angle remains constant. As for the log section, the bank angle remains constant, resulting in a constant radius of turn R_H. The name of that section comes from the mathematical construct describing this configuration. The equations for the section's horizontal component are the same as for the horizontal turn. However, the Z-component is simplified to the expression shown below:

$$\begin{aligned} S_X &= R_H \cos(v \cdot v_c \, \Delta\gamma_r + \gamma_{r,0}) + C_X \\ S_Y &= R_H \sin(v \cdot v_c \, \Delta\gamma_r + \gamma_{r,0}) + C_Y \\ S_Z &= v \cdot (\tan(\gamma_2) \, R_H \, \Delta\gamma_r) + Z_0 \end{aligned} \qquad 5.7$$

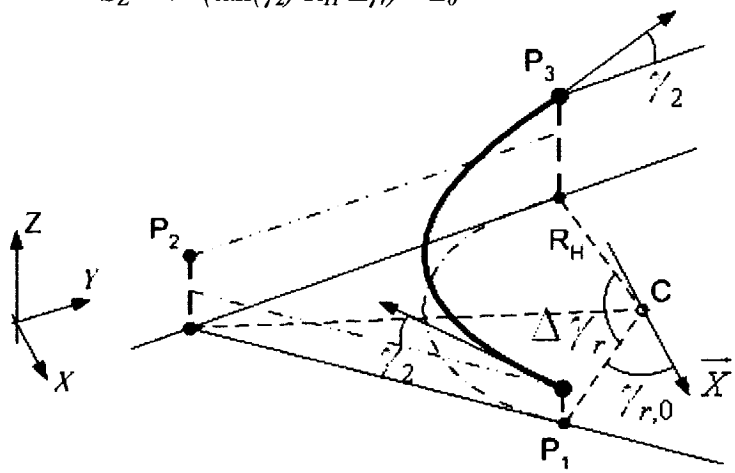


Figure 5.7: *Typical configuration of a Helix section*

5.1.2.5. Line section

This section can be used for three maneuvers, a level flight, a steady climb and a steady descent. The limits imposed on velocity and pitch angles are covered in Appendix-B which presents a brief overview of subsonic aircraft dynamics. The parametric equations of a line section are given as:

$$\begin{aligned} S_X &= v \, (P_{2,X} - P_{1,X}) + P_{1,X} \\ S_Y &= v \, (P_{2,Y} - P_{1,Y}) + P_{1,Y} \\ S_Z &= v \, (P_{2,Z} - P_{1,Z}) + P_{1,Z} \end{aligned} \qquad 5.8$$

with P_1 and P_2 the first and last waypoints respectively defining the section in global coordinates.

Since a change in altitude will result in changes in atmospheric conditions, the speed of the aircraft may have to change as well for long straight sections. Thus, long line

sections may have to be fractioned into a series of smaller one with varying average velocities. This factor is of limited concern for the previous three sections since the first one involves no altitude changes, and the other two can be completed in a relatively narrow altitude range. However, the major portion of a UAV's trajectory will be composed of line sections with potentially large altitude variations. Since each section may have a different velocity, the resulting overall velocity distribution along a trajectory might be discontinuous. Even if the autopilot attempts to follow that distribution, the inertia of the aircraft would unavoidably smooth out any velocity changes. Thus, the use of more complex model defining non-constant velocity profiles along a section may not be necessary. The five sections described above do not necessarily have to be precisely followed by the autopilot. Although following the position profile of the trajectory is essential to avoid potential collisions, the velocity profiles can differ with a relatively large margin of error without affecting the safety of the aircraft, as long as the velocity is below the limits defined by its dynamics (which is the prime condition verified by the controller for each section computed). The main purpose of these sections is to check if a trajectory is dynamically feasible without damaging the aircraft by excessive accelerations, turning angles, etc.

5.1.3. Continuous transition between trajectory segments

This section describes how the basic blocks outlined in the previous section are assembled together to create segments with continuous curvature at their junctions. The approach is to construct the geometry of the trajectory iteratively. Starting with the first pair of waypoints, the geometry of each segment is computed sequentially up to the end of the augmented path. In order to ensure continuity between segments, this iterative process always uses the VoA of the last point of a segment as the initial VoA of the next one. The rules used to construct the augmented path assign VoA's in a way that supports that process as discussed in section 5.1.1. Since these rules do not assign vectors of approach for segments containing only transitional waypoints, two cases can be encountered when processing a segment:

a) In the first case, the segment's initial and final waypoints have VoA's. This configuration is the simplest one when considering continuity. The VoA's are fully

defined and the next segment will use the last one for its starting point. This configuration is referred as a *pair of VoA*.

b) In the second case, the segment has only one VoA assigned to its starting waypoint, which is either the VoA of the end of the previous segment or the VoA of the first waypoint of the augmented path. This case can be further divided into two situations:

   i. In the first situation, the initial VoA is collinear with the line linking the segment's pair of waypoints. The missing VoA is then set in the direction of the next straight segment. This specific configuration is referred as a *corner point configuration.*

   ii. In the second situation, the initial VoA points in a direction that differs from the straight segment of the augmented path. As for case (i), the missing VoA is set in the direction of the next straight segment. However, the configuration becomes equivalent to (a) once the direction of the VoA has been set for the final waypoint of the segment. This situation can thus be treated as a "pair of VoA".

Since case (a) and case b(ii) are essentially the same, only two cases are presented in this section: the *corner point configuration* and the *pair of VoA* configuration. These two configurations are the two basic mechanisms that link segments together while ensuring continuity between them. The following sections provide the equations required to fully define their geometry.

5.1.3.1. Corner point configuration

A corner point consists of a configuration where a segment composed of an initial waypoint P_1 and a final waypoint P_2, has only one VoA assigned to P_1 pointing in the direction of the vector P_1P_2. The missing VoA of P_2 is then set equal to the vector P_2P_3. Since all objective waypoints have VoA, the point P_2 is necessarily a transitional waypoint which do not require to be traversed by the trajectory of the UAV. Thus, the sharp turn at P_2 can be replaced by a smooth turn from the P_1P_2 segment to the P_2P_3 segment.

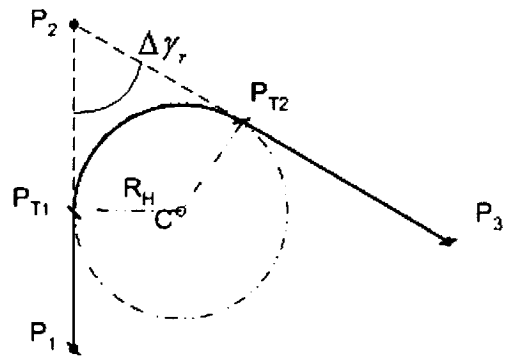


Figure 5.8: *Top view of a corner point configuration*

The horizontal component of this turn corresponds to a circle tangent to both segments. P_{T1} is the point of contact of the first tangent with the circle C, and P_{T2} corresponds to the second tangent. A top view of a corner point configuration is depicted schematically in Figure 5.8.

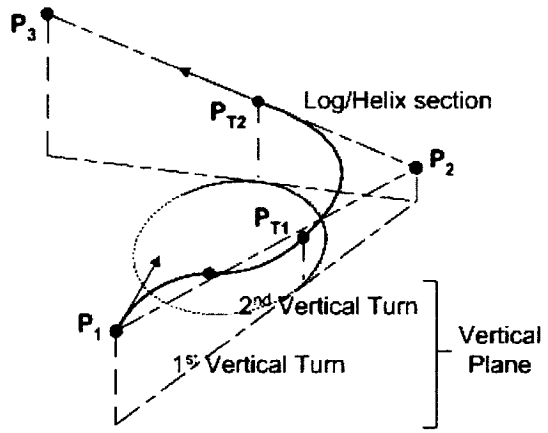


Figure 5.9: *Corner point configuration in 3D*

In 3D, this configuration is composed of two vertical turn sections between P_1 and P_{T1}, and one log or helix section from P_{T1} to P_{T2}. This configuration is shown in Figure 5.9. In other words, the segment P_1P_2 is essentially replaced by a segment P_1P_{T2} with a VoA at P_{T2} equal to the vector P_2P_3. The section $P_{T2}P_3$ is treated as a separate segment that is to be processed after P_1P_{T2}. As discussed in section 5.1.2.3. , the maximum variation in pitch during a turn is directly proportional to the total angular variation of the aircraft in heading (i.e. in the horizontal plane). Thus, fixing the final pitch angle equal to the VoA at P_{T2} may result in situations where the achievable pitch angle cannot meet the required value at P_{T1} (i.e. the beginning of the turn in the horizontal plane). In other words, situation may occur when P_1 and P_{T1} cannot be aligned along a single linear

section. Thus, two vertical turns are required between P_1 and P_{T1} to adjust the pitch angle to a feasible value at the beginning of the log (or helix) section P_{T1} P_{T2}.

The horizontal components of this configuration are the same as for a horizontal turn described in section 5.1.2.1. Thus, the equations used to find C, $\Delta\gamma_r$, and the two tangent points are also the same. Having found P_{T1} and P_{T2}, the line section that connects the pairs (P_1, P_2) can be replaced by two vertical turn sections connecting P_1 to P_{T1}, such that the curvature of the second vertical turn and the log/helix section is continuous. However, the variable R_H cannot be found solely on dynamic considerations as it was the case for the horizontal turn section. An additional constraint must be considered on the maximum radius R_H allowable by the geometry of the configuration. Defining this maximum radius as the critical radius R_C, the following relation must hold if the segment is to be dynamically feasible:

$$R_C \geq \min(R_H) \qquad 5.9$$

The radius of a horizontal turn is proportional to the square of the velocity, and inversely proportional to the load factor n. If condition 5. 9 is not met, the aircraft would either attempt to slow its speed below its stall speed to further decrease R_H, or increase the load factor for that maneuver to a value above its physical limit. In both cases, the aircraft may be damaged. The critical radius R_C can be found by the following method:

1. *Find the angle $\Delta\gamma_{r,c}$*

   Compute the complementary angle $\Delta\gamma'_{r,c}$ (see Figure 5.10) from $\overrightarrow{P_2P_1}$ to $\overrightarrow{P_2P_3}$.

2. *Find the norm of the shortest segment*
   i. Find the shortest line segment L_S between P_1P_2 and P_2P_3.
   ii. Select the end point P_S of L_S opposite to the common corner point P_2 of the two line segments. The point P_S represents the critical tangent point of the turn.
   iii. Calculate the norm of the vector P_SP_2.

3. *Find the critical radius*
   i. Using the geometrical relationships described for the horizontal turn segment, the relation linking the radius R to the tangent point and $\Delta\gamma_r$ is given as:

$$\left\|\overrightarrow{P_T P_2}\right\| = R\tan\left(\frac{\Delta\gamma_r}{2}\right) \qquad 5.10$$

ii. Using the critical tangent point and corresponding angle $\Delta\gamma_{r,c}$, the critical radius is:

$$R_C = \frac{\left\|\overrightarrow{P_S P_2}\right\|}{\tan\left(\Delta\gamma_{rC}/2\right)} \qquad 5.11$$

And the tangent point P_T can be found by solving equation 5. 10.

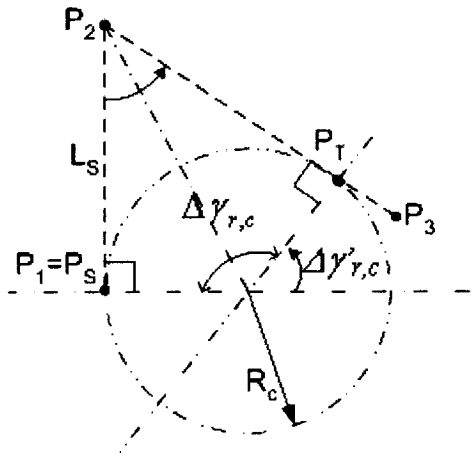


Figure 5.10: *Finding the critical radius of a horizontal turn*

Figure 5.10 shows the geometrical configuration for the critical radius of a turn. Since the optimization process implemented to compute the vertical component of the sections of this configuration is the same as for the "pair of VoA" configuration, its description is given in section 5.2.

5.1.3.2. Configuration for a pair of VoA's

The second basic type of configuration that ensures continuity of the curvature between segments corresponds to a segment that has both its initial and final waypoints associated with VoA. As presented in section 5.1.2. and shown in Figure 5.2, the minimal trajectory that can connect any pair of waypoints with VoA's consists of two spiral shaped curves and two vertical turns contained in the same vertical plane. Since the vertical component of this configuration is directly dependent on the horizontal one, the latter is solved first, reducing the first part of the problem to a 2D configuration. As for the corner point configuration, the detailed algorithm used to find and optimize the

vertical component of the trajectory is given in the description of the overall TG algorithm in section 5.2. Figure 5.11 shows the horizontal component of this configuration.

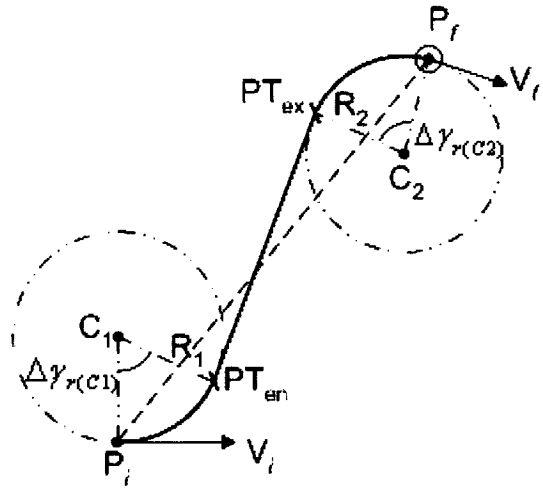


Figure 5.11: *Configuration for a pair of VoA's*

Considering only the horizontal component, the pair of VoA's configuration can be decomposed into two horizontal turn sections and one line section connecting PT_{en} and PT_{ex}. The names PT_{en} and PT_{ex} stand for "entry tangent point" and "exit tangent point" referring to the aircraft entering the line segment after turn *C1* and exiting it before turn *C2*. Thus, the radius of curvature of the turns *R1* and *R2* can be found based on the same dynamic considerations presented in section 5.1.2.1. The rest of this section describes the algorithm that has been developed to find the remaining variables: *C1, C2,* PT_{en}, and PT_{ex}, given the input parameters P_i, P_f, V_i and V_f.

1. Characterization of the 2D configuration

Taking P_f and V_f as the local reference of the whole configuration, the geometry of this type of segment can be entirely defined based on the parameters shown in Figure 5.12. These parameters are defined as follows:

- The *configuration section (CS)* angle θ_{CS}, which defines the position of P_i with respect to a P_f using V_f as the axis of reference.
- The *distance D* between P_i and P_f.
- The *tangent* T_{if} to circle *C2* (which can either be $C2_S$ or $C2_P$ depending on θ_{CS}) passing through point P_i.

T_{if} is set to respect the direction of the turn at *C2*, i.e. CCW if *C2* is set to be on port and CW if set on starboard.

- The angle θ_{Tif}, which defines the orientation of V_i with respect to T_{if}.
- The radius of curvature of *R1* and *R2* associated to circles *C1* and *C2*.

All angles are always defined CCW from one reference vector to a second vector. The nomenclature used in Figure 5.12 describes the position of *C1* (or *C2*) as either *starboard* (S) or *port* (P), depending if the circle is respectively to the right or the left of the aircraft's flight directions V_i and V_f. The circles defining the vehicle's turns are simply referred in this thesis as *C1* or *C2* when their relative position is not relevant to the part of the problem being described. The centers *C1* and *C2* are positioned on a line perpendicular to their respective vector of approach V_i and V_f, at a distance *R1* and *R2* from their respective end points P_i and P_f. Assuming that the radii *R1* and *R2* are given, the centers of the horizontal turns are readily known from the two waypoints and their VoA's.

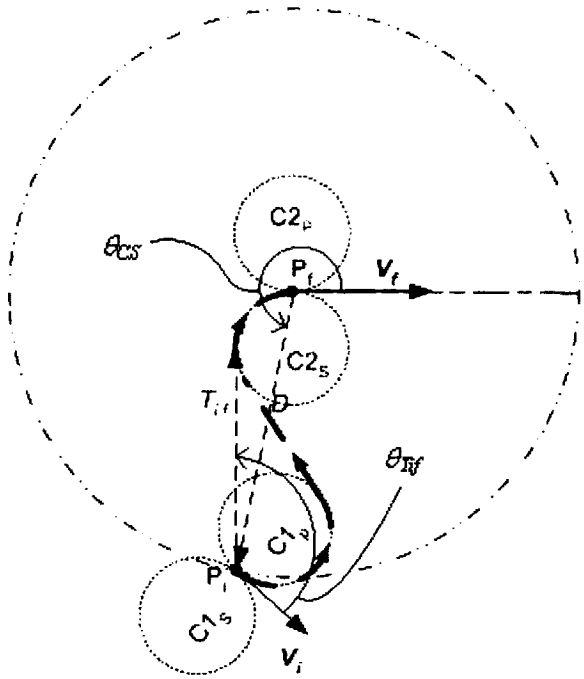


Figure 5.12: *General Configuration for a pair of VoAs*

2. Finding the position of C1 and C2 based on θ_{CS}

As shown in Figure 5.11, a valid solution for this kind of configuration consists of finding the tangent $PT_{en}\,PT_{ex}$ between circles *C1* and *C2*. Since there are four tangents

between two circles, and these two circles can each have two possible positions (i.e. on the starboard or port side of the vectors V_i and V_f), the mathematical solution is not unique. Figure 5.13 illustrates the four possible tangents between two circles, given that the positions of these two circles are known.

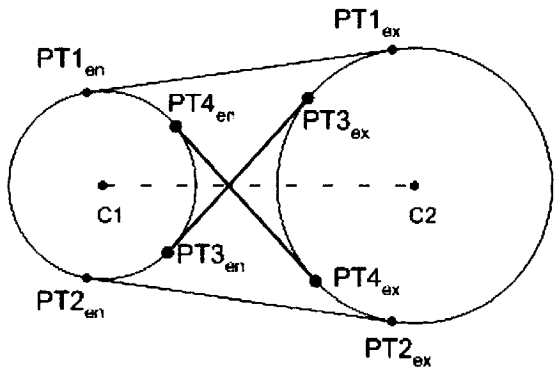


Figure 5.13: *Four possible tangent linking two circles*

The constraints regarding the turning direction around *C1* and *C2* add to the complexity of the problem without necessarily reducing the number of possible solutions. Thus, the problem can be defined as selecting the right *C1* and *C2*, choosing between the port and starboard sides for each of them, finding the tangents between these circles, and selecting the tangent that gives the shortest trajectory while respecting the direction in which the vehicle will turn around *C1* and *C2*.

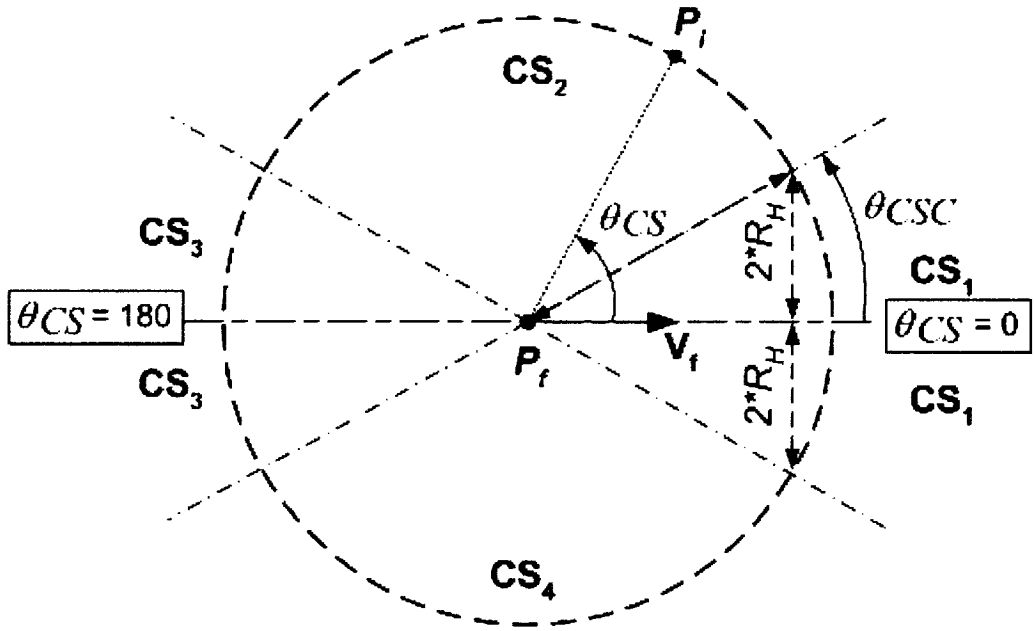


Figure 5.14: *Configuration sections*

According to the parameters descriptions given in this section, it is possible to find general solutions by dividing the range of possible waypoint arrangements into four groups with two special cases. Those groups are referred to as *configuration sections* (*CS*) in Figure 5.14. A straight forward mathematical solution has been derived for each group. The configuration section, and thus, the procedure that must be selected to solve this problem, depends only on the *configuration section angle,* θ_{CS}. As shown in Figure 5.14, the limits of the intervals defining each group can be determined based on the critical angle θ_{CSC}. This angle can be found by constructing the right-angle triangle with a side length of $(2 \cdot R1)$ and a hypotenuse of length D equal to the distance between P_i and P_f Based on this construct, the angle θ_{CSC} can be found using equation 5. 12.

$$\theta_{CSC} = \sin^{-1}\left(\frac{2R1}{\|\overrightarrow{P_i P_f}\|}\right) \qquad 5.12$$

3. The geometric solutions of the three CS

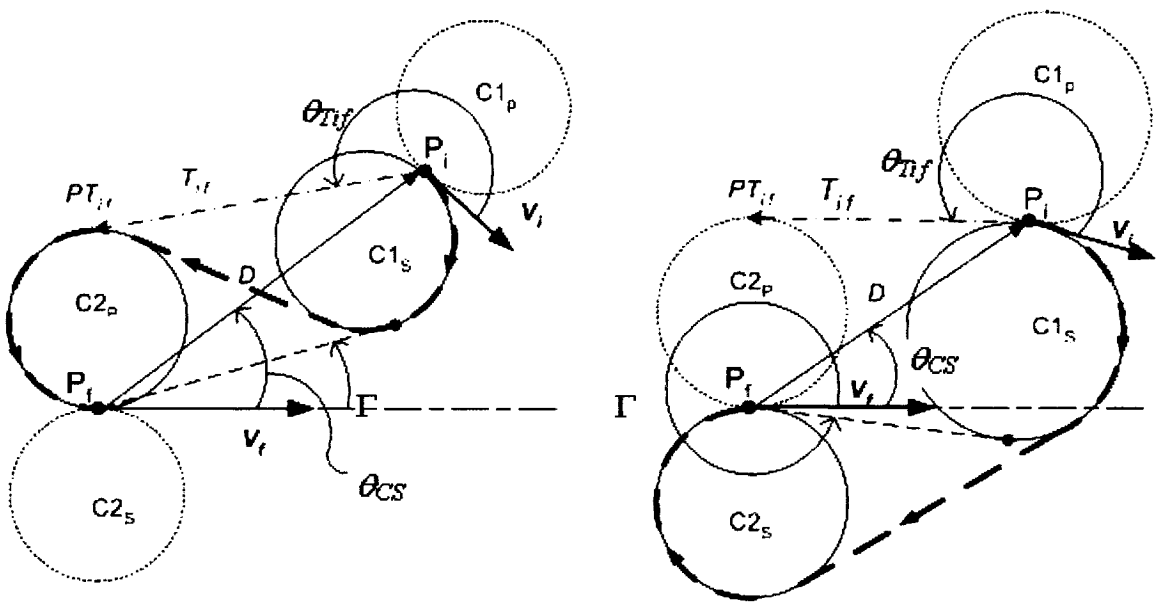


Figure 5.15: *Configuration section #1, finding the position of C1 and C2 (left) for* $\Gamma \leq \pi$ *and (right) for* $\Gamma > \pi$

The solution for the four configuration sections are as follows:

- For *CS = 1 or CS = 3:*

  If $\theta_{CS} \leq \pi$ then $C2_P$ is used to compute tangent T_{if}. Otherwise, $C2_S$ is used.

If $\theta_{Tif} \leq \pi$ then $C1$ is on port. Otherwise $C1$ is on starboard.

If the angle Γ between the vector V_f and the tangent of $C1$ crossing P_f, is less or equal than π, then $C2$ is on port. If it is greater than π, $C2$ is on starboard.

The tangents are calculated such that the turning direction around $C1$ and $C2$ are respected. This constraint is satisfied when the following equalities are satisfied:

$$\begin{aligned} \overrightarrow{T_{if}} \hat{\bullet} \overrightarrow{PT_{if}C2} &= \overrightarrow{V_f} \hat{\bullet} \overrightarrow{P_fC2} \\ \overrightarrow{PT_1P_f} \hat{\bullet} \overrightarrow{PT_1C1} &= \overrightarrow{V_i} \hat{\bullet} \overrightarrow{P_iC1} \end{aligned} \qquad 5.13$$

with the operator [$\hat{\bullet}$] calculating the angle θ from a reference vector V_r to another vector V_f measured counter clockwise. The convention for this operator is to place the reference vector as the first argument, i.e. $\theta_{r \to f} = V_r \hat{\bullet} V_f$

- For $CS = 2$:

  The tangent T_{if} is computed based on circle $C2_P$.

  If $\theta_{Tif} \leq \pi$ then $C1$ is on port. Otherwise $C1$ is on starboard.

  C2 is set on port.

- For $CS = 4$:

  The tangent T_{if} is computed based on circle $C2_S$.

  If $\theta_{Tif} \leq \pi$ then $C1$ is on port. Otherwise $C1$ is on starboard.

  C2 is set on starboard.

The two special cases are as follows:

- $\theta_{CS} = 0$ *or* π:

  The tangent T_{if} is equal to the vector $\overrightarrow{P_iP_f}$.

  If $\theta_{Tif} \leq \pi$ then $C1$ and $C2$ are on port. Otherwise both are on starboard.

- $\theta_{CS} = \pi$ and $\theta_{Tif} = 0$:

  The vectors V_i and V_f are collinear. No turn is required.

4. Finding the position of all tangent points

Once the centers $C1$ and $C2$ have been correctly positioned with respect to their associated VoA, the tangents of the two circles have to be found. The geometry of the problem is shown in Figure 5.16.

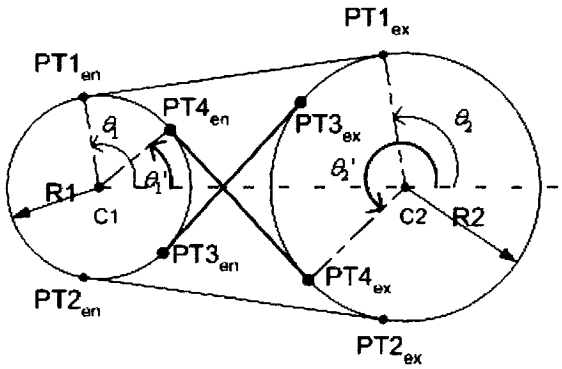


Figure 5.16: *Tangent line segments*

The geometric relations describing this configuration are and the equations giving the coordinates of $PT1(PT1_{en}, PT1_{ex})$, $PT2(PT2_{en}, PT2_{ex})$, $PT3(PT3_{en}, PT3_{ex})$ and $PT4$ $(PT4_{en}, PT4_{ex})$ are given in Appendix D.4.

5. Selecting the appropriate set of tangent points

The appropriate set of tangent points PT (PT_{en}, PT_{ex}) can be found *a priori* based on the configuration's parameters determined in the previous steps:

- $CS = 1$ and $CS=3$:

  For $\vec{V_i} \hat{\bullet} \overrightarrow{P_iC1} = \vec{V_f} \hat{\bullet} \overrightarrow{P_fC2}$, the tangent points PT should be set to $PT1$ if $\theta_{Tif} \leq \pi$ and $PT2$ otherwise.

  For $\vec{V_i} \hat{\bullet} \overrightarrow{P_iC1} \neq \vec{V_f} \hat{\bullet} \overrightarrow{P_fC2}$, the tangent points PT should be set to $PT3$ if $\theta_{Tif} \leq \pi$ and $PT4$ otherwise.

- $CS = 2$:

  The tangent points PT should be set to $PT1$ if $\theta_{Tif} \leq \pi$ and $PT4$ otherwise

- $CS = 4$:

  The tangent point PT should be set to $PT3$ if $\theta_{Tif} \leq \pi$ and $PT2$ otherwise

6. Limiting parameters of the pair of VoA configuration

The set of tangent points PT_{en}, PT_{ex} can be used to compute the angle $\Delta\gamma_{r1}$ and $\Delta\gamma_{r2}$ of the two horizontal turns $C1$ and $C2$ respectively, as well as the equation of the line segment in between as shown in Figure 5.11. Thus, the parameters PT_{en}, PT_{ex}, $C1$ and $C2$ fully define the position profile of the horizontal component of the trajectory's segment.

Since this configuration is dependent on the radius *R1* and *R2* which vary with the speed and load factor of the aircraft, two geometric constraints must be defined before any of these dynamic characteristics are determined: the angles $\Delta\gamma_{r(Ci)}$ and the critical radius R_C. Assuming that the aircraft makes two turns *C1* and *C2* with the same speed and load factor, these parameters can be computed as follows:

a) Find the critical radius

The critical radius is reached when the length of the tangent segment between the two circles is zero. Assuming *R1* and *R2* are equal (i.e. constant velocity through the entire segment), R_C is defined as:

$$R_C = \frac{\left\|\overrightarrow{C1C2}\right\|}{2} \qquad 5.14$$

b) Computing the angle $\Delta\gamma_{r(Ci)}$

The critical angles $\Delta\gamma_{r(C1)}$ and $\Delta\gamma_{r(C2)}$ are used to evaluate the maximum variation in pitch allowable for a log or helix section (see sections 5.1.2.3. and 5.1.2.4.) as described in more details in section 5.3.5.4. They are computed based on the turning direction of *C1* & *C2*:

$\Delta\gamma_{r(C1)} = \overrightarrow{C1P_i} \hat{\bullet} \overrightarrow{C1PT_{en}}$. If turn *C1* is CCW, then $\Delta\gamma_{r,c(1)} \in [0; 2\pi]$, otherwise its complement is used such that $\Delta\gamma_{r(C1)} \in [-2\pi; 0]$.

$\Delta\gamma_{r(C2)} = \overrightarrow{C2\,PT_{ex}} \hat{\bullet} \overrightarrow{C2\,P_f}$. If turn *C2* is CCW, then $\Delta\gamma_{r,c(2)} \in [0; 2\pi]$, otherwise its complement is used such that $\Delta\gamma_{r(C2)} \in [-2\pi; 0]$.

The value of R_C, $\Delta\gamma_{r(C1)}$ and $\Delta\gamma_{r(C2)}$ define the maximum values for which the series of sections constituting the *pair of VoA* configuration remains continuous. Increasing further the radius or the angle results in a disjoint configuration which destroys the continuity of the entire segment, making it dynamically unfeasible.

5.2. Overall algorithm of the trajectory generator

The task of the TG is to convert the augmented path into a trajectory with its position and velocity profiles. The TG must compute dynamically feasible sections of the trajectory, ensuring smooth transitions between them. Section 5.1.2. showed the geometric constructs used to defined each section of a segment. Section 5.1.3. described the algorithm that ensured continuity between consequent segments. This section shows the overall algorithm of the TG and the iterative process that optimizes the trajectory with respect to the velocity and load factor, based on the UAV dynamics

The TG operates on three levels; a higher level, an intermediate level, and a lower level. The higher level builds the augmented path; parses the augmented path segment by segment; identifies the type of configuration for each one (i.e. either a pair of VoA or a corner point configuration); and call the proper mid-level function to solve it. The intermediate level functions take a pair of two or three waypoints (depending on the configuration) and two VoA's as inputs, and optimize the dynamics and geometry of the trajectory for a given segment. The lower level consists of specialized functions which solve specific dynamic parameters such as the velocity and the load factor, as well as geometric parameters such as the horizontal component of a segment. For the sake of conciseness, only the high and intermediate level of the TG is presented in this thesis. The code implementing the low-level of the TG is included in Appendix-E.

The TG is also responsible for testing the dynamic feasibility of each section of every segment of the final trajectory. Dynamic unfeasibility is detected during the optimization process using mid-level functions. The high-level algorithm of the TG is defined as:

Input: A path of transitional and objective waypoints.

Output: A complete set of segments describing the trajectory in terms of its geometry, velocity, load factor, fuel consumption and dynamic feasibility.
A set of statistics characterizing the trajectory in terms of its total length, total fuel consumed and number of dynamically unfeasible segments if any.

The procedure of the high-level algorithm is illustrated as a flowchart in Figure 5.17. Sections 5.3. and 5.4. present the two main algorithms of the intermediate level. They correspond to the boxes entitled *Solve the Corner Point configuration* and *Solve the Pair of VoA configuration* in Figure 5.17.

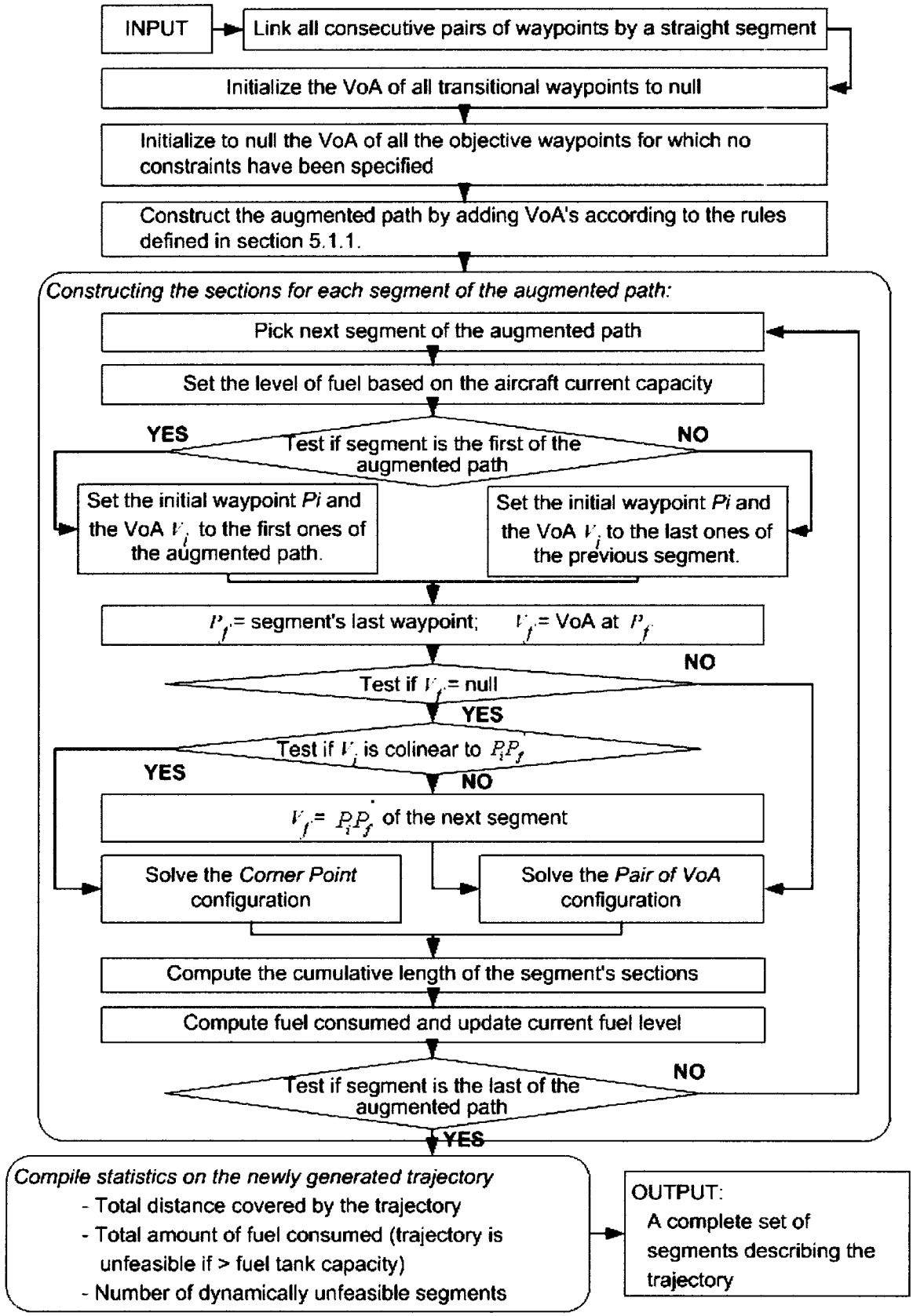


Figure 5.17: *High level algorithm of the Trajectory Generator*

5.3. Solving the configuration for pair of VoA in 3D

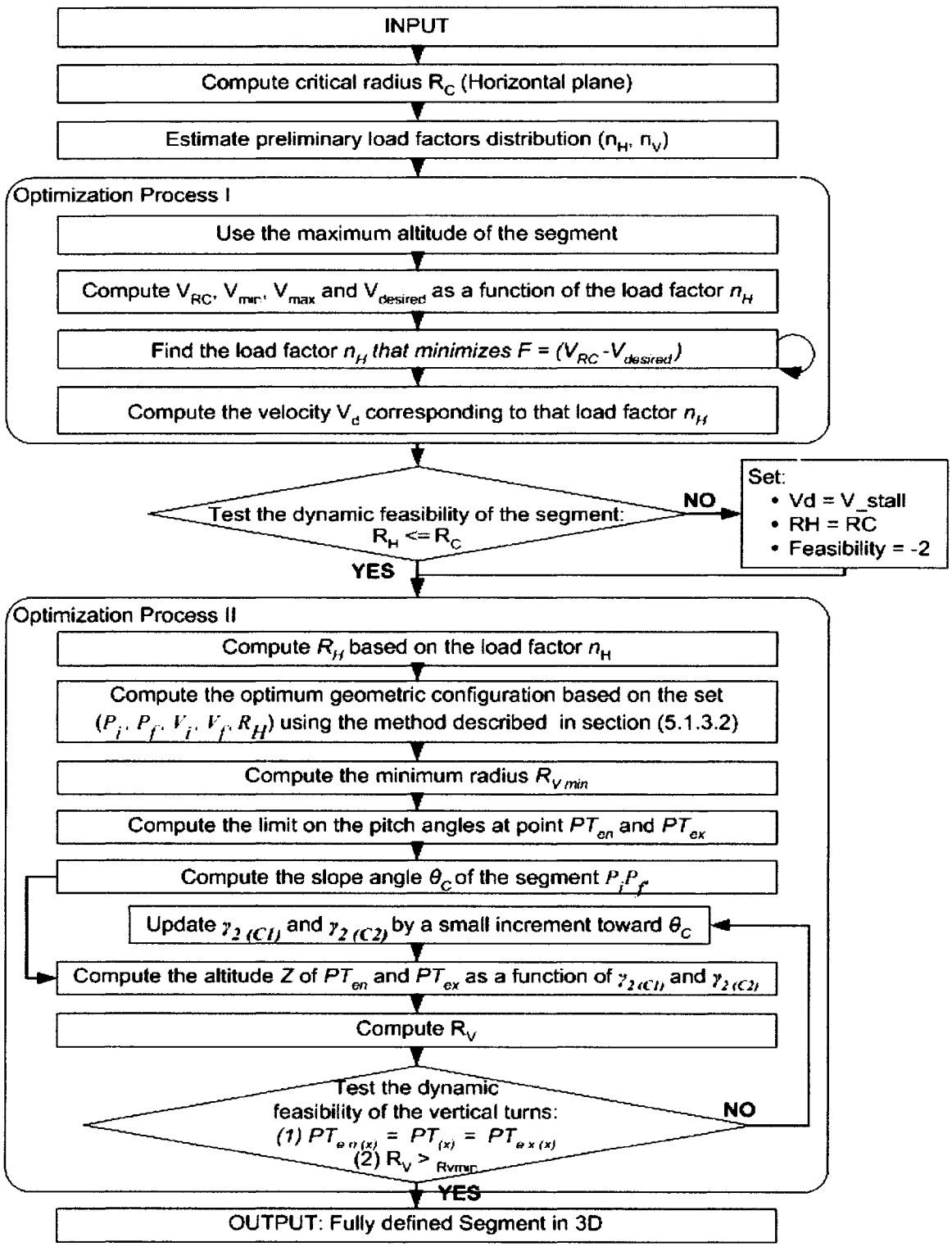


Figure 5.18: *Overall TG algorithm for a pair of VoA configuration*

The *pair of VoA* configuration corresponds to the situation where a series of sections must be linked together to define a trajectory's segment between an initial and final point P_i and P_f, associated to the vector of approach V_i and V_f. This configuration is represented in three dimensions in Figure 5.11 and a top view is shown in Figure 5.2.

Input: - An initial and final point P_i and P_f,
- An initial and final vector of approach V_i and V_f,
- The desired ASO strategy for the speed (i.e. a rating ranging from zero to ten).
- The current fuel capacity of the aircraft

Output: The complete description of a segment in terms of its geometry, velocity, load factor and dynamic feasibility.

The overview of the algorithm is shown as a flowchart in Figure 5.18. The following sub-sections of section 5.3. give the details of each of the steps shown in that flowchart.

5.3.1. Compute the critical radius R_C

For the first step of this algorithm, the critical radius R_C is calculated based on equation 5. 14 for the two circular sections required to connect the pair of waypoints of the segment. This calculation considers only the horizontal plane and takes in account the respective VoA's of each waypoint. The two circular sections of the segment are set to have equal radii.

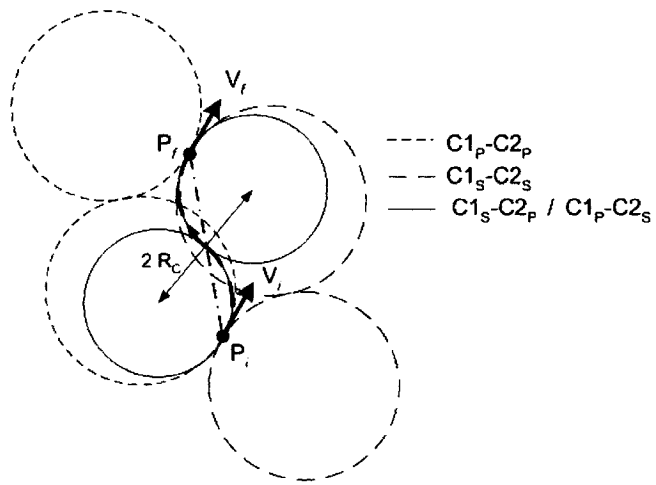


Figure 5.19: *Critical radius for a pair of VoA's*

Figure 5.19 depicts the four possible pairs of circles based on a segment P_iP_f with VoA's V_i and V_f. According to the direction of rotation imposed by the VoA's, only the combinations $C1_P$-$C2_S$ and $C1_S$-$C1_P$ are feasible (i.e. positive radii based on vectorial

geometry). The set with the smallest radii are taken for R_C. In contrast with the particular situation represented in the figure above, the radius of the two feasible sets of circles are not necessarily of the same magnitude. Their relative R_C depends on the orientation of V_i with respect to V_f.

5.3.2. Estimate preliminary load factor distribution (n_H, n_V)

For the second step of the algorithm, the preliminary load factor distribution (n_H, n_V) is estimated using a heuristic based on the total angular change between the vector P_iP_f and the VoA's V_i and V_f. The heuristic computes the ratio of the angular change in the vertical plane over the one in the horizontal plane. This ratio is used for the n_H, and n_V distribution but bounded by condition 5. 5.

5.3.3. Optimization process I

The third step of the algorithm consist of an optimization process which maximize the velocity, minimize the radius of curvature R_H of the horizontal component, and maintain the load factor n_H within allowable range. The constraints are:

- The limits of the manoeuvre envelope defining the load factor range
- The maximum speed for which $R_H \leq R_C$.
- The overall maximum speed based on the aerodynamic capabilities of the aircraft.
- The "stall buffet" speed (i.e. stalling speed during turn)

The output of this algorithm is the optimized velocity $V_{optimum}$.

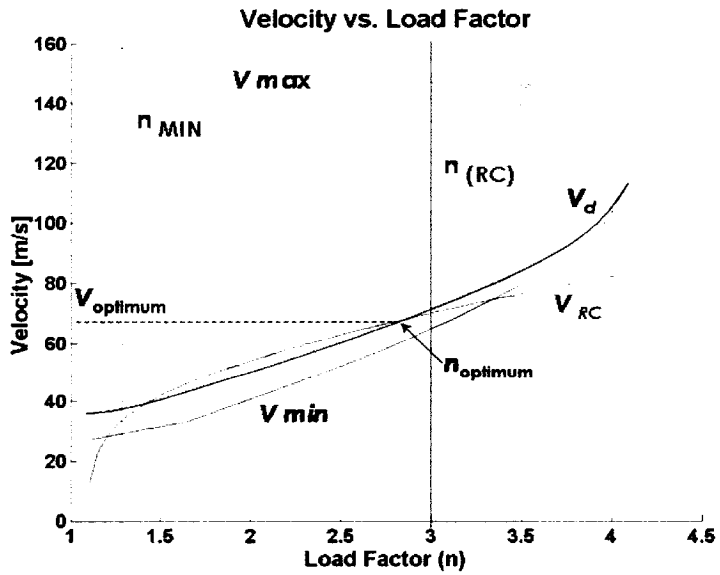


Figure 5.20: *Velocity (EAS) vs. Load factor n_H*

The maximum speed and the stalling speed listed in the constraints are function of the air density, and thus, of the altitude. Since the altitude is not necessarily constant for a segment, the worst case scenario is taken, setting the altitude to the highest one between P_i and P_f. Figure 5.20 represents the constraints mentioned above. The annotation "$R_H = max$" indicate that $R_H = R_C$. The limit $n_{(RC)}$ represent the maximum load factor n_H as defined by the preliminary load factor distribution.

An additional constraint regarding the desired velocity V_d specified by the ASO strategy must also be considered. This rating is used as a ratio between the minimum and maximum speed at a given altitude. Its value is usually higher than the maximum speed allowed by the R_C constraint as it is the case in the example given in Figure 5.20. However, the speed can still be within the V_{min}-V_{RC} range for relatively low values of the ASO strategy. The velocity profiles representing the constraints illustrated in Figure 5.20. are given by the equation 5. 16 and 5. 17. The minimum and maximum speed can be derived from equation 5. 15 which sets the thrust F_N equal to the drag D to account for steady flying conditions (*Eshelby 2000*):

$$\sigma_{hi}^{x} \cdot F_{N0} = \frac{1}{2}\rho_0 V^2 S\left(C_{DZ} + K\left(\frac{n_H W}{0.5\rho_0 V^2 S}\right)^2\right) \qquad 5.15$$

with F_{N0}, the thrust at sea level

σ_{hi}, the relative air density at altitude h_i

ρ_0, the air density at sea level

V, the equivalent air speed

S, the gross area of the wings

C_{DZ}, the constant drag coefficient (for subsonic flight)

K, the lift dependent drag factor ($\approx$ constant for subsonic flight)

n_H, the horizontal component of the load factor

W, the weight of the aircraft

x, the constant characteristics of the jet engine.

Equation 5. 15 can be expanded into a quadratic expression which can be solved to obtain the velocities V_{min} and V_{max}. The velocity profiles shown in Figure 5.20 are obtained by using V_{min} and V_{max} equations and varying them with respect to n_H.

$$V^4 C_{DZ} + V^2 \left(\frac{-2\,\sigma_{hi}^{x} \cdot F_{N0}}{\rho_0 S} \right) + 4K \left(\frac{n_H W}{\rho_0 V^2 S} \right)^2 = 0 \qquad 5.16$$

The velocity profile V_{RC} (labeled $V_{(RH = max)}$ in the example of Figure 5.20), is obtained by deriving equations D. 1 and D. 2 of Appendix-D for a fixed radius of curvature R_C:

$$V_{RC} = (R_C \cdot g \cdot \tan(\cos^{-1}(1/n_H))^{1/2} \qquad 5.17$$

Figure 5.20 illustrates the velocity profiles obtained when using the quadratic equation 5. 16 for V_{min} and V_{max}, and equation 5. 17 for $V_{(RH = max)}$.

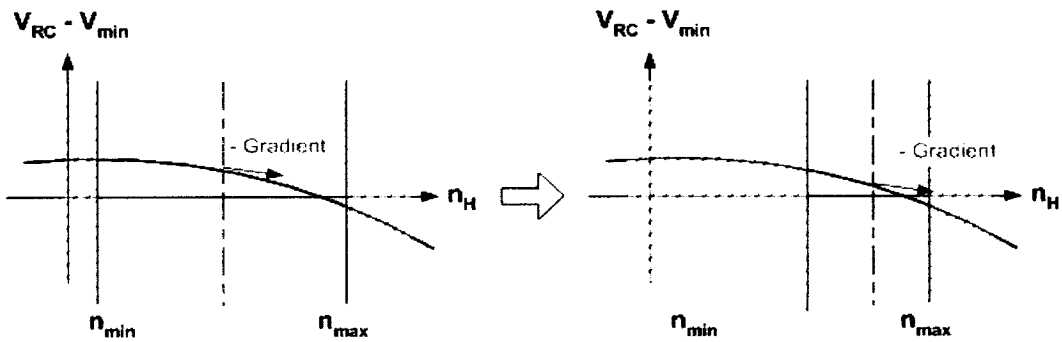


Figure 5.21: *Search algorithm for first optimization of load factor n_H vs. velocity*

The optimum velocity $V_{optimum}$ and its corresponding load factor (indicated as the *optimal point* in the example shown in Figure 5.20), is found using a standard algorithm that iteratively splits the search space in half, evaluates the gradient of the search space at that point, and selects the half leading to the smallest difference between $V_{(RH = max)}$ and V_{min}. If the desired velocity V_d set by the ASO strategy is within constraints, it is used instead of V_{min} in this search algorithm. The procedure to compute V_d is described in section 3.1. Figure 5.21 schematically represent the gradient search algorithm (i.e. steepest descent algorithm) used to find $V_{optimum}$, the output of this *optimization process I.*

5.3.4. Test the dynamic feasibility of the segment

The next step of the *pair of VoA* algorithm consists of testing the dynamic feasibility of the segment obtained from the previous step described in section 5.3.3. The segment is deemed feasible if the search algorithm finds a load factor n_H and a velocity V such that the radius of curvature of the horizontal component satisfies:

$$R_H \leq R_C \qquad 5.18$$

If the segment is not feasible:

- The velocity is set to the stall velocity
- The radius of curvature R_H is set equal to R_C
- The feasibility *flag* is set to a negative value of *-2*

The controller assigns a series of flags for each segments and each trajectory. These flags indicate the feasibility of each part of the segment and the factors that caused them to be unfeasible if that is the case. The main flag of each segment states its overall feasibility in 3-D. A series of 6 secondary flags can be switched from zero, the inactive state, to a negative value signaling unfeasibility. Two of them signal a discontinuity in the segment's geometry. Three others indicate if the rate of change of the pitch is too high at different parts of the segment, namely the two turns and the sections in between. The last secondary flag indicate if the turn is too sharp based on the maximum load bearable by the aircraft.

5.3.5. Optimization process II

Step 5 of the *pair of VoA* algorithm consists of obtaining a fully defined 3D configuration of a segment as shown in Figure 5.2. All sections of the segment are optimized by

- Minimizing radius R_H in order to avoid large overshoot from the path's segments;
- Maximizing radius R_V of the vertical turns to decrease the load on the aircraft during the maneuver;
- Optimizing the velocity to be as close as possible to the desired value defined by the ASO strategy.

The following sub-sections give the details of this optimization process. The logic of this process is given in the form of a flowchart presented in Figure 5.18.

5.3.5.1. Compute R_H

The horizontal radius of curvature R_H is computed based on the load factor n_H found previously in the *pair of VoA* algorithm, as described in section 5.3.3. Equation D. 2 of Appendix-D is used to compute the bank angle associated with the given n_H, while equation D. 1 is used to calculate the corresponding radius R_H.

5.3.5.2. Compute the optimum geometric configuration of the horizontal component

The algorithm used to compute the optimum geometric configuration of the horizontal component is described in section 5.1.3.2. This algorithm uses (P_i, P_f, V_i, V_f, R_H) as inputs. The main output variables are:

- The (x, y) coordinates of the center *C1* and *C2* of the two horizontal turns.
- The (x, y) coordinates of the two tangent points PT_{en} and PT_{ex}.
- The total angular change $\Delta\gamma_{r(C1)}$ between the velocity vector at P_i and PT_{en}.
- The total angular change $\Delta\gamma_{r(C2)}$ between the velocity vector at PT_{ex} and P_f.

5.3.5.3. Compute the minimum radius $R_{V\,min}$

The minimum radius $R_{V\,min}$ of the vertical turns for the $PT_{en}\,PT_{ex}$ section is calculated using equation D. 9 of Appendix-D. It uses the maximum load factor n_m (which is specific to each aircraft) and velocity $V_{optimum}$ resulting from the *optimization process I* described in section 5.3.3. Since the $PT_{en}\,PT_{ex}$ section is a straight line in the horizontal plane, the acceleration experienced by the aircraft has no horizontal component. Thus, the load factor used for the calculation of R_{Vmin} can be set to the maximum allowable by the aircraft.

5.3.5.4. Compute the limits on the pitch angles at point PT_{en} and PT_{ex}

The pitch angle γ_2 at point PT_{en} and PT_{ex} is computed based on the dynamic limitations of the UAV, its speed, load factor, and total angular change in yaw $\Delta\gamma_{r(C1)}$, $\Delta\gamma_{r(C2)}$ from the beginning to the end of the turn C1 and C2. Using equation D. 18 of Appendix-D with the relations given in the set of equations D. 17, the limits on pitch angles can be calculated as:

For PT_{en} :

$$\Delta\gamma_{2(C1)} = \left| \frac{(n_V - 1)\,\Delta\gamma_{r(C1)}}{V} \sqrt{\frac{g \cdot R_H}{\sqrt{(n_H^2 - 1)}}} \right| \qquad 5.19$$

$$\gamma_{2(C1)} = \gamma_2(PT_{en}) \in [\gamma_2(P_i) - \Delta\gamma_{2(C1)};\ \ \gamma_2(P_i) + \Delta\gamma_{2(C1)}] \qquad 5.20$$

For PT_{ex} :

$$\Delta\gamma_{2(C2)} = \left| \frac{(n_V - 1)\,\Delta\gamma_{r(C2)}}{V} \sqrt{\frac{g \cdot R_H}{\sqrt{(n_H^2 - 1)}}} \right| \qquad 5.21$$

$$\gamma_{2(C2)} = \gamma_2(PT_{ex}) \in [\gamma_2(P_f) - \Delta\gamma_{2(C2)};\ \ \gamma_2(P_f) + \Delta\gamma_{2(C2)}] \qquad 5.22$$

with $\Delta\gamma_{t(Ci)}$ the total angular change in the horizontal plane as defined in Figure 5.11.

$\Delta\gamma_{2(C1)}$ the total change in pitch angle along the section P_i - PT_{en}

$\Delta\gamma_{2(C2)}$ the total change in pitch angle along the section $PT_{ex} - P_f$

$\gamma_{2(Ci)}$ the final pitch angle of the turn Ci

$\gamma_{2(P_k)}$ the pitch angle at point P_k

The maximum and minimum on the pitch angle $\gamma_{2(C1)}$ are given in equation 5. 20 as the pitch angle at the segment's first point P_i plus or minus the total angular change $\Delta\gamma_{2(C1)}$ allowable for the turn $C1$. The limitations on the pitch angle $\gamma_{2(C2)}$ is given in equation 5. 21 as the pitch angle at the tangent point PT_{ex} at the end of the segment's middle sections, plus or minus the total angular change $\Delta\gamma_{2(C2)}$ allowable for the turn $C2$.

5.3.5.5. Compute the slope of the line P_iP_f

The slope of the straight line between P_i and P_f corresponds to an angle referred to as θ_C in the program code in Appendix-E. Figure 5.22 shows how the angle θ_C is defined geometrically. This angle is used as a reference angle for the search algorithm that optimizes the values of the final pitch angle $\gamma_{2(C1)}$ of the turn $C1$, and the initial pitch angle $\gamma_{2(C2)}$ of the turn $C2$. Making $\gamma_{2(C1)}$ and $\gamma_{2(C2)}$ converge toward the angle θ_C maximizes the radius R_V of the two vertical turns required to go from PT_{en} to PT_{ex}. Thus, it reduces the magnitude of the vertical load factor n_V.

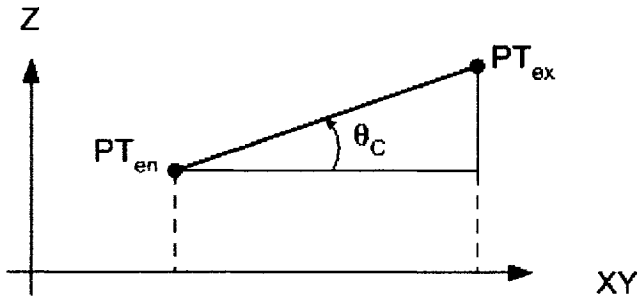


Figure 5.22: *$PT_{en}PT_{ex}$ section*

5.3.5.6. Update $\gamma_{2\,(C1)}$ and $\gamma_{2\,(C2)}$

The radius of curvature of the two vertical turns between PT_{en} and PT_{ex} needs to be maximized in order to have the lowest vertical load factor possible. The optimal case is a configuration where the trajectory between PT_{en} and PT_{ex} is a straight line. In such a case the radius of curvature is infinite and the load factor is minimal. This ideal straight trajectory is used as reference and is defined using the angle θ_C. The pitch angles $\gamma_{2\,(C1)}$

and $\gamma_{2\,(C2)}$ are iteratively updated toward this optimum angle, although the latter may not necessarily be achievable based on the constraints imposed on the section preceding PT_{en} and the one following PT_{ex}. The algorithm used to update $\gamma_{2\,(C1)}$ and $\gamma_{2\,(C2)}$ is as follows:

- If $\gamma_{2\,(Ci)} < \theta_C$, the limiting range $[\gamma_{2\,(Ci)\,min}\,;\,\gamma_{2\,(Ci)\,max}]$ is modified by setting the lower bound to the value of $\gamma_{2\,(Ci)}$ while the upper bound remains unchanged.
- Otherwise, the upper bound of the limiting range is set to $\gamma_{2\,(Ci)}$ while the lower bound remains unchanged.
- Once the limiting range has been modified, the value of $\gamma_{2\,(Ci)}$ is set to middle value of the limiting range.

5.3.5.7. Compute the new values for the altitude Z of PT_{en} and PT_{ex}

Modifying the pitch angles $\gamma_{2\,(C1)}$ and $\gamma_{2\,(C2)}$ modifies the total angular variations $\Delta\gamma_{2(C1)}$ and $\Delta\gamma_{2(C2)}$, and thus, changes the altitude profiles of the sections P_iPT_{en} and $PT_{ex}P_f$. Therefore, the altitude of the points PT_{en} and PT_{ex} needs to be recalculated after updating $\gamma_{2\,(C1)}$ and $\gamma_{2\,(C2)}$. Those values are found using the equations of a log section if $\gamma_{2(C1)} = \gamma_2\,(P_i)$ and the one of a helix section otherwise. The equations of the log and helix sections are given in section 5.1.2.3. and 5.1.2.4. respectively.

5.3.5.8. Compute R_V

The sections between PT_{en} and PT_{ex} consist of two consecutive vertical turns as shown in Figure 5.23.

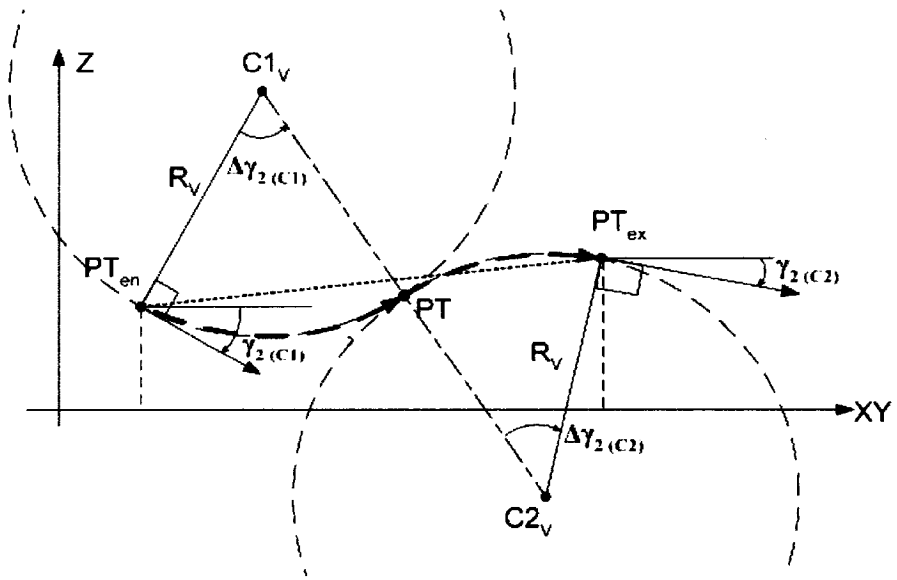


Figure 5.23: *Vertical turns configuration*

Once the new values for the altitudes of PT_{en} and PT_{ex} have been found, the radius of curvature R_V of the vertical turns can be calculated based on the following relations:

$$\begin{aligned} C1_V &= \left[P_{i(x)} - R\cdot\sin\left(\gamma_{2(C1)}\right)\cdot s;\ P_{i(y)} - R\cdot\cos\left(\gamma_{2(C1)}\right)s\right] \\ C2_V &= \left[P_{f(x)} - R\cdot\sin\left(\gamma_{2(C2)}\right)\cdot s;\ P_{f(y)} - R\cdot\cos\left(\gamma_{2(C2)}\right)s\right] \\ R_V &= \frac{\left\|\overrightarrow{C1_V C2_V}\right\|}{2} \\ s &= \sin(\pm\pi/2) \end{aligned} \qquad 5.23$$

The point $P_{n\,(x,\,y)}$ corresponds to the projection of $P_n\ (X,Y,Z)$ into the vertical plane in which PT_{en} and PT_{ex} are contained. Given that this plane makes an angle γ_3 with the vertical plane passing through the reference X-axis, the projection of P_n can be found according to equation.

$$\begin{aligned} P_{n\,(x)} &= P_{n\,(X)}/\cos(\gamma_3) = P_{n\,(Y)}/\sin(\gamma_3) \\ P_{n\,(y)} &= P_{n\,(Z)} \end{aligned} \qquad 5.24$$

Equation 5. 24 uses a transform matrix describing a rotation around the Z-axis with an angle γ_3. The symbol s used in the set of equation 5. 23 can take the value +1 or -1 depending if the vertical turn's center is, respectively, above (i.e. $C1_V > PT_{en}(Z)$ and $C2_V > PT_{ex}(Z)$) or below the corresponding tangent point PT_n. The combinations of the set of equations 5. 23 result in a quadratic expression which is a function of R_V. The complete equation can be found in Appendix-D, under the function called "compute_R_critical" in the file TGenerator.cpp.

5.3.5.9. Test the dynamic feasibility of the vertical turn sections

In order to be feasible, two dynamic conditions and one geometric condition have to be satisfied. The vertical turn sections need to have a radius of curvature higher than the one defined by the maximum vertical load factor n_V allowable for a given maneuver. The pitch angle need to be within the allowable range for the inflection point PT where the section starting from PT_{en} meet the one ending at $PT_{ex,}$. Finally, the inflection point PT has to be positioned within PT_{en} and PT_{ex}. The first two conditions mentioned above define the dynamic feasibility of the sections while the last one define its geometric feasibility as represented in Figure 5.24.

The conditions defining the dynamic feasibility of the two vertical turns are defined by the following equations:

$$R_{V\,min} \leq R_V \qquad 5.25$$

$$\gamma_{2\,min} < \gamma_2(PT) < \gamma_{2\,max} \qquad 5.26$$

with $(\gamma_{2\,max}, \gamma_{2\,min})$ the maximum and minimum pitch angles of the aircraft.

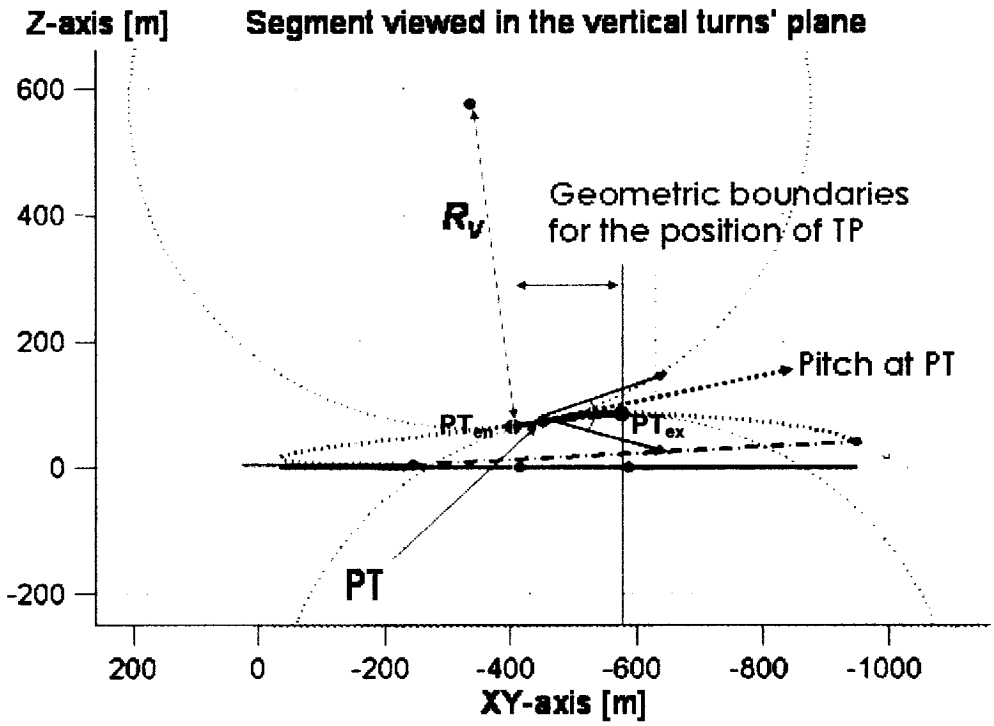


Figure 5.24: *Geometric constraint on tangent point between vertical turns $C1_V$ and $C2_V$*

The point PT and the pitch angle associated to it can be found based on the coordinates of $C1_V$ and $C2_V$ which have been determine previously in step 8, section 5.3.5.8.

$$\overrightarrow{C1_V PT} = 1/2\ \overrightarrow{C1_V C2_V} \qquad 5.27$$

The geometric feasibility of the vertical turn sections may be verified based on the following equation which checks that PT is within the allowable range:

$$PT_{e\,n\,(x)} \leq PT_{(x)} \leq PT_{ex\,(x)} \qquad 5.28$$

5.3.6. Iterate optimization process II

The algorithm of the *pair of VoA configuration* is designed as an iterative process which optimizes the trajectory between two waypoints by progressively modifying its geometry while checking its dynamic feasibility. The *optimization process I* defines the limits on the dynamic parameters of the trajectory, specifying ranges for velocities and load factors. The algorithm defines and optimizes the trajectory in the *optimization process II.* This second optimization process runs until all geometric and dynamic constraints are satisfied to produce a feasible trajectory segment. A maximum number of iterations is also specified, letting the algorithm abort its search if a feasible segment can't be found within a predefined amount of time. The limit on the number of iterations should be fixed to a value between 8 and 12 for optimal performances in terms of processing speed and level of optimization of the resulting trajectory. Based on experimental observations, lower values give poorer results while higher ones slow the algorithm without significant gains.

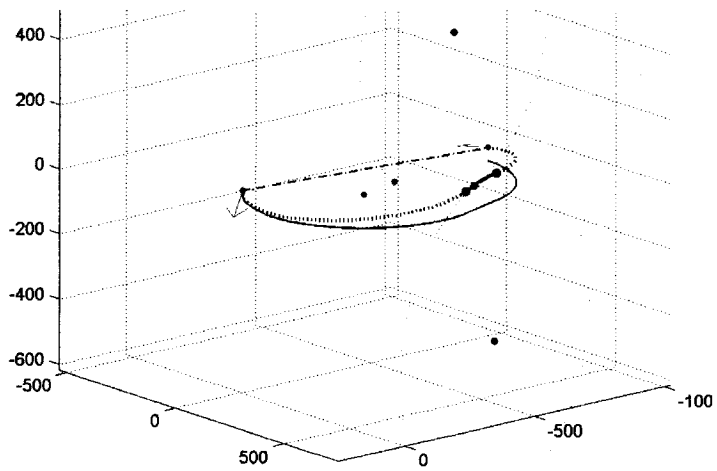


Figure 5.25: *Fully defined segment in 3D*

Once feasibility has been reached, the resulting segment is fully defined in 3D. For each section, the load factor, velocity and geometry are stored into memory. Figure 5.25 represents a fully defined segment composed of two log sections in doted lines, and two vertical turn sections in bold. The blue path is the projection of the segment in the horizontal plane. The blue points represent the center of circle *C1* and *C2*, while the red points represent the center of the vertical turns $C1_V$ and $C2_V$.

5.4. Solving the configuration for a corner point

The algorithm outlined in section 5.3. solves the most general case where a segment has a VoA's specified for each of its two end points. The *corner point configuration* is a more specific case where only the first waypoint has a VoA. As shown in Figure 5.26, this configuration is mainly characterized by the critical radius R_C defined as the maximum radius R_H allowable by the geometry of the segments $P_i P_{mid}$ and $P_{mid} P_f$. The calculations required to find R_C have been presented in section 5.1.3.1. While the dynamic limitations of the aircraft gives a lower bound on the radius of curvature R_H, the geometry of the corner point configuration gives an upper bound. Considering this additional geometric constraint on R_H and that no VoA is specified at the end point P_f, the segment $P_i P_f$ needs to be adapted to fit the input format of the algorithm derived for the pair of VoA configuration. Adapting the segment to a working algorithm rather than redesigning one from scratch is more desirable when considering the principle of reusability and maintainability in software engineering. Indeed, having a single algorithm solving both configurations reduces the amount of code to be tested and maintained.

Adapting the corner point configuration to a pair of VoA configuration requires four steps. Firstly, a VoA V_f is defined for the waypoint P_f. It is set as a unit vector pointing in the direction of the segment $P_{mid} P_f$:

$$V_f = (P_f - P_{mid}) / || P_f - P_{mid} || \qquad 5.29$$

Secondly, the critical radius of curvature setting the upper bound on R_H is calculated and compared to the minimum radius of curvature defined by the load factor limit of the aircrafts. If the geometry of the corner point configuration gives a critical radius of curvature smaller than the minimum one imposed by the dynamic limits of the aircraft, the segment is deemed unfeasible and a set of default values is given for the geometry and the velocity of the segment. If the critical radius of curvature is dynamically feasible, the process of adapting the configuration to a pair of VoA continues. The third step consists of replacing the end point of the segment P_f by a point P_f'. This point corresponds to the tangent point of the circle R_H to the line $P_{mid} P_f$. The last step in the adaptation of the corner point configuration consists of setting the first section of the segment (which is either a log or helix section for the pair of VoA configuration) to a

horizontal turn with zero angular change. This operation effectively sets the first tangent point PT_{en} equal to the first segment's point P_i.

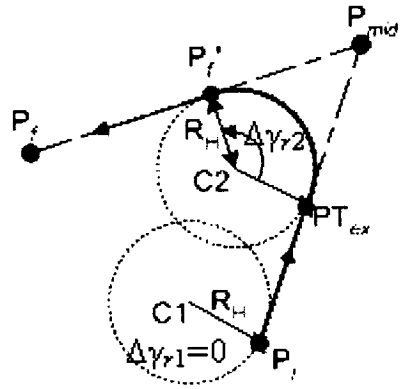


Figure 5.26: *Configuration of a pair of VoA's for a corner point*

The implementation of the algorithm for the corner point configuration reuses the code of the pair of VoA configuration. Its performances are optimized by skipping the calculation of the first segment's section, directly assigning the proper value to the relevant parameter (i.e. setting PTen = Pi). Thus, the optimization process II described in section 5.3.5. does not require any calculations for the pitch angle $\gamma_{2\,(C1)}$, which stays constant and equal to the pitch angle of P_i.

Chapter 6: Multi-Objectives Genetic Algorithm

The Multi-Objectives Genetic Algorithm (MOGA) and the TG are responsible for generating an ordered list of objectives for a given UAV; generating and optimizing the path that completes all objectives; and generating an optimized trajectory in terms of the load factor, the velocity, and the geometry for a given sequence of waypoints.

Conceptually, the MOGA is only responsible for the first and second steps described above. However, the complexity of the TAP comes from the fact that these three aspects are interdependent. The cost resulting from a path depends on the optimality of the trajectory, while the trajectory's optimality depends on the sequence of waypoints constituting the path. The TAP being categorized as "NP hard" as discussed in section 2.1. , standard algorithms based on exhaustive search are inefficient if not useless. Thus, the approach proposed for this work consists of using a certain type of random search algorithm. Since the time frame in which the controller has to find a solution may vary depending on the mission type and its current status, the key is to use an algorithm that finds a solution that is increasingly optimized as the allotted time increases. Such algorithms are known as "any-time algorithm" and include evolutionary computing, randomized optimization, and the more commonly known Genetic Algorithms. The latter offers the advantage of allowing the input space to be divided into segments, which can be individually searched through by parallel processors. Thus, the optimization problem could be equally divided among several CPU's or several UAV's of a fleet, speeding-up the search while ensuring that the loss of one UAV will not prohibit further processing. This feature adds robustness to the overall search algorithm.

6.1. Basics of genetic algorithms

The basic steps of a GA are as follows:

1. Generate a random population of N Paths, each one representing a potential solution of the TAP.
2. Evaluate the fitness F_i of each individual i based on a cost function F most commonly referred as "fitness function" in GA-specialized literature.
3. Select n Paths from the current path population PP_1 with a probability (p).

$$p_i = F_i / \sum_{j=1}^{N} F_j \qquad 6.1$$

4. Generate a population PP_2 of $(2n)$ paths using the *Crossover* operation on the n selected paths.
5. Use *Mutation Operators* on a small fixed percentage (Y) of the population PP_2 (using a value in the order of 5%).
6. Select $(N - 2\ n)$ Paths from PP_1 with probability (p) and add them to PP_2 using the "roulette wheel selection" technique.
7. Delete PP_1 and repeat steps 2 to 7 for a predetermined number (X) of generations, or until convergence is reached.

There are five fundamental elements that make a GA. They are: the chromosome, which defines the format of a candidate solution; the population, which defines the number of candidates being considered at the same time; the fitness function, which assess the level of relative optimality of each candidate with respect to the rest of the population; the genetic operators, which are the core of the evolutionary process, providing the means to produce fittest candidates; the selection process, which determine which individual should be removed or kept in a population; and the termination condition(s) which determines when the GA should stop. The following sections describe these elements.

6.2. Chromosome implementation of paths

The complexity of a Genetic Algorithm is linearly proportional to the size of its chromosome. In addition, the size of a population will be limited by the amount of memory available to the system, and thus by the size of each individual constituting that population. Thus, it is imperative to strip down the internal representation of a solution to the minimum.

Since a complete description of a UAV's trajectory would result in an inefficient use of the system's resources, it was decided that a chromosome would simply consist of the sequence of key waypoints defined as nodes. This approach makes the overall system architecture more modular and resource efficient. In order to differentiate objectives and intermediary waypoints, two types of nodes are considered:

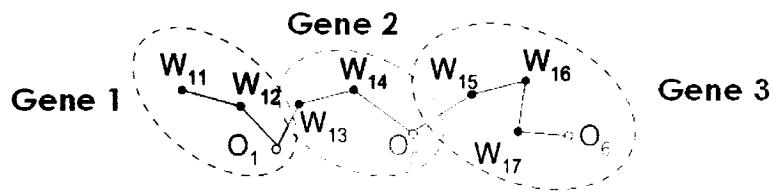
- *Transitional nodes*:

  They correspond to intermediate points linking objectives. They consist of only one element, a waypoint given in GPS coordinates.

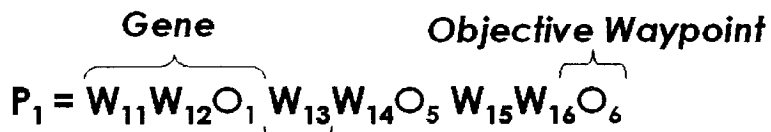
- *Objective nodes*:

  They correspond to a single waypoint associated with an objective as defined in section 3.2. The structure of these nodes is meant to incorporate all objective related constraints, and to implement both objective and goal waypoints. Objective nodes are composed of the following elements:

  - A reference to the associated objective.
  - An index corresponding to the position of that point within an objective's goal sequence. If the corresponding objective is represented by a single waypoint (i.e. the sequence contains only one element), the index is set to zero and the coordinates of that point are used as waypoint for the objective node.
  - A vector of approach defining from which direction the objective's waypoint should be approached. If no vector of approach is specified, the vector is set to the null vector.
  - The desired time of arrival. If this element is of no concern for a mission, the time is set to a value less or equal to zero.



(a)



(b)

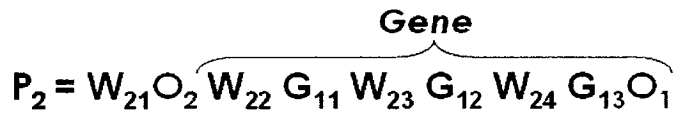


Figure 6.1: *Chromosomes and Genes*

Since transitional waypoints are randomly generated by the GA, the formal definition of a chromosome can be interpreted as: *"One possible path passing through all goal and objective waypoints of a mission"*.

A gene is defined as the sequence of transitional and goal waypoints preceding an objective waypoint. Since each objective have a unique identifier, the terminating objective waypoint of a gene is used as the identifier of that gene. Figure 6.1 (a) shows graphically the structure of a chromosome and its gene. Figure 6.1 (b) shows a chromosome containing one gene which has three goal waypoints.

During the mutation process of the GA, the relative position of goal waypoints within a gene cannot be altered. Transitional nodes can be inserted, deleted or moved anywhere within the gene, but goal waypoints remain in the same order. Moreover, it is assumed that no objective nodes belonging to another sequence should be executed until the first sequence is completed. Thus, no objective waypoint "O_j" can be inserted within a goal sequence belonging to a gene with a terminating objective "O_k".

6.3. Population

The population is a pool of solutions that can either be feasible or unfeasible with respect to the constraints of the problem. A given set of solutions between two mutation cycles is referred as a "generation". The purpose of a population is to provide a representative sample of the search space, such that most of the elements required for an optimal solution are present within that pool. The goal of the mutation cycles is to construct new solutions by directly manipulating these elements through what is called "genetic operators". In essence, new solutions are built based on the knowledge inherited from previous generations. To complete the process, some mechanisms also exist to introduce new elements that are not present within the solution pool.

6.4. Genetic operators

A series of operators are used during the evolutionary process of the GA. Their purpose is to modify a population in order to get more optimal paths. Genetic operators are divided into two groups, the *crossover* operator and the *mutation operators*. The *crossover* operator is the tool generating new candidate solution, while *mutation operators* are used to induce small perturbations in these new candidate solutions.

At each generation, part of the pool is replaced by new chromosomes referred as offspring. This process is twofold. First, the crossover operation is performed on a predefined percentage of the population, creating a new set of chromosomes. Secondly, another fixed percentage of the set of new chromosome is altered by using mutation operators. The results are the offspring that will be integrated into the population to form the next generation.

6.4.1. Crossover

This is the basic operator on which GA's are based. The classic approach consists of taking two chromosomes, cutting them in two parts, and recombining each one with the completing segment of the other chromosome. Due to the particularities of the current application, the point at which a chromosome is cut in two can't be determined randomly. A path implementing a continuous action (such as surveillance) will require that a sequence of goals be reached in a specific order within the path. Thus, the part of the chromosome corresponding to that sequence should not be split and added to another part of a sequence which does not match the same objective. Moreover, two paths of the same population will have the same list of objectives, but ordered differently, and with different transitional nodes between them. Thus, a classic approach to crossover may result in chromosomes having the same objective repeated more than once.

In order to keep the resulting chromosomes meaningful toward the search of new solution, the crossover operator has been implemented in three levels: the gene level, the goal sequence level, and the transitional node level.

6.4.1.1. Crossover at the gene level

This procedure is referred as *Pr1* in this section. It consists of taking two chromosomes and combining the waypoints sequence preceding each objective as shown in Figure 6.2. This procedure is broken down into 6 steps. To avoid any confusion with other procedures used in the crossover operation, these steps are preceded by *Pr1* which stands for "procedure one".

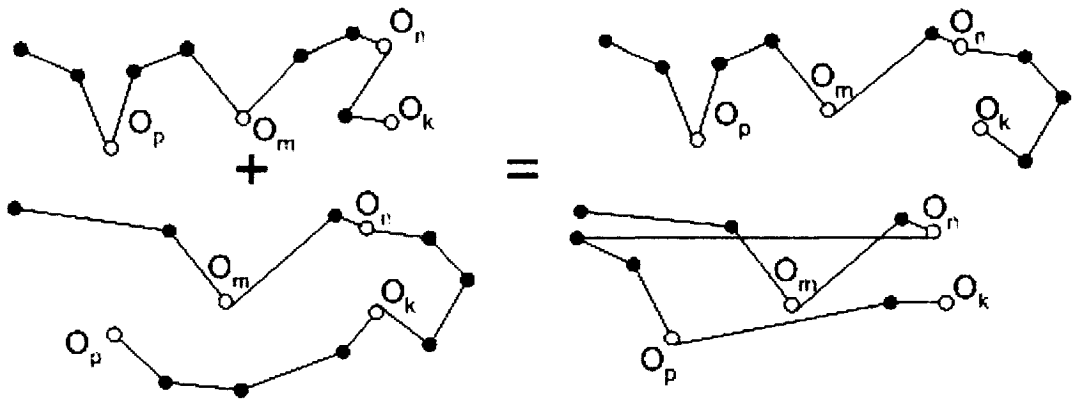


Figure 6.2: *General case for the crossover operation*

Pr1.1. *Chromosomes selection*:

Two paths are randomly selected for crossover based on a probability proportional to their fitness. Assuming that W_{ij} is the sequence j of transitional nodes belonging to path i, and that O_k correspond to the objective k, we can represent two parent paths as:

$$P1 = W_{11}\, O_1\, W_{12}\, W_{13}\, O_3\, W_{14}\, O_2\, W_{15}\, W_{16}\, W_{17}\, O_4$$
$$P2 = W_{21} W_{22}\, O_4\, W_{23}\, O_3\, W_{24}\, W_{25}\, O_1\, W_{26}\, O_2$$

The sequences W_{ij} can be of any length, the crossover operation remaining unaffected by that aspect. The parents above are taken as an example. The general case doesn't have to be limited to the number of objective and transitional nodes shown in it.

Pr1.2. *Genes identification*:

The two paths are divided into sub-segments such that each one of them have exactly one objective node.

$$P1 = W_{11} O_1 |\, W_{12} W_{13}\, O_3 |\, W_{14}\, O_2 |\, W_{15} W_{16} W_{17}\, O_4$$
$$P2 = W_{21} W_{22}\, O_4 |\, W_{23}\, O_3 |\, W_{24} W_{25}\, O_1 |\, W_{26}\, O_2$$

Pr1.3. *Splitting chromosomes in halves*:

The cutting point is chosen at the middle gene. If the number of elements N is even, the cutting point is after the gene at the $(N/2)$ position. If the number of elements is odd, the position is determined randomly between $(N/2+1)$ and $(N/2-1)$. The cutting point is the same for each chromosome.

$$P1 = W_{11}\, O_1 |\, W_{12} W_{13}\, O_3 \,||\, W_{14}\, O_2 |\, W_{15} W_{16} W_{17}\, O_4$$
$$P2 = W_{21} W_{22}\, O_4 |\, W_{23}\, O_3 \,||\, W_{24} W_{25}\, O_1 |\, W_{26}\, O_2$$

Pr1.4. *Generating first half of the children*:

The first half of the parents is given directly to their offspring.

$$C1 = W_{11} O_1 | W_{12} W_{13} O_3 \,||\, X_1$$
$$C2 = W_{21} W_{22} O_4 | W_{23} O_3 \,||\, X_2$$

Pr1.5. *Generating second half of children*:

Two lists of genes are established for each child. For child *C1*, this list corresponds to the genes of parent *P2*, minus all the genes of *P2* which have the same objective nodes as *C1*. The list of *C2* is determined in the same way, although the index number is switched.

$$L1 = \overline{P2(O_i) \cap C1(O_i)} = W_{21} W_{22} O_4 | \, W_{26} O_2$$
$$L2 = \overline{P1(O_i) \cap C2(O_i)} = W_{11} O_1 | \, W_{14} O_2$$

Pr1.6. *Resulting chromosomes*:

The list *Li* is appended to the end of *Ci*, giving the final result:

$$C1 = W_{11} O_1 \, W_{12} W_{13} O_3 \, W_{21} W_{22} O_4 \, W_{26} O_2$$
$$C2 = W_{21} W_{22} O_4 \, W_{23} O_3 \, W_{11} O_1 \, W_{14} O_2$$

6.4.1.2. Crossover at the goal sequence level

In the case of paths containing goal sequence(s), an extra step is used to crossover the sequence without interrupting it with other objective nodes. The crossover operation for that specific case is represented in Figure 6.3. The procedure is broken down into 9 steps preceded by the *Pr2* indication.

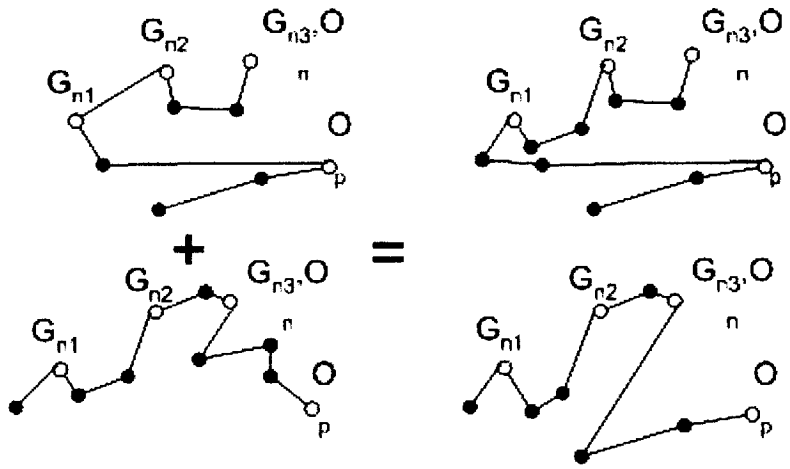


Figure 6.3: *Special case of crossover operation on a path including a goal sequence*

Pr2.1. *Chromosome selection*:

As before, two chromosomes are selected.

$$P1 = W_{11}\, O_2\, W_{12}\, G_{11}\, W_{13}\, G_{12}\, W_{14}\, G_{13}\, O_1\, W_{15}\, O_3$$
$$P2 = W_{21}\, G_{11}\, W_{22}\, G_{12}\, W_{23}\, G_{13}\, O_1\, W_{25}\, O_2\, W_{25}\, O_2$$

The notation introduces an objective node O_i right after the last goal G_{ij} of the sequence associated to it. It indicates the end of the sequence. As in the previous case, these parents represent an example rather the general case.

Pr2.2. *Genes identification*:

$$P1 = W_{11}\, O_2 | W_{12}\, G_{11}\, W_{13}\, G_{12}\, W_{14}\, G_{13}\, O_1 | W_{15}\, O_3$$
$$P2 = W_{21}\, G_{11}\, W_{22}\, G_{12}\, W_{23}\, G_{13}\, O_1 | W_{24}\, O_3 | W_{25}\, O_2$$

Pr2.3. *Chromosome fragmentation*:

$$P1 = W_{11}\, O_2 \,||\, W_{12}\, G_{11}\, W_{13}\, G_{12}\, W_{14}\, G_{13}\, O_1 \,|\, W_{15}\, O_3$$
$$P2 = W_{21}\, G_{11}\, W_{22}\, G_{12}\, W_{23}\, G_{13} O_1 \,||\, W_{25}\, O_3 \,|\, W_{25}\, O_2$$

which is rewritten to lighten the notation as:

$$P1 = W_{11}\, O_2 \,||\, PGS1\, O_1 |\, W_{15}\, O_3$$
$$P2 = PGS2\, O_1 \,||\, W_{25}\, O_3 \,|\, W_{25}\, O_2$$

Note: $PGS_N\, O_J$ = Parent Goal Sequence N for Objective J

Pr2.4. *Sequence fragmentation*:

Fragmentation is repeated for each pair of goal sequence from the two parent paths.

$$PGS1 O_1 = W_{12}\, G_{11} | W_{13}\, G_{12} | W_{14}\, G_{13}\, O_1$$
$$PGS2 O_1 = W_{21}\, G_{11} \,|\, W_{22}\, G_{12} \,|\, W_{23}\, G_{13}\, O_1$$

Pr2.5. *Generating first half of the new goal sequences*:

$$CGS1\, O_1 = W_{12}\, G_{11} \,|\, W_{13}\, G_{12} \,||\, GLS1\, O_1$$
$$CGS2\, O_1 = W_{21}\, G_{11} \,|\, W_{22}\, G_{12} \,||\, GLS2\, O_1$$

Note: $CGS_N\, O_J$ = Child Goal Sequence N for Objective J

with:

$$GLS1\, O_1 = \overline{S2(O_i) \cap CS1(O_i)} = W_{23}\, G_{13}\, O_1$$
$$GLS2\, O_1 = \overline{S1(O_i) \cap CS2(O_i)} = W_{14}\, G_{13}\, O_1$$

Note: GLS=Goal List Sequence N

giving:

$$CGS1\, O_1 = W_{12}\, G_{11} \,|\, W_{13}\, G_{12} \,||\, W_{23}\, G_{13}\, O_1$$
$$CGS2\, O_1 = W_{21}\, G_{11} \,|\, W_{22}\, G_{12} \,||\, W_{14}\, G_{13}\, O_1$$

Pr2.6. *Generating first halves of children*:

$C1 = W_{11}\ O_2 \,||\, X_1$
$C2 = PGS2\ O_1 \,||\, X_2$

Pr2.7. *Generating second halves of children*:

$L1 = \overline{P2(O_i) \cap C1(O_i)} = PGS2\ O_1 \mid W_{25}\ O_3$
$L2 = \overline{P1(O_i) \cap C2(O_i)} = W_{11}\ O_2 \mid W_{15}\ O_3$

Pr2.8. Sequences substitution in offspring:

Parent goal sequences $PGS_N\ O_J$ are replaced by child goal sequence $CGS_N\ O_J$.

$L1 = CGS2\ O_1 \mid W_{25}\ O_3$
$L2 = W_{11}\ O_2 \mid W_{15}\ O_3$

Pr2.9. *Resulting offspring*:

$C1 = W_{11}\ O_2\ W_{21}\ G_{11}\ W_{22}\ G_{12}\ W_{14}\ G_{13}\ O_1\ W_{25}\ O_3$
$C2 = W_{21}\ G_{11}\ W_{22}\ G_{12}\ W_{23}\ G_{13}\ O_1\ W_{11}\ O_2\ W_{15}\ O_3$

6.4.1.3. Crossover at the transitional node level

When there is more than one transitional node "W_{ij}" preceding an objective node O_k, the crossover operation is performed at the gene level. This procedure is referred as *Pr3* and goes as follows:

Pr3.1. *Chromosomes selection*:

$P1 = W_{11}\ O_1\ W_{12}\ W_{13}\ O_3\ W_{14}\ O_2\ W_{15}\ W_{16}\ W_{17}\ O_4$
$P2 = W_{21} W_{22}\ O_4\ W_{23}\ O_3\ W_{24}\ W_{25}\ O_1\ W_{26}\ O_2$

Pr3.2. *Crossover operation at gene level*:

Crossover is performed on the selected chromosomes according to the procedure *Pr1*. The resulting offspring are:

$C1 = W_{11}\ O_1\ W_{12} W_{13}\ O_3\ W_{21} W_{22}\ O_4\ W_{26}\ O_2$
$C2 = W_{21} W_{22}\ O_4\ W_{23}\ O_3\ W_{11}\ O_1\ W_{14}\ O_2$

Pr3.3. *Finding corresponding genes between child Ci and parent Pi*:

Since child *Ci* already contains the genes of parent *Pj*, crossover must be performed between child and parent chromosomes having the same index. Since only genes with more than one transitional node are considered for this procedure, the following pairs between *C1* and *P1* can be identified:

Table 6.1: *Pair of genes between parents and children for transitional node level crossover*

Parent	Genes with more than one transitional node	Child	Corresponding genes
(P1):	$W_{12}W_{13}\,O_3$	(C1):	$W_{12}W_{13}\,O_3$
	$W_{15}W_{16}W_{17}\,O_4$		$W_{21}W_{22}\,O_4$
(P1):	$W_{21}W_{22}\,O_4$	(C2):	$W_{21}W_{22}\,O_4$
	$W_{24}W_{25}\,O_1$		$W_{11}\,O_1$

Pr3.4. *Finding cutting points*:

Each gene is cut in half. If the gene has only one transitional node, the cutting point is taken before that node. If there is an odd number n of transitional nodes, the cutting point is randomly chosen between the closest integer below $(n/2)$ and the one above.

Table 6.2: *Cutting point for transitional node level crossover*

Parent	Genes with more than one transitional node	Child	Corresponding genes
(P1):	$W_{12} \mid W_{13}\,O_3$	(C1):	$W_{12} \mid W_{13}\,O_3$
	$W_{15}W_{16} \mid W_{17}\,O_4$		$W_{21} \mid W_{22}\,O_4$
(P1):	$W_{21} \mid W_{22}\,O_4$	(C2):	$W_{21} \mid W_{22}\,O_4$
	$W_{24} \mid W_{25}\,O_1$		$\mid W_{11}\,O_1$

Pr3.5. *Replacing the second halves of the children's genes*:

The second half of each child's gene is replaced with the one of the corresponding parent. Due to the procedure used to reorganize genes between chromosomes (i.e. procedure *Pr1*), some genes may regain their original form by performing this step.

Table 6.3: *Transitional nodes crossover*

Parent	Genes with more than one transitional node	Child	Corresponding genes
(P1):	$W_{12} \mid W_{13}\,O_3$	(C1):	$W_{12}\,[\,W_{13}]\,O_3$
	$W_{15}W_{16} \mid W_{17}\,O_4$		$W_{21}\,[\,W_{17}]\,O_4$
(P1):	$W_{21} \mid W_{22}\,O_4$	(C2):	$W_{21}\,[W_{22}\,]\,O_4$
	$W_{24} \mid W_{25}\,O_1$		$[W_{25}\,]\,O_1$

Pr3.6. *Resulting chromosomes*:

The modified genes are reinserted into the children's chromosomes.

$$C1 = W_{11}\, O_1\, W_{12}\, W_{13}\, O_3\, W_{21}\, W_{17}\, O_4\, W_{26}\, O_2$$
$$C2 = W_{21}\, W_{22}\, O_4\, W_{23}\, O_3\, W_{25}\, O_1\, W_{14}\, O_2$$

6.4.2. Mutation operators

Mutation operators are applied to a subset of the offspring obtained from the crossover operation. Only one mutation operator can be applied to each chromosome of that subset. Most mutation operators can only be applied under certain conditions. This implies that even if the selection of a mutation operator is random, only a subset of these operators may be applicable for a given chromosome. If a chromosome has been selected for mutation, the system will attempt to use all five operators until one can be applied. If none works, the mutation phase is skipped for that chromosome.

This section presents the five mutation operators.

6.4.2.1. Perturb transitional

This operator picks a transitional waypoint from a path and randomly modifies the coordinates of its waypoint. This operation is schematically represented in Figure 6.4. In order to avoid wasting resources on unfeasible path, this operator ensures that the new coordinates of the waypoint always correspond to a free cell from the Octree. This operator is meant to induce small perturbation in a chromosome. Thus, the change of coordinates is limited to a local region for which the initial waypoint is the center. If no free cells are found, another point is selected and the process is repeated.

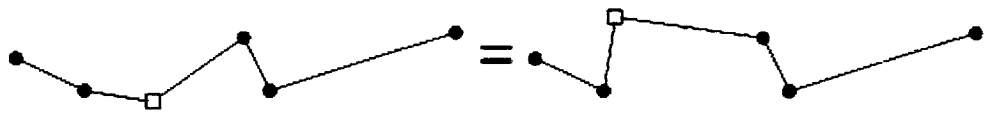


Figure 6.4: *General case for the Perturb Transitional operation*

6.4.2.2. Insert transitional

This operator inserts a randomly generated transitional node in a path. As for the *perturb transitional* operator, the inserted waypoint must correspond to a free cell. The new node is inserted in between the two closest waypoints of the path. The probability of applying this operator increases for path's segments in the proximity of obstacles.

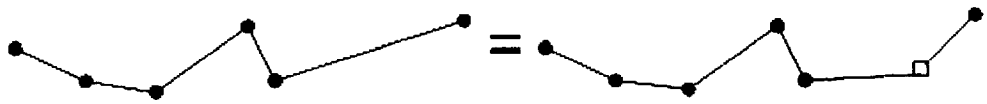


Figure 6.5: *General case for the Insert Transitional operation*

6.4.2.3. Delete transitional

This operator deletes a randomly selected transitional waypoint in a path. When the path is unfeasible, the waypoint to be deleted is selected randomly with a probability of one. For feasible paths, a node is selected for deletion with a probability based on the following heuristics:

- Sharpness of turn: the probability increases as the angle between the two segments joining at the selected node decreases.
- Proximity of obstacles: the probability increases for increasing distance from obstacles (i.e. more probable in large open spaces of the map).

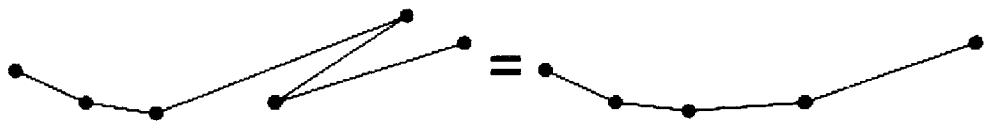


Figure 6.6: *General case for the Delete Transitional operation*

6.4.2.4. Swap genes

A path is randomly selected. As for the crossover operator, this path is fragmented into genes. A first gene is selected based on a probability proportional to the length of the segment joining the gene's objective waypoint and the waypoint following it. The second gene is selected randomly. Once identified, the two genes are swapped. This operator allows the optimization of the objectives order. In the special case where time constraints are associated with one or both objective waypoints being swapped, the relative order required to satisfy them is maintained. Thus, objective waypoints of feasible path which fall in that category remain unaltered by the swap operator, the constraints on their relative order being already satisfied. In the case of unfeasible paths, the swap operator performs more than a simple switch. It goes through the whole path reordering all objectives (and thus, all genes), such that all time constraints are ultimately satisfied. This operation will speed up the process of mutating an unfeasible path into a feasible one. Thus, the probability of applying this operator on newly generated paths with time

constraints is set to a 1, implying that it is automatically performed after the crossover operation if the conditions apply.

6.4.2.5. Smooth turn

This operator measures the sharpness of a turn by calculating the angle between two adjacent segments joining at a waypoint. If the angle falls below a certain limit, the turn is smoothed out by inserting a new transitional waypoint on the perpendicular of the second segment passing by its midpoint, as represented in Figure 6.7. The waypoint's coordinates is adjusted such that the angle between the first and the modified segments is equal to a predefined value above the sharpness threshold. The probability of applying this operator is proportional to the sharpness of the turn; i.e. inversely proportional to the angle of two adjacent segments.

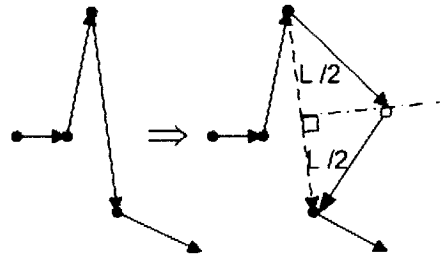


Figure 6.7: *General case for the smooth turn operation*

6.5. Selection mechanism

The selection mechanism is called the *Roulette wheel selection.* This probabilistic approach consists of assigning to each chromosome a probability directly proportional to their relative fitness with respect to the rest of the population.

Before describing the implementation of the selection mechanism, it is important to mention that the fitness function used in this work is designed to be minimized in the optimization process. Thus, more optimal solutions (i.e. solution with higher fitness) have lower "fitness scores". Moreover, chromosomes which have been evaluated as unfeasible are assigned a negative fitness score. As for feasible ones, smaller absolute value of the fitness score indicates better solutions. Due to these specificities, Equation 6. 1 cannot be applied directly. In order to compute the selection probability for a given chromosome, an intermediary probability (p_i') is calculated:

feasible chromosomes :

$$p_i' = \frac{\min(\textit{fitness of all individuals})}{\textit{fitness of individual } (i)}$$

unfeasible chromosomes :

$$p_i' = \frac{\min(\textit{fitness of all individuals})}{|\textit{fitness of individual } (i)| + \max(\textit{fitness of all feasible individuals})} \qquad 6.2$$

The sum of the all chromosome's (p_i') value is taken over the entire population. Equation (6-1) can then be applied to compute the selection probability of each chromosome:

$$p_i = \frac{p_i'}{\sum_{j=1}^{N} p_j'} \qquad 6.3$$

A conceptual roulette is then built by concatenating intervals based on these probabilities. The following table provides an example considering four chromosomes to demonstrate the probability distribution resulting from this selection process.

Table 6.4: *Example of a Roulette wheel distribution*

Chromosome #	Fitness score	Probability	Interval
1	56	0.65	0 ~ 0.65
2	128	0.20	0.65 ~ 0.85
3	144	0.10	0.85 ~ 0.95
4	152	0.05	0.95 ~ 1.0

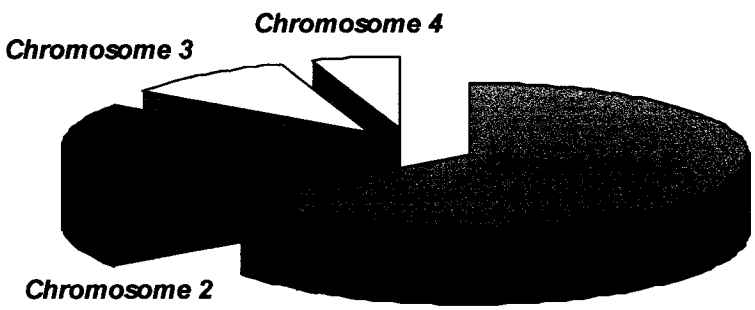


Figure 6.8: *Probabilities distribution for the Roulette wheel selection*

The last step simply consist of generating a number of random values between 0 and 1 equal to the number of chromosomes to be kept for the next generation. Because of the characteristic distribution of this selection mechanism, the most probable candidates are always the ones with the lowest fitness score as illustrated in Figure 6.8.

6.6. Fitness function

In contrast with other optimization algorithms such as the A* algorithm mentioned in the literature review, the values generated by GA's fitness functions do not need to be bounded in any way. Thus, the costs associated to the different criteria do not need to be proportional to the physical aspects they represent. In practice, scaling problems may occur if the relative weights and cost values associated with each criterion are out of proportion. All weights and criteria are thus normalized for that reason.

The amount of information provided by chromosomes is too limited to assess their fitness score directly. A trajectory must be computed first for each chromosome. Their relative fitness score may then be assessed by comparing them with each other. In order to speed up the fitness evaluation process, feasible and unfeasible chromosomes are treated differently. Thus, trajectories' feasibility must be checked before their fitness is assessed. In order to be deemed feasible, a chromosome (and its associated trajectory) must satisfy the following conditions:

1. No obstacle cells are crossed.
2. The fuel required to complete the trajectory is less than the vehicle's initial capacity.
3. The order of the objectives satisfies the time constraints associated with them.
4. The trajectory is dynamically feasible.

For feasible trajectories, the system assesses the level of optimization, while for unfeasible ones, it assess the level of constraint violations. In order to ensure that feasible trajectories always have a higher fitness than unfeasible ones, the highest fitness score of all feasible trajectories is added to the scores of all unfeasible trajectories just before the selection process is started. Different types of optimization can be achieved by tuning the weights of each criterion evaluated in the fitness function. The following section gives a detailed description of the GA's fitness function.

6.6.1. General form of the Fitness function

The general form of the fitness function (*FF*) is as follows:

$$FF(T_i) = \frac{\sum_{j=1}^{M} w_j \sigma_j}{\sum_{j=1}^{M} w_j} \qquad 6.4$$

with Ti, the trajectory of the ith chromosome of a population

σ_j, the cost computed for criteria j

w_j, the weight associated to criteria j (as set by the user)

M, the total number of criteria based on the feasibility of trajectory i

The weights w_j are the parameters that can be tuned based on mission requirements. Their relative value defines which criteria should predominate in the optimization process. The criteria to which these weights are associated are presented in the following sections.

6.6.2. Criteria for feasible chromosomes

- *Time ratio*

The time ratio is defined by the total time required by a given trajectory, divided by the longest time found among all trajectories of the population. The time required by a trajectory is given by the length of each trajectory's segment, divided by the associated velocity, and summed over all segments. Each trajectory's segment has a near constant velocity that validates the approximation used here.

$$\sigma_{tr} = \frac{\left(t_{f_i} - t_{0i}\right)}{\max_{k=1}^{N}\left(t_{f_k} - t_{0k}\right)} \qquad 6.5$$

with N, total number of chromosomes of a population

t_{fi}, the final time of trajectory i

t_{0i}, the initial time of trajectory i

- *Range ratio*

This ratio is equal to the total distance traveled for a given trajectory, divided by the longest one found among all other trajectories of the population. The total distance of a trajectory is calculated by performing a line integral on the parametric equations of each

section for every segment, and summing over all segments. The parametric equations correspond to the equations describing the sections discussed in section 5.1.2.

$$\sigma_{Rr} = \frac{(\Delta L_i)}{\max_{k=1}^{N}(\Delta L_k)} \qquad 6.6$$

with ΔL_i, the total distance traveled for trajectory i

- *Altitude deviation*

The altitude deviation is equal to the normalized standard deviation between the desired altitude (specified by the ASO strategy) and the altitude of each trajectory's waypoints (which is greater than the number of waypoints of the associated path):

$$\sigma_h = \frac{1}{N_T h_d}\left[\sum_{k=1}^{N_T}(W_k(z) - h_d)^2\right]^{1/2} \qquad 6.7$$

with N_T, the total number of trajectory's waypoints

$W_k(z)$, the z coordinate of the k^{th} waypoint

h_d, the desired altitude specified by the ASO parameter

- *Speed deviation*

The speed deviation is equal to the normalized standard deviation between the desired speed and the speed of each trajectory's segments.

$$\sigma_V = \frac{1}{T_S V_d}\left[\sum_{k=1}^{T_S}(S_k(V) - V_d)^2\right]^{1/2} \qquad 6.8$$

with T_S, the total number of trajectory's segments

$S_k(V)$, the velocity associated to segment k

V_d, the desired velocity based on the ASO parameter

- *Obstacle distance deviation*

Octree regions characterized by small cells indicate a higher level of discretization that happens only near obstacles. Consequently, the obstacle deviation is calculated by counting the number of cells traversed by the trajectory, which have a cell size less than the specified threshold. The normalized standard deviation is then computed according to the ASO parameter and the actual size of these smaller cells:

$$\sigma_D = \frac{1}{N_{SC} D_d}\left[\sum_{k=1}^{N_{SC}} (D_d - C_k(s))^2\right]^{1/2} \quad \text{for} \quad D_d > C_k(s) \qquad 6.9$$

with N_{SC}, the total number of cells of smaller size

$C_k(s)$, the side length of the cell k belonging to the N_{SC} list

D_d, the desired obstacle distance based on the ASO parameter

- *Time of Arrival deviation*

For objectives with time constraints, the deviation on the time of arrival (ToA) is calculated as the normalized standard deviation between the desired ToA (specified in the objective's constraint) and the estimated time of arrival of the trajectory at that waypoint:

$$\sigma_{tc} = \frac{1}{N_{Otc}}\left[\sum_{k=1}^{N_{Otc}} \left(\frac{ETA_k - O_k(t_c)}{O_k(t_c)}\right)^2\right]^{1/2} \quad \text{for} \quad ETA_k > O_k(t_c) \qquad 6.10$$

with N_{Otc}, the total number of objectives with type (4) time constraints

ETA_k the trajectory's estimated time of arrival at objective k

$O_k(t_c)$, the time of arrival required by the objective's time constraint

Note: objective with no time constraints have their ToA parameter set to zero.

- *Wind effects*

The corrections required to compensate for wind conditions come under the authority of the local planner and/or some other low-level controller. However, the effect of having the wind with or against the motion of the UAV will affect the extent at which the vehicle will have to compensate in order to follow the plan established by the global planner. Rather than trying to predict the deviation of the planned trajectory versus a predicted true trajectory which will be based on potentially erroneous forecast of the wind conditions, this criterion evaluates the extent at which the wind works against the UAV along the planned trajectory. This is achieved by evaluating the difference between the angle of the aircraft velocity vector and the wind velocity vector. The angle is calculated at all waypoints of the path. Since the optimum wind angle is zero (i.e. speed and wind vectors are in the same direction), the criterion's equation can be expressed as:

$$\sigma_W = \frac{1}{T_S N_v}\left[\sum_{j=1}^{T_S}\sum_{k=1}^{2}\left(\overrightarrow{S_{jk}(V)} \hat{\bullet} \overrightarrow{P_{jk}(W_V)}\right)^2\right]^{1/2} \qquad 6.11$$

with T_S, the total number of trajectory's segments

k, the index of the first and last waypoints of the segment j.

$\overrightarrow{S_{jk}(V)}$, the velocity vector of segment j for parameter $v = k$

$\overrightarrow{P_{jk}(W_V)}$, the wind velocity vector at the waypoint k.

$\hat{\bullet}$ an operator calculating the angle from a reference vector V_r to another vector V_f (measured counter clockwise), given in a range of [+2π, -2π] radians. Defining that angle as θ, this operator is computed as follows:

$$\begin{aligned}
\sin(\theta) &= \frac{V_{f(y)} \cdot V_{r(x)} - V_{f(x)} \cdot V_{r(y)}}{\left(V_{r(x)}^{\ 2} + V_{r(y)}^{\ 2}\right)} \\
\cos(\theta) &= \frac{V_{f(y)} \cdot V_{r(y)} - V_{f(x)} \cdot V_{r(x)}}{\left(V_{r(x)}^{\ 2} + V_{r(y)}^{\ 2}\right)} \\
\theta &= \tan^{-1}\left(\sin(\theta)/\cos(\theta)\right)
\end{aligned} \qquad 6.12$$

6.6.3. Criteria for unfeasible chromosomes

The fitness function for unfeasible trajectories uses the following criteria in the same way as they are evaluated for feasible chromosomes:

- *Time ratio*
- *Altitude deviation*
- *Time of Arrival deviation*
- *Obstacle distance violation*

The last criterion computes the total number of obstacle cells traversed by the trajectory and divides it by the maximum number of cells crossed by a single trajectory among the entire population:

$$\sigma_{OC} = \frac{N_{C(i)}}{\max_{j=1}^{N}(N_{C(j)})} \qquad 6.13$$

with $N_{C(i)}$, the total number of obstacle cells traversed by the i^{th} trajectory

N, the total number of chromosomes of the population

All other criteria for unfeasible trajectories are set to zero.

6.7. Termination conditions

The default termination condition implemented in the current version of the system is the maximum number of generations allowed during the evolution process. The best solution of the final generation is then found by selecting the fittest chromosome (i.e. the one with the lowest positive fitness score). If no feasible path has been found by the GA, the best unfeasible one is selected (i.e. the one with the lowest absolute fitness score).

Due to the particularities of the current application, the GA may have varying amount of time to find a valid solution. A second stopping condition can easily be implemented at by using a virtual timer, which would check the time elapsed after each generation. The evolution loop could then be forced to stop when the time allocated to the process has elapsed.

Chapter 7: System performances & limitations

While the previous sections presented the theory of each individual module, this section describes their performance and limitations. Sections 7.1. to 7.3. present the performances of the each modules individually, while Section 7.4. presents it for the overall system.

7.1. Octree performance

Since the system described in this work combines standard mapping and octree mapping techniques, the following discussion presents their respective performance separately. The term standard map will be used in this section to describe an evenly divided 3D map with a linear data structure, as opposed to the tree data structure used for octrees. In order to compare their relative efficiency, it is easier to consider them as databases with varying level of overhead, i.e. varying amounts of memory used only for maintaining data integrity and accessibility. Thus, the best way to assess their performance is to evaluate the level of memory used under a common set of conditions. Based on the specific implementation of the hybrid octree developed for this work, the amount of memory used depends on the following factors:

- The ground area to be covered (in meter squared).
- The minimum altitude of the region and the UAV's ceiling (i.e. maximum altitude).
- The obstacle density of the region.
- The number and types (i.e. vector fields or density functions) of dynamic layers.
- The number of descriptors defined for each basic and dynamic layer.
- The maximum depth of the octree, which sets the maximum resolution of the map.
- The number of standard map (SM cells) defined by the user.
- The number of sub-cells defined for each SM cell.
- The maximum power of the polynomials used for descriptors.
- The number of overlapping descriptors.

For the purpose of comparing octree and standard map separately, the number of SM cells in the octree is set to zero, while the number of layers and descriptors must be set equal in order to have the same amount of information stored in both types of maps.

Since layers and descriptors are implemented independently of the SM or octree data structure, these factors do not contribute to the performance of one against the other. Thus, the number of dynamic layer has been set to zero while the number of descriptors has been arbitrarily set to three for the wind layer and the TPD layers, setting the overlap between descriptors to zero. Table 7.1 lists all relevant conditions used in this analysis to compare octree and standard maps.

The variables considered for this performance analysis were the ground area and the obstacle density of the region. Since only the ground area impacts memory usage, the region to be mapped is arbitrarily chosen as a square region with a variable side length. In this way, only one variable needs to be considered. The obstacle density was modeled as a constant representing the percentage of children that must be further subdivided after each iteration of the octree creation process described in section 4.1. A high obstacle density region implies that the octree spreads through many parallel branches, while a low one result in a tree with fewer branches.

Table 7.1: *Set of common initial conditions for the comparison of memory usage between octrees and Standard Maps*

Initial Conditions	
Factor	**Value**
Altitude range	[0 - 5000] m
Number of Dynamic layers	0
Number of descriptors per layer	3
Maximum resolution of the octree	300 m
Standard map resolution	300 m
Maximum order of polynomials	6
Number of overlapping descriptors	0

Figure 7.1 shows the results for the conditions stated in Table 7.1. It can be seen from this figure that the major factor influencing the performance of the octree is the density of obstacles. Since the number of cells in a standard map will not change with the number of obstacles, memory usage for SM remains unchanged with respect to that parameter. Using the SM as the reference curve, the octree shows equal performance for an obstacle

density of approximately 55 %. Its performance degrades exponentially above that value and improves at the same rate below that value. The relation between map dimension and memory usage is almost linear for octrees when considering an obstacle density of 50 % or lower. Significant reduction in memory usage can be seen for that range. A standard map of 200 x 200 km requires 1630 MB while an octree with an obstacle density of 40% requires less than 45 MB. This value drops below 5 MB if the obstacle density is less or equal to 30 %. These results are of special interest to UAV applications since the topography of typical 3D maps considered for such applications rarely exceeds an obstacle density of 40 %.

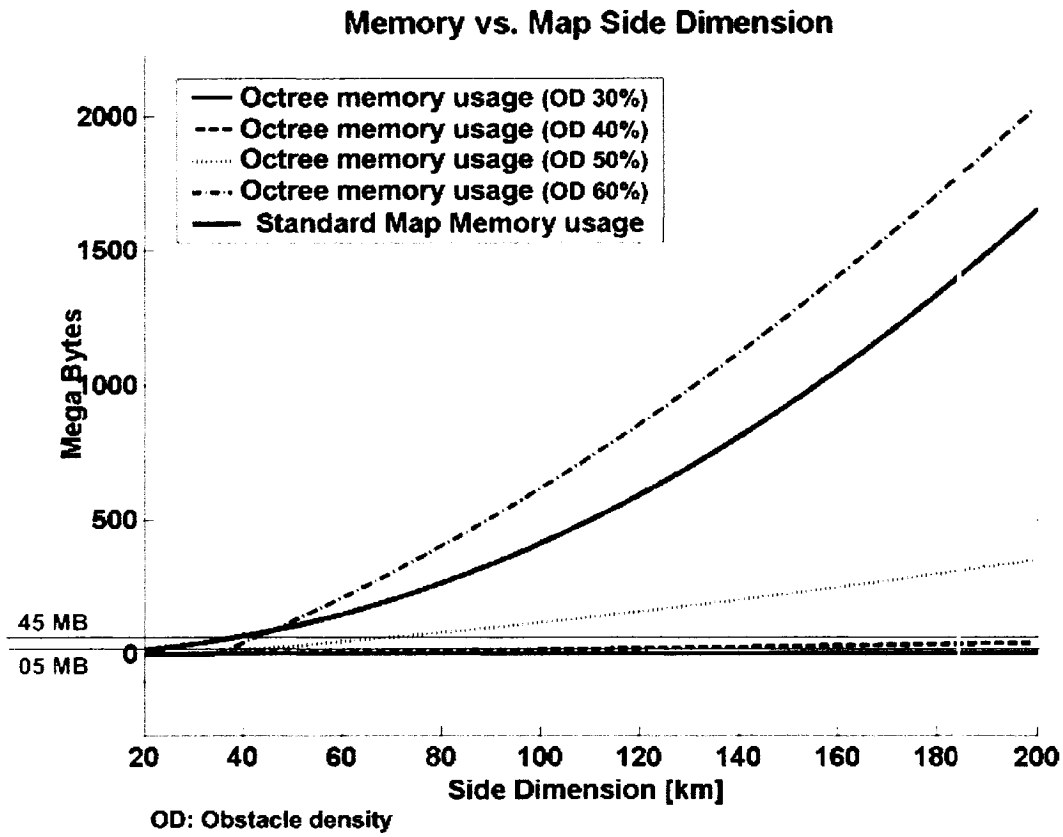


Figure 7.1: *Relative performance of octrees and Standard Maps in terms of memory usage and map dimensions*

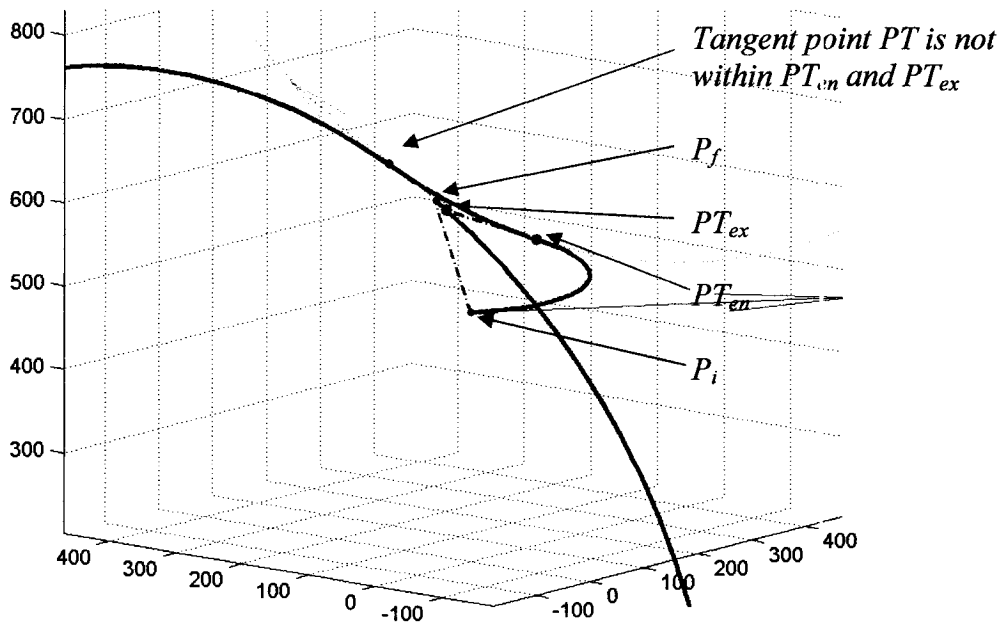
The magnitude of memory usage shown in Figure 7.1 could seem excessive at first sight. However, it must be recalled that topography is not the only information represented in the maps compared above. The weather, including wind, temperature, pressure, and air density are also parts of them. Thus, these maps can be considered to have four dimensions (excluding the extra one for each dynamic layer) in addition to the

standard three dimensions required to describe the spatial distribution of obstacles. This inevitably leads to much higher memory usage when comparing it to the standard 3D maps found in many studies in the literature.

As a final point, it must be observed that the increased efficiency of SM over octrees for high obstacle densities can be overcome by the hybrid octree proposed in this work. In order to achieve this, the user needs to specify a relatively low maximum depth for the octree, and to define specific areas of interest on the map to be converted into SM cells with the desired maximum resolution.

7.2. Performance of the trajectory generation module

The overall processing time of the TG mainly depends on the total number of segments of the path given as input (i.e. the total number of pairs of waypoints). The processing time for a given segment depends on the computer's hardware performance as well as the maximum number of steps allowed for the two optimization processes outlined in Figure 5.18, and the search algorithm that these processes use. On average, the number of steps required is 7 for the first optimization process, 3 for the second optimization process, and 8 for the internal loop of this last process. The worst case gives a total maximum of 34 steps per segment.



(a)

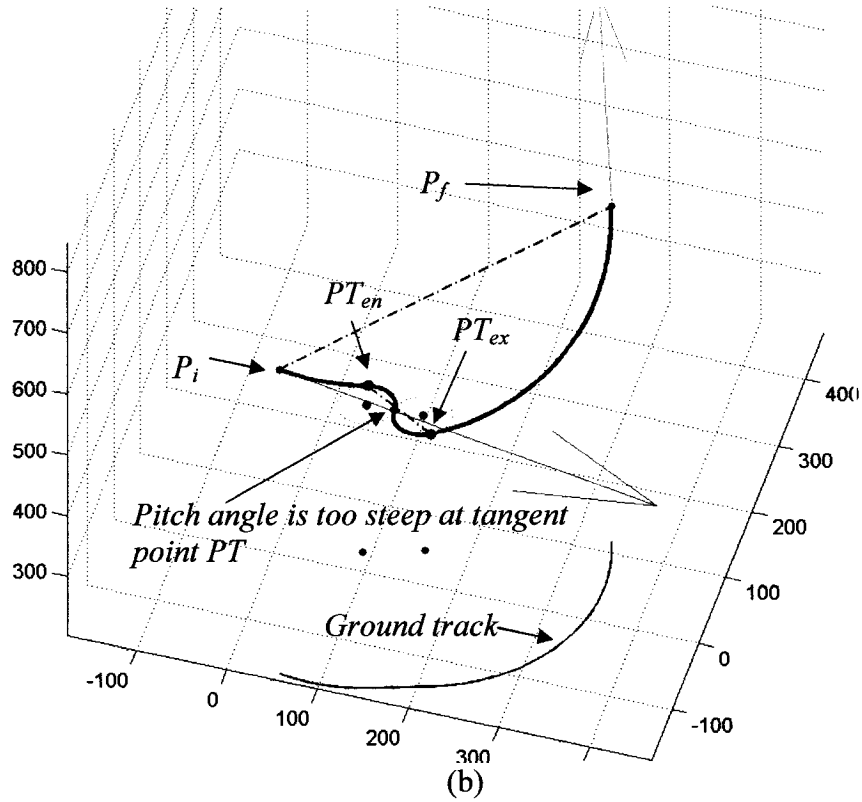


Figure 7.2: *Examples of unfeasible segments*

Fewer iterations in the *optimization process I* results in an over or under estimate of the trajectory's dynamic parameters. In both cases, the computational load of the *optimization process II* is increased and the level of optimization of the final trajectory is reduced. Ultimately, reducing the number of steps may lead to a situation where the overall process may become unable to find a dynamically feasible solution. Figure 7.2 (a) and (b) give examples of unfeasible trajectory's segments.

The average processing time for a single segment was found to be $0.030 \cdot 10^{-2}$ sec, using an Intel based PC of 2.4 GHz, with a RAM capacity limited to 16 MB for this application alone (excluding all other processes running in parallel). Values ranged from $0.025 \cdot 10^{-2}$ to $0.052 \cdot 10^{-2}$ seconds per segment for all trials. These values were measured by averaging the amount of time required to generate the trajectories for an entire population of 40 chromosomes (i.e. paths). The time overhead required to convert paths from the genetic algorithm to the TG format was not included in these measurements. The contribution of this conversion process is limited to 4 ± 2% of the total processing time for one segment.

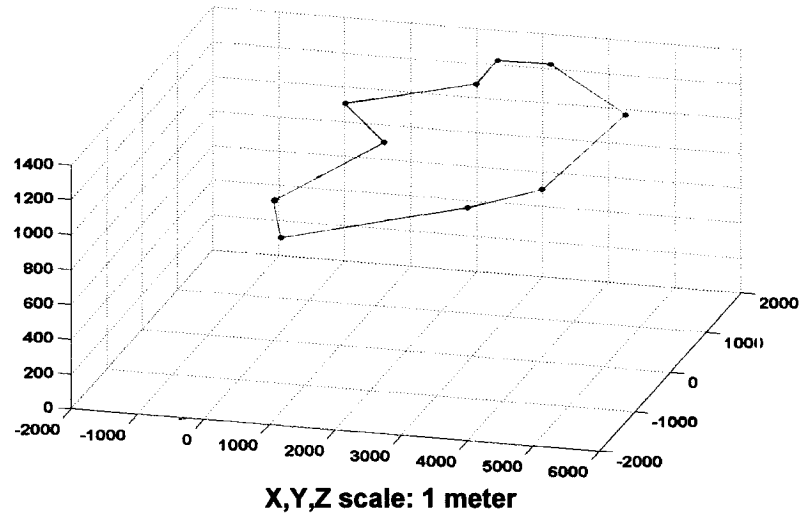


Figure 7.3: *Test path configuration*

The task of the TG becomes more difficult when waypoints are very close to each other. Turns in the vertical and the horizontal plane end up with small R_H and R_V radii, which in turn increases the load factors n_H and n_V required for dynamic feasibility. Balance between loads in the horizontal and vertical planes becomes harder to achieve, requiring more iterations before a solution can be found. In order to test the robustness of the TG algorithm, a test path was set by defining the distance, the pitch and the yaw angles between each pair of waypoints. While the angles remained fixed, the distance was varied, effectively modifying the scale of the path. Figure 7.3 shows the test path configuration. The trajectory computed by the TG for a path set at a large scale is shown in Figure 7.4.

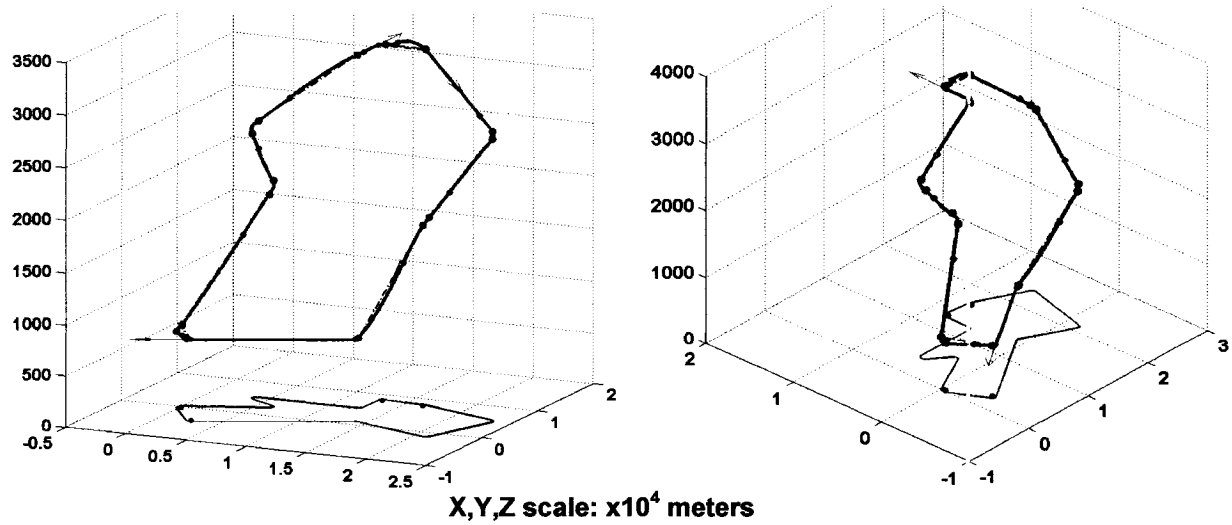


Figure 7.4: *Trajectory for test path at a large scale*

All segments of the path shown in Figure 7.4 are relatively straight, indicating that all vertical turns have a very large radius R_V and a low load factor n_V. Thus, turns in the horizontal plane can use higher load factors n_H, resulting in sharper turns. The high level of curvature at the end of each segment is confirms it. The overall trajectory is properly optimized, each segment linking the waypoints of the path without detours or loop. The load factor is minimized for the middle portion of each segment which maximizes the aircraft velocity. Turns in the horizontal plane are sharp, limiting the overshoot with respect to the straight path shown in Figure 7.3.

The trajectory shown under two different angles in Figure 7.5 was obtained by setting the test path at the smallest scale. The resulting trajectory remained feasible. Since some waypoints are too close to each other, the segment has to make a loop in order to be dynamically feasible, overshooting the position of the segment's ending waypoint before going back to it at the specified VoA. The only two linear segments existing for the small scale trajectory result from the fact that two segments have their initial and final vectors of approach collinear. All other segments show a relatively high level of curvature in the vertical and horizontal planes, indicating that the aircraft would constantly be under high loads (i.e. the aircraft would be continuously manoeuvring). Despite the high constraints inherent to the small scale test path, the TG is still capable of finding a dynamically feasible trajectory for it, demonstrating its robustness.

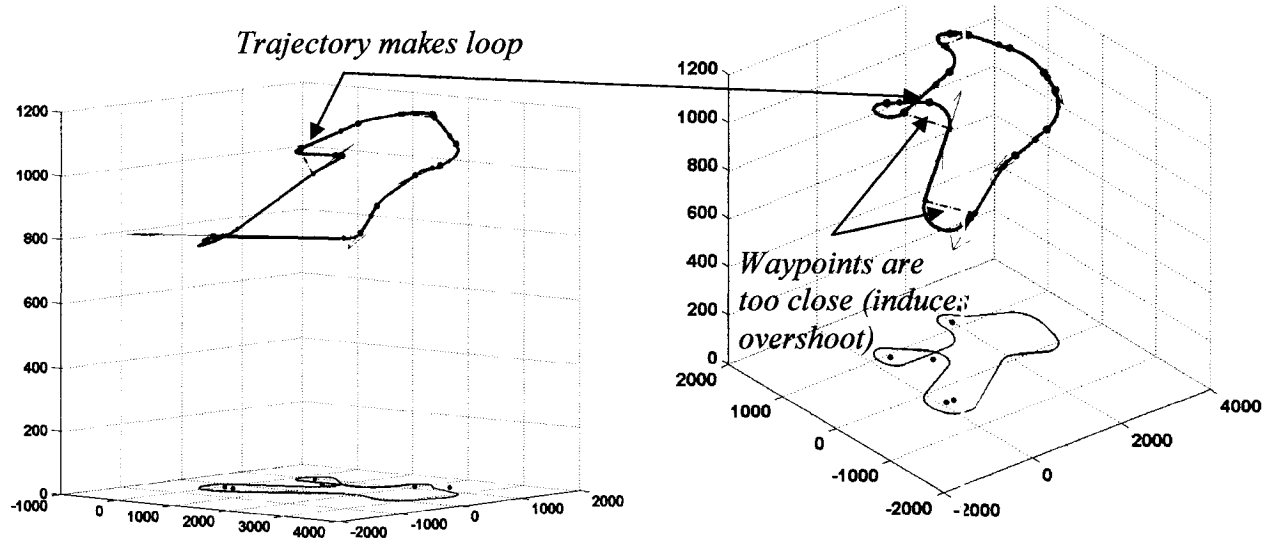


Figure 7.5: *Trajectory for test path at the smallest scale*

7.3. GA performances

In order to focus on the GA capabilities in dealing with multiple objectives and multiple goals, all tests are based on a standard mission profile with no time constraints, and no pre-specified vector of approaches (although VoA's are automatically assigned to each objectives by the TG when the initial population is created). The ASO strategies are set to the values (1; 3; 1) respectively, implying that the desired altitude is set to the minimum (A=1); the speed is set to the endurance velocity (i.e. minimum drag velocity, S=4); and the minimum obstacle distance is equal to the largest aircraft dimension (i.e. wing span time a safety factor, O=1).

7.3.1. Missions with 1 Objective

The first set of tests is performed under a mission profile of a single objective and no goals. Figure 7.6 gives a good representation of the typical performances of the GA for this type of mission.

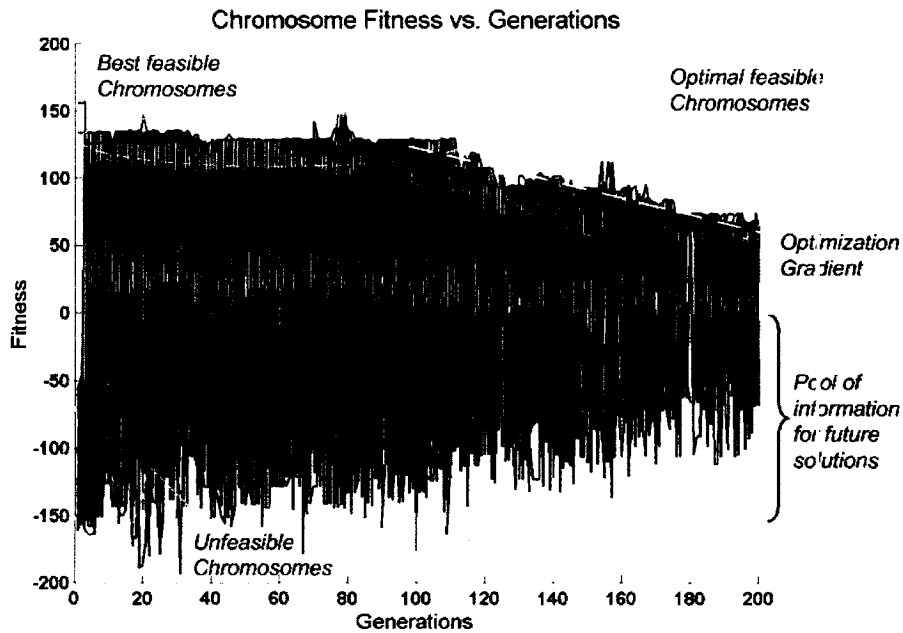


Figure 7.6: *Evolution of population for a mission with 1 Objective, 0 Goal*

The first 10 generations are characterized by a population mainly composed of unfeasible chromosomes. A shift toward feasible solutions starts between 20 and 25 generations, showing the effect of the mutation operators over a sub-part of the population which starts to converge toward some minima in the search space. As the

entire population gradually convergences toward the most optimal chromosomes of the feasible region, the number of feasible chromosomes increases while the fitness score of unfeasible chromosomes tend to increase toward zero (i.e. tend to shift toward positive fitness scores). A first plateau is usually encountered near the 60th generation, with an average duration of 25 generations as shown in Figure 7.7. Past this plateau, optimization resumes crossing smaller plateaus.

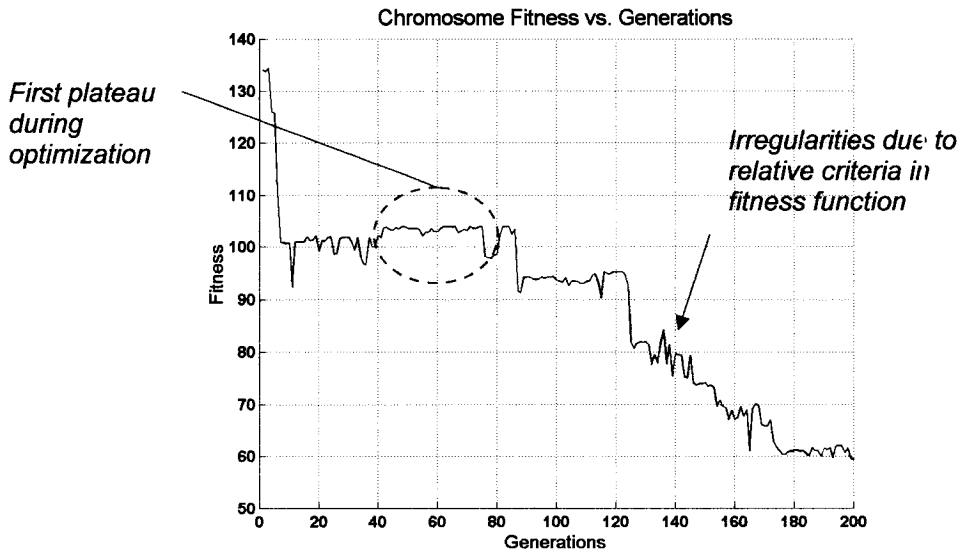


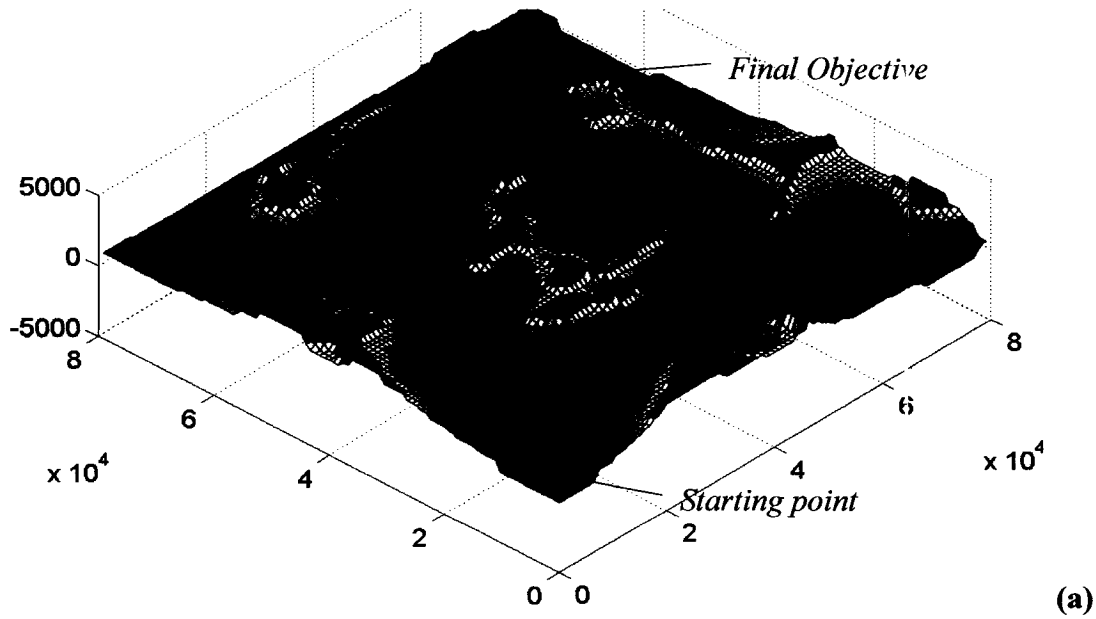
Figure 7.7: *Evolution of the best chromosomes for a mission with 1 Objective, 0 Goal*

The unfeasible part of the population remains relatively high during the entire evolution process. In average, 50 to 70 % of the chromosomes of a population remain unfeasible during the entire optimization process. This experimental observation is well illustrated in Figure 7.6 with the high density of chromosomes remaining in the negative part of the graph. The fact that a large number of chromosomes remain unfeasible means that most chromosomes have not converged toward a specific optimum and thus, a large variety of different candidate solutions are still present in the population. This implies that the pool of information kept in the form of genes and specific transitional nodes remains diversified enough to avoid having the GA getting trapped in local optimum.

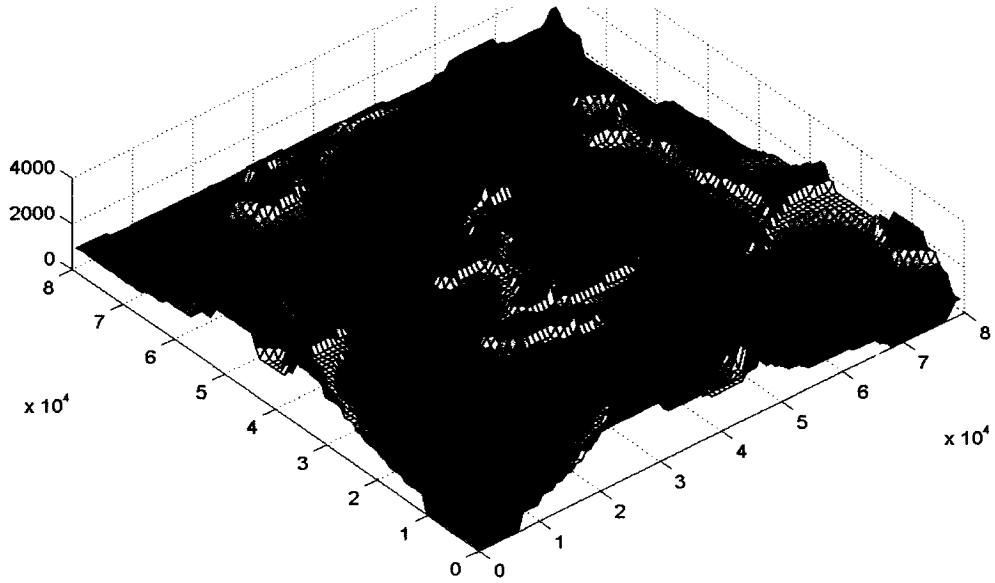
The irregularities in the trace of the best chromosome's fitness score may seem abnormal when considering that elitism in the GA process should ensure that the next generation is at least as optimal as the previous one (the best individual being kept without alteration in the next generation). In fact, the fitness scores for individual

chromosome are based on several criteria which are computed as ratios of the chromosome's performance over the worst performance of all chromosomes within the same generation. Thus, fitness scores are actually computed as relative values with respect to the population at each generation. Considering that the maximum length d_i of all trajectories of generation i is greater than the one of generation j, the range ratio (i.e. the first fitness function criteria) of an unaltered chromosome would be greater for generation j than for generation i. This situation would result in a fitness score higher for generation j than for generation i. Thus, the best chromosome of a given generation may have a greater fitness score at the next one, resulting in the irregularities shown above.

Despite continuous optimization of the fitness function, the complexity of the resulting trajectory is not necessarily minimized past the 60th generation. Figure 7.8 shows two trajectories obtained for the same mission profile after 200 generations of optimization. In both cases, the fitness score of the best chromosome is continuously decreasing in a manner similar to the one shown in Figure 7.7. While the first test results in a relatively smooth trajectory as shown in Figure 7.8 (a), the second one shows that the remaining 140 generations after the 60th one did not succeed in smoothing out all loops and detours as shown in Figure 7.8 (b).



(a)



(b)

Figure 7.8: *Resulting trajectories after 200 generations for the same mission with 1 Objective, and 0 Goal (a) smooth trajectory, (b) trajectory with loops and unnecessary detours*

7.3.2. Missions with 2 Objectives

The second set of tests is performed under a mission profile with two objectives and no goal. As shown in Figure 7.9, the population's evolution for two objectives is slower than for one objective. The starting population is almost exclusively made of unfeasible chromosomes. Feasible chromosomes start to be generated in a significant number only past the 20th generation. Global convergence of the population toward the best feasible chromosomes does not start until the 35th generation.

The optimization gradient is also much steeper for missions with one objective (Figure 7.6) compared to missions with two (Figure 7.9). As shown in Figure 7.10, minimization of the fitness function is very slow and only small improvements are achieved past the 30th generation.

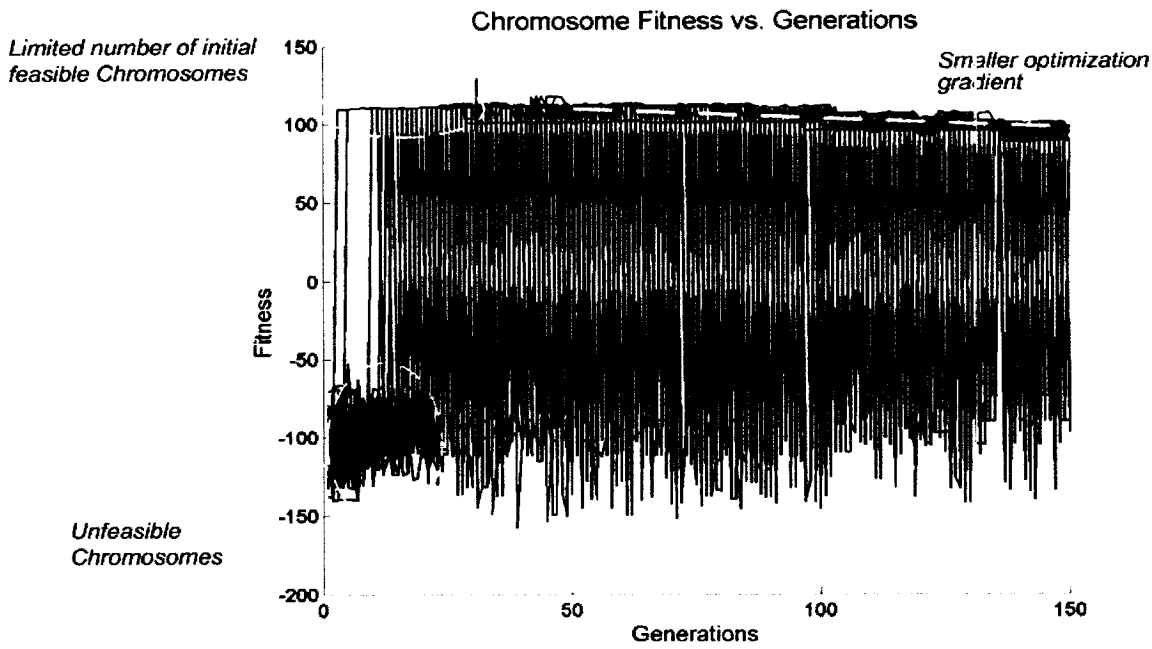


Figure 7.9: *Evolution of population for a mission with 2 Objectives, 0 Goal*

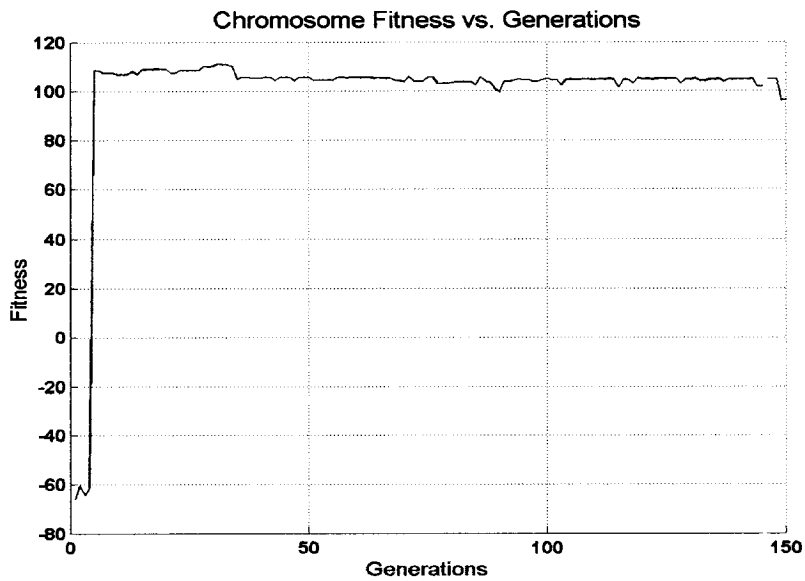


Figure 7.10: *Evolution of the best chromosomes for a mission with 2 Objectives, 0 Goal*

7.3.3. Missions with 1 Objective and 1 Goal

A comparative set of tests is performed with a mission consisting of one objective and one goal. In contrast with 2-Objectives mission, experimental results for 1-Objective 1-Goal missions show an increase in the number of generation required before

establishing a steadily growing sub-population of feasible chromosomes as shown in Figure 7.11. Although a marginal number of feasible chromosomes are usually present after the 10 or 20 first generations, an average number of 90 generations is required to obtain multiple feasible solutions as shown in the example of Figure 7.11.

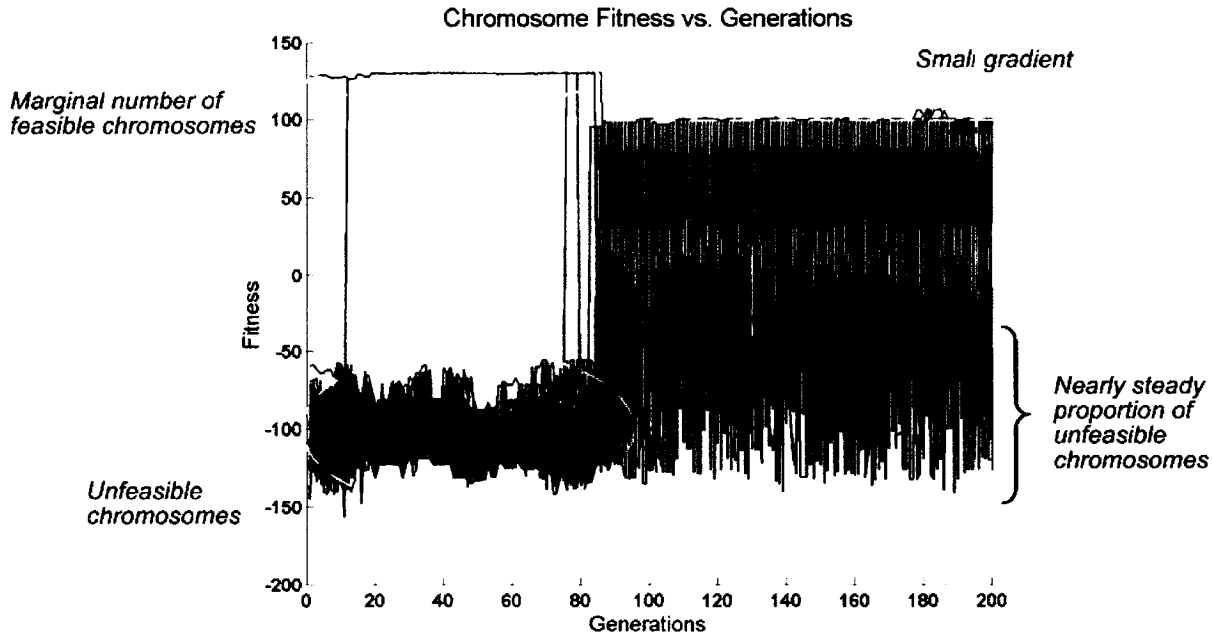


Figure 7.11: *Evolution of population for a mission with 1 Objectives, 1 Goal*

Although Figure 7.12 shows a great and sudden decrease of the fitness score (i.e. indicating that chromosomes are becoming more optimal), the scale of the figure indicate that the level of optimization achieved is actually much more limited than for 1-Objective missions. The average fitness score reached after 200 generations is approximately 60 for 1-Objective missions, and ranges around 90 for 2-Objectives or 1-Objective 1-Goal missions. Although the final fitness scores of 1-Objective 1-Goal missions and 2-Objectives missions are within the same order of magnitude, the total variation of the fitness score over time is usually greater for the first type of mission. Thus, using goals instead of objectives seems to increase the level of complexity of the TAP. However, the average processing time drops from 170 seconds for 2-Objectives missions to 80 seconds for 1-Objective 1-Goal missions. This can be explained by the fact that the system does not need to perform the crossover operation between objectives when there is only one of them (the other ones being converted into goals). Also, using goals instead of objectives actually reduces the size of the search space by specifying the order in which they must

be performed during the mission. These two aspects reduce the computational load on the system, resulting in shorter processing time.

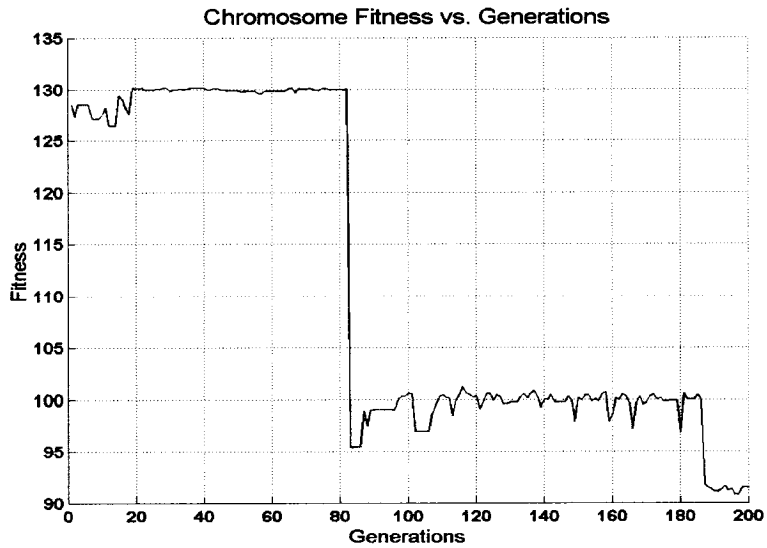


Figure 7.12: *Evolution of the best chromosomes for a mission with 1 Objective, 1 Goal*

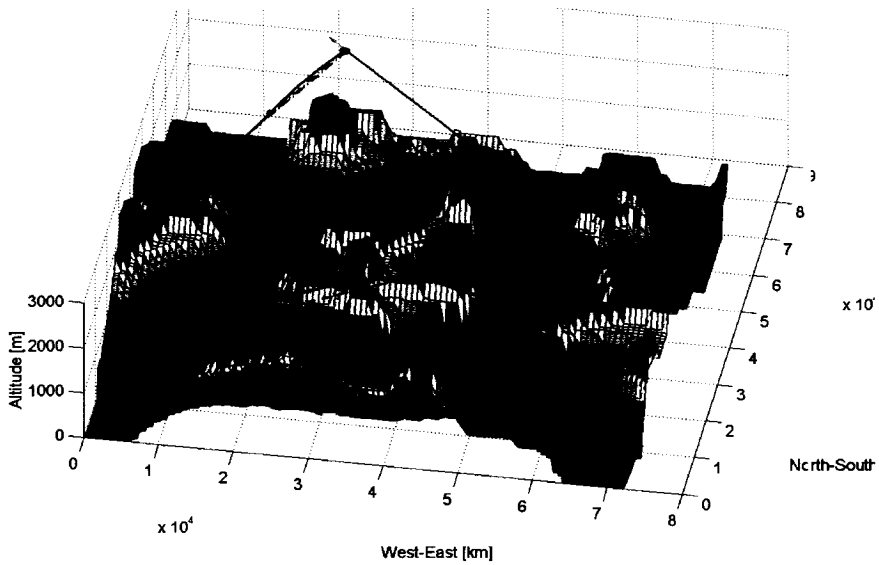


Figure 7.13: *Resulting trajectory for a mission with 1 Objective, 1 Goal*

According to empirical results, using goals instead of objectives simplifies the trajectories' geometry of the end result, as the example of Figure 7.13 shows. Analogously, reducing the number of transitional waypoints in chromosomes has the same effect. Since transitional waypoints are added for every objective but not for goals, using goals instead of objective result in a decrease of the number of transitional node in the chromosome. Thus, the simplification of the trajectory's geometry may be attributed

to the decrease of the number of transitional waypoints rather than the simplification of the objective/goal ordering problem (i.e. goals have predefined order with respect to their corresponding objective).

7.3.4. Missions with 3 Objectives vs. 1 Objective and 2 Goals

Missions with more than 2 objectives become a real difficulty for the GA. Convergence toward feasible results is obtained in only 10 % of the cases for 3 objectives. The probability of finding a feasible solution within 300 generations is slightly higher for 1-Objective 2-Goals mission than for 3-Objectives missions, increasing from 10 to 20 %. Under the current level of development of the system, these two types of mission represent the limit cases for which solutions can be found. Figure 7.14 shows an example of a successful convergence for a mission with 3 objectives while Figure 7.15 shows the results for a mission with as 1 objective and 2 goals

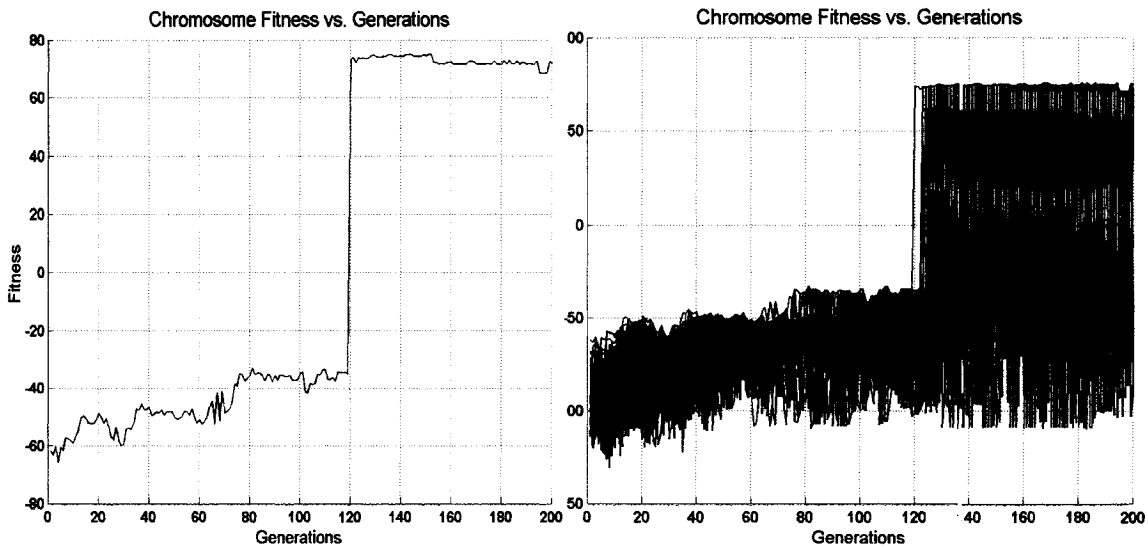


Figure 7.14: *Evolution of the population for a mission with 3 Objectives, 0 Goal*

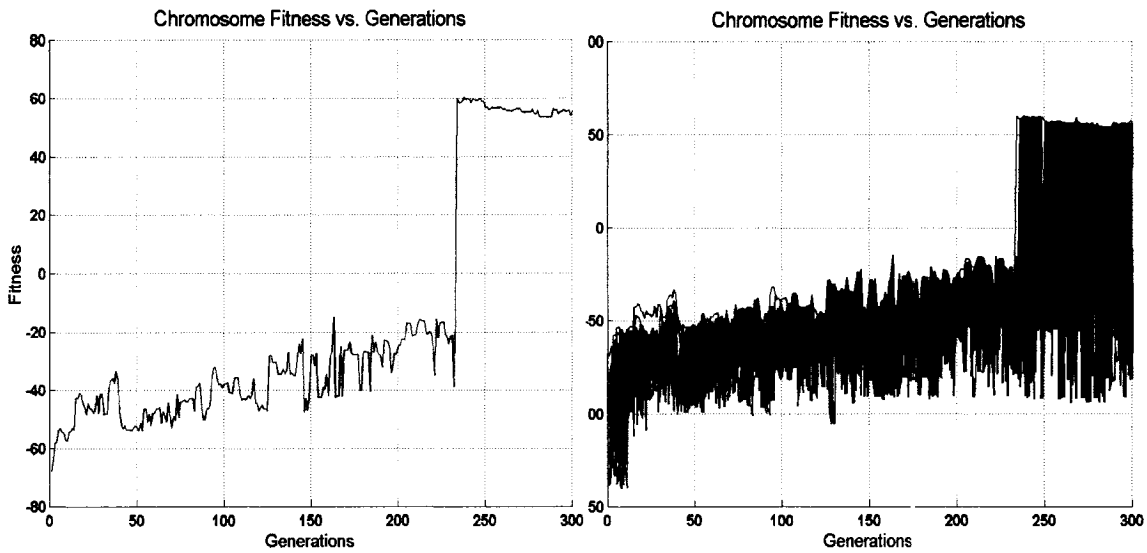


Figure 7.15: *Evolution of the population for a mission with 1 Objectives, 2 Goals*

As it is the case for 2-Objectives missions compared to 1-Objective 1-Goal missions, the GA requires more generations to find feasible chromosomes when goals are present. However, the number of generations required to do so is highly dependent on the initial set of chromosomes. This implies that the random number generator of the computer used to run the system is one of the major factors determining the GA's performance. Moreover, great variations are observed in processing time depending on the initial set of objectives defined in the mission profile. Certain set seems to lead more rapidly to feasible solutions while others never give any. Processing time to obtain feasible solutions for 3-Objectives missions varies from 623 seconds to 385 seconds with an average in the order of 450 seconds. Following the same trend as before, the time required to process 2-Goals missions is much less than for 3-Objectives missions, varying from 128 seconds to 277 seconds with an average in the order of 200 seconds.

7.3.5. Fitness function and trajectory optimization

As mention earlier, improvements in the fitness function is not directly related to the simplification of the actual trajectory when inspecting it visually. To demonstrate this characteristic, Figure 7.16 shows two tests performed for a mission with 2 objectives and no goals. The trace of the best fitness scores during the optimization process is shown for each test with the associated trajectory resulting at the end of the evolution process.

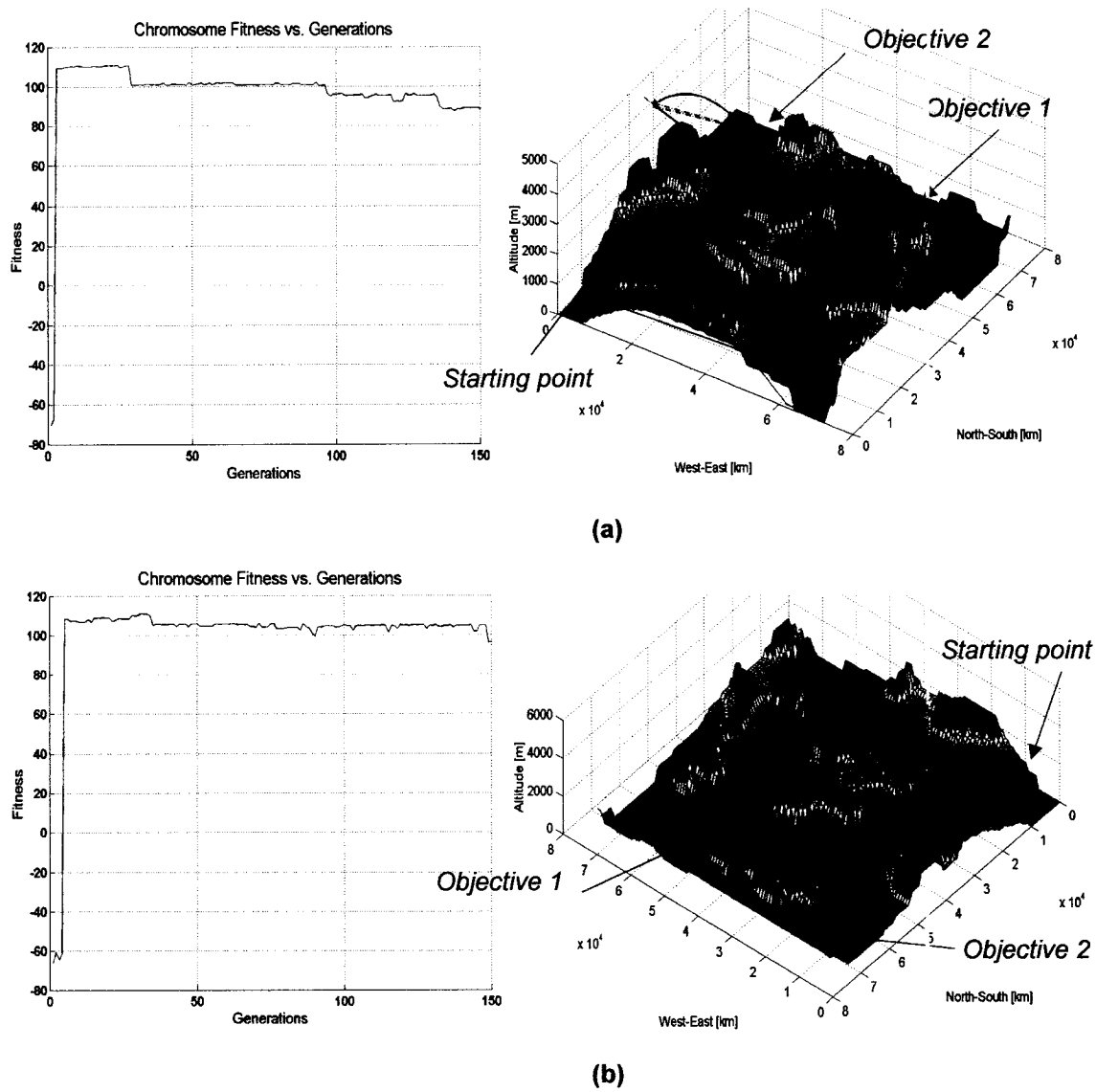


Figure 7.16: *Optimization of the fitness function vs. Final trajectory of best chromosome (for a mission with 2 Objectives, 0 goals)*

Discrepancies between the simplification of the trajectory's geometry (i.e. the main criteria on which visual inspection is based) and the level of optimization of the fitness function may be caused by two factors. The first one is the inherent variability in the generation of new transitional waypoints. This variability affects the mutation of chromosomes after each generation as well as the creation of initial populations. The second factor is the difficulty of the fitness function in optimizing all criteria evenly and efficiently. The number of criteria to be simultaneously taken in account seems to slow down the performance of the genetic algorithm to a point where the final solutions remains sub-optimal or even unfeasible. A minimum of 7 basic criteria must be optimized

in parallel if no dynamic layers are considered. On a mathematical point of view, the gradient search performed by the GA takes place in a multi-dimensional space of at least 7 dimensions. Based on empirical observations, the "obstacle distance" criterion must dominate the function in order to avoid unfeasible trajectory crossing land elements. Its relative weight must be set to at least 37% of all factors in the function. This implies that if all remaining criteria are of equal importance with respect to each other, their relative weight must be limited to 10.5 % each. The level of optimization achievable for these criteria is thus very limited. Future development of the system should focus on generalizing the approach used for the constraint on the desired speed specified in the ASO ratings. Indeed, integrating constraints directly into the calculations of candidate solutions would decrease the number of criteria to be optimized at the GA level and should also diminish the number of generations required to obtain satisfactory solutions.

The second aspect mentioned above concerns the variability of transitional waypoints. The fact that new transitional waypoints are generated completely at random within the limits of the octree map may be a significant factor slowing down the generation of solutions, and preventing trajectories from having simpler geometry (i.e. less detours and loops). Random modification of transitional waypoints during mutation of new chromosomes may cause similar effects. The problem comes from the fact that the generation of transitional waypoint does not take in account the objective waypoint to which they are attached. Narrowing down the search space for new transitional waypoints based on the previous waypoint and the following objective waypoint may reduce the number of detours in the final trajectory. On the other hand, limiting the level of randomness of genetic algorithms tends to decrease their capabilities of finding global optimums. Thus, the alternative of using a filter that would smooth out trajectory based on the GA result may also be a promising approach to enhance the system's performance.

7.3.6. Real time performance

The key component of the GA is the crossover operation. It is the engine that creates new solutions which should theoretically convergence toward an optimum. This genetic operator is implemented using three nested algorithms. The top level algorithm performs crossover at the gene level (i.e. between objectives). The second level performs crossover at the goal level and the last one at the transitional node level. Thus, the computational

load of the crossover operation for this application is essentially three times greater compared to the standard form of the GA. Thus, increasing the number of goals or objectives significantly slows down the performance of the system. As shown in Figure 7.6, Figure 7.9 and Figure 7.14, the amount of time required to get a feasible solution is in the order of magnitude of 5 to 10 generations for 1 objective, 20 to 30 generations for 2 objectives and 120 generations for 3 objectives. The number of generations required to obtain an optimized solution in terms of fitness score and trajectory's geometry tends to be relatively constant for 1-Objective missions and 1-Objective 1-Goal missions. However this number and the processing time associated to more complex missions usually varies by ±50% of the average values shown in Table 7.2. All tests shown in this table were performed on a platform consisting of a Pentium® 4 processor of 2.54 MHz, running on Microsoft Windows XP Pro ®, with a total RAM allocation of 16 MB for this application.

Table 7.2: *Real-time performance of the system*

Mission Type	Average number of Generations		Average processing time (seconds)**
	First feasible solution	*Optimized solution*	
1 Objective	5 – 10	60	15
2 Objectives	20	150	170 *
1 Objective – 1 Goal	90	200	80
3 Objectives	120	200	450 *
1 Objective – 2 Goals	150	300	200 *

* Subject to wide variations depending on the initial set of waypoints generated by the GA

** All test performed on a Pentium® 4 processor of 2.54 MHz, running on Microsoft Windows XP Pro ®, with a total RAM allocation of 16 MB for this application.

Acknowledging the fact that no code optimization was performed on the system, it can be concluded that its actual real-time capabilities are limited to mission with 1 objective and, in simple cases, to missions with 1 objective and 1 goal. The major limiting factor is the TG which requires most of the resources provided to the system. Each generation result in the computation of tens of trajectories which may have tens of segments. The fact that most segments remain the same between an old generation and a

new one may open the opportunity to reduce the amount of computation. This could be achieved by reusing segments of trajectory computed in the old generation, and integrating them directly into the computation of the new chromosomes' trajectory.

7.4. Overall system's performance and limitations

7.4.1. Loops

Due to the random nature of the creation of transitional waypoints, loops may appear in the final trajectories computed by the GA. The number of loops increases with the number of transitional waypoints between objectives. Although reducing the number of transitional waypoints decreases the probabilities of having these loops, it also increases the size of the loops if they actually occur. Also, decreasing transitional waypoints may decrease the capabilities of the algorithm to optimize the trajectory, depending on the complexity of the map.

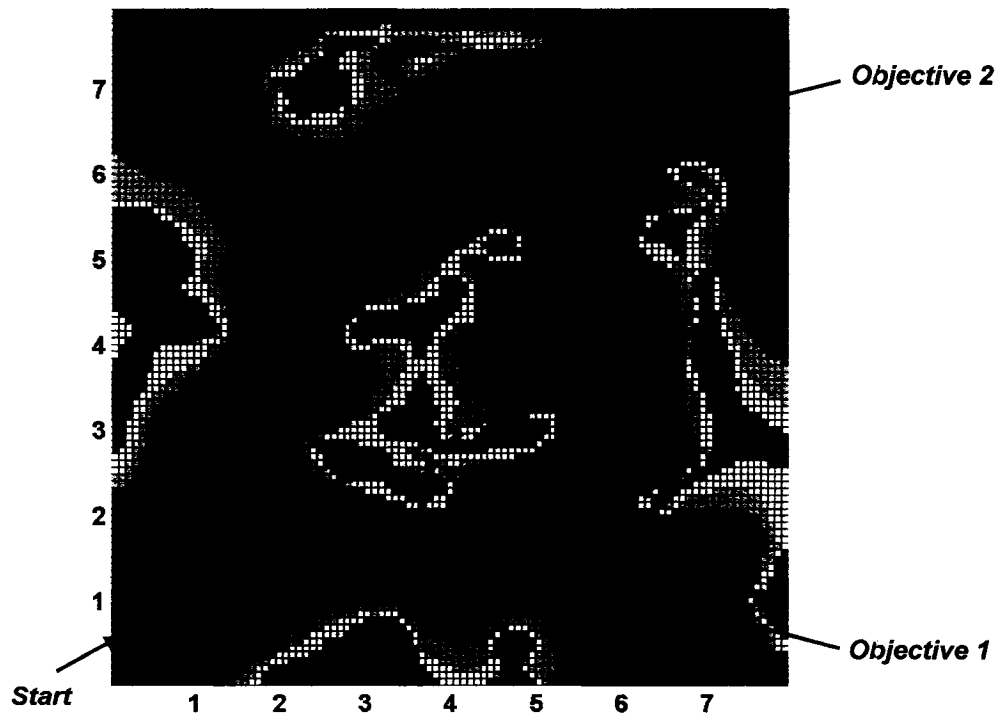


Figure 7.17: *Potential occurrence of loops in trajectories*

7.4.2. Objective and Goal waypoints

All objective and goal waypoints must have their coordinates within the limits of the maps. An error occurs if this constraint is not respected in the mission profile specified by the user. The system has been tested for a maximum number of 3 objectives. Computational time of less than 20 seconds can be obtained for one objective. However, the amount of time required before finding a feasible solution increases rapidly with the number of objectives. Two or three objectives can easily take more than one minute to process, with an average time of 170 seconds for 2 objectives and 450 seconds for 3 objectives (when convergence is actually achieved). Moreover, the optimality of the final result rapidly degrades with the number of objectives (see the number of detours and loops between one, two and three objectives missions in Figure 7.17 and Figure 7.18). Visual inspection of the resulting trajectories shows an increasing number of loops for missions with more than one objective.

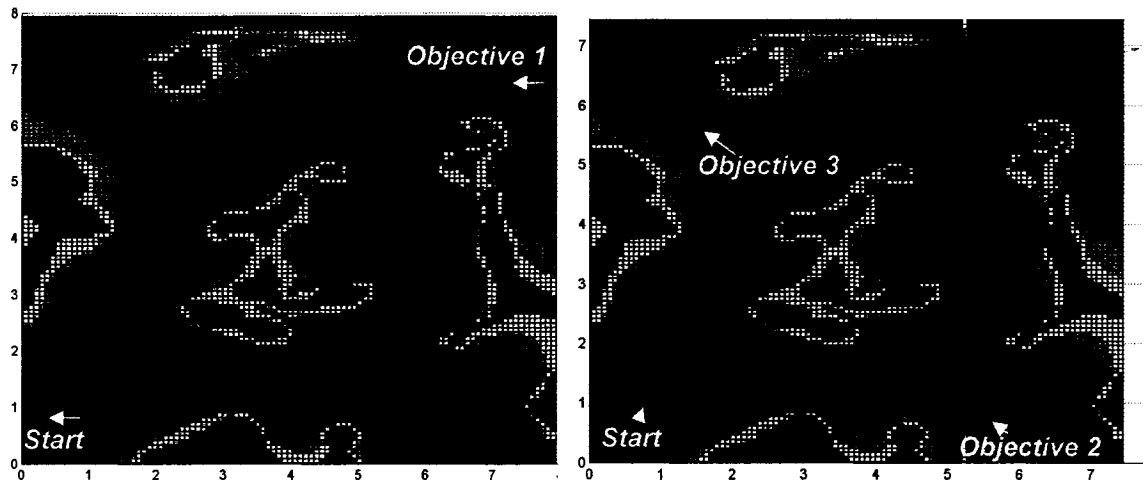


Figure 7.18: *Resulting trajectories for (left) 1 Objective (right) 3 Objectives*

7.4.3. False feasible trajectories

Trajectories computed between two waypoints may go outside the limits of the octree map without crossing any obstacles. In such conditions, the algorithm which takes care of scanning the trajectory to detect obstacles along the way will not detect anything outside the boundaries of the map. This is rarely a problem if the altitude of the UAV is high enough.

However, if the octree has some free cells at zero altitude in the map's local coordinates, the trajectory may traverse the ground (i.e. Z = 0, local frame) without having it detected as an obstacle. Thus, a trajectory traversing the ground in these conditions (and effectively going outside the map's boundaries) would be considered globally feasible if it crosses no other obstacles and is dynamically feasible. The solution is to have DEM maps that have at least one obstacle cell for altitude greater than zero (in the map's local frame).

7.4.4. Fitness Function & criteria

In contrast with other optimization algorithms, the values generated by GA's fitness functions do not need to be bounded in any way. Thus, the costs associated to the different criteria do not need to be proportional to the physical aspects they represent. In practice, scaling problems may occur if the relative weights and cost values associated with each criterion are out of proportion. Thus, all weights and criteria are normalized. However, it was found during tests that the weight associated to the obstacle distance deviation criteria had to be set higher than any other in order to obtain trajectories that do not cross any obstacles. The weight could be varied between 0.375 and 0.45. Higher values left little improvements in the results with respect to the other criteria.

Having more than four criteria at the same time did not result in a significant optimization of the corresponding aspects of the trajectory. After comparing several optimization trials performed by the GA with the same mission statement, it seems that a maximum of three criteria should be considered in order to see significant improvements for all of them.

Considering these criteria, it was found that the second factor having the most impact on the optimality of the path was the range ratio. For the same weight changes, modifying the time ratio did not shorten the resulting trajectory as much as the range ratio did. The small influence that the time ratio had on the optimization process may also be related to the fact that constraints regarding the desired speed (i.e. one of the ASO parameters) were already hard coded in the TG module. This would suggest that the integration of constraints into lower-level modules may decrease the number of criteria that need to be taken into account in the fitness function. Other key aspects may then be optimized with greater efficiency using the fitness function.

7.4.5. Number of generations vs. number of chromosomes

After testing missions with two or more objectives, it was found that increasing the number of chromosomes or increasing the maximum number of generations resulted in solutions with approximately the same level of optimization. However, increasing the number of chromosomes while keeping the number of generations low, significantly reduced the processing time of the algorithm. Successful trials used up to 80 chromosomes for a maximum of 30 generations.

Sensitivity of parameters

The main limitation of the system is the GA parameters described in Appendix-C. After many trials, it must be concluded that different sets of values for these parameters greatly changes the performance of the GA with respect to processing time and trajectory's optimality. Moreover, the best set of parameters' value changes with the ASO strategies ratings, and the relief of the terrain. Unfortunately, no formal procedure exists to optimize these parameters. However, different heuristics could be implemented for that purpose. But this is the subject of future developments.

Chapter 8: Conclusion and Future Work

Addressing the need for intelligent controllers, this work presents a control scheme that autonomously solves the task allocation problem (TAP) in real-time for on-line trajectory planning of individual UAV's. The system takes a generic approach to integrate UAV dynamics using the standard performance equations for aircraft. It also takes into account constraints with respect to times of arrival, vectors of approach close to target points, multiple objectives, weather conditions and behavioural guidelines with respect to speed, altitude and obstacle distance.

The knowledge base of the system relies on a hybrid octree which stores the atmospheric conditions comprising the temperature, pressure, density, wind direction and wind intensity, in an adaptable structure which minimizes memory usage to represent the terrain relief of the area over which the UAV must fly. The structure was also designed to integrate user-defined information which can be mapped over the area of interest (e.g. radar exposure). Experimental results have shown that for large areas with an obstacle density of less than 50%, the memory requirements of a hybrid octree could be reduced to 3% of the ones for a standard map (i.e. an evenly divided 3D map with a linear data structure). These results are of special interest for UAV applications since the topography of typical 3D maps considered for such applications rarely exceeds an obstacle density of 40 %.

The core of the system's path planner was based on a multi-objectives genetic algorithm (GA) with a fitness function having the capability of taking into account 7 or more criteria. Chromosomes were defined as possible paths linking all objectives specified in a mission. A trajectory generator integrating the performance equations of aircraft was interfaced with the GA to compute dynamically feasible trajectories for each chromosome. Simulations have show that the system is capable of the solving the TAP within 10 to 20 seconds for missions with one objective, using a PC with a Pentium®4 of 2.54 MHz. All resulting trajectories were dynamically feasible, and could be readily used by the autopilot. All solutions avoided all terrain elements while respecting behavioural constraints related to speed and altitude, providing a workable tool to define tactical aspects of missions. Considering that no code optimization took place, these results demonstrate the feasibility of using this control scheme for real-time applications.

Although the implementation of the system can be considered as a success when considering its performance under its current state of development, some limitations have been observed. Acknowledging the fact that no formal method exists to tune the mission-dependent parameters of the genetic algorithm, the first limitation lies in the difficulty of finding the right set of values for them. The second one lies in the number of fitness function criteria that can be effectively optimized during the evolution process. Although more than seven criteria can theoretically be optimized simultaneously, simulations have shown that the GA can't deal with more than four at a time. Moreover, the one related to *obstacle distance* must always be the dominant criterion (according to the weight distribution of the fitness function) in order to obtain trajectories that do not cross terrain elements. The last system's limitation concerns its real-time processing capabilities which are severely impaired for missions with more than one goal and one objective.

Future developments of this promising control scheme for UAV's should first focus on decreasing the computational load of the GA's optimization process. To that effect, the different constraints currently implemented through criteria to be optimized in the fitness function should be hard coded into the lower-level modules of the software (such as the trajectory generator). This strategy would ensure that constraints are satisfied while the total number of criteria is reduced to a level where all of them can effectively be optimized during the evolution process. The second step should consist of refining the fuel consumption model of the trajectory generator, in order to obtain a more realistic assessment of each trajectory's feasibility. The third step should aim to design an interface module to integrate the information of the dynamic layers of the hybrid octree into the fitness evaluation process. The last step in the development of this system should attempt to replace the algebraic dynamic model used in the trajectory generator by one that can provide the same results with a much shorter time response. Backpropagation neural networks may be good candidates, their time response being particularly fast once trained. This development should also include extending the model to propeller aircraft.

As a final remark, it should be said that the system proposed herein shows very good performance for its current state of development, and thus, constitute a very promising avenue for the next generation of UAV's path planners.

References

Al-Hasan, S. and Vachtsevanos, G., 2002, "Intelligent route planning for fast autonomous vehicles operating in a large natural terrain", *Robotics and Autonomous Systems*, vol 40, pp 1-24

Allouche, M., 2000, "The Integration of UAVs in Airspace", Operations & Safety, *Air & Space Europe*, vol 2, No 1, pp 101-104

Anderson, E.P., Beard, R.W., and McLain T.W., 2005, "Real-Time Dynamic Smoothing for Unmanned Air Vehicles", IEEE transactions on Control Systems Technology, Vol 13, No 3, pp 471-477

Autere, A., 2002, "Hierarchical A* based path planning – a case study", *Knowledge-Based Systems*, vol 15, pp 53-66

Barbier M. and Chanthery, E., 2004, "Autonomous mission management for unmanned aerial vehicles", *Aerospace Science and Technology*, vol 8, pp 359-368

Beard, R.W., McLain, T.W., Goodrich, M.A., Anderson, E.P., 2002, "Coordinated target assignment and intercept for unmanned air vehicles", *IEEE Transactions on Robotics and Automation*, vol 18, No 6, pp 911-922

Bellingham J., Tillerson, M., Richards, A. and How, J.P., 2003, "Multi-task allocation and path planning for cooperating UAVs", extract from the book: "Cooperative Contol: Models, Applications and Algorithms", edited by S. Butenko, R. Murphey, P.M. Pardalos, published by Kluwer Academic, Chapter 2, pp 23-41

Blyenburgh, P. V. 1999, "UAVs: an Overview", Uninhabited Aerial Vehicles (UAVs), *Air & Space Europe*, vol 1, No 5/6, pp 43-47

Brock, D.L., Montana, D.J. and Ceranowicz, A.Z., 1992, "Coordination and Control of Multiple Autonomous Vehicles", *IEEE Proceedings of the International Conference on Robotics and Automation*, Nice, France, May, pp 2725-2730

Brumitt, B. and Stentz, A., 1998, "GRAMMPS: A generalized mission planner for multiple robots in unstructured environments", *Proceedings of the IEEE International Conference on Robotics & Automation*, Leuven, Belgium, May, pp 1564-1571

Brumitt, B., Stentz, A., Hebert, M. and the CMU UGV group, 2001, "Autonomous Driving with Concurrent Goals and Multiple Vehicles: Mission Planning and Architecture", *Autonomous Robots*, vol 11, pp 103-115

Brumitt, B., Stentz, A., Hebert, M. and the CMU UGV group, 2002, "Autonomous Driving with Concurrent Goals and Multiple Vehicles: Experimental and Mobility Components", *Autonomous Robots*, vol 12, pp 135-156

Cameron, S., 1994, "Obstacle Avoidance and Path Planning", *Industrial Robot*, vol 21, No 5, pp 9-14

Chandler, P.R., Pachter, M., Rasmussen, S.R., Schumacher, C., 2003, "Distributed Control for Multiple UAVs with Strongly Coupled Tasks", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 11-14 August, Austin, Texas, USA

Choi, S.K. and Yuh, J., 1996, "Experimental Study on a Learning Control System with Bound Estimation for Underwater Robots", *Autonomous Robots*, vol 3, pp 187-194

Apollo Shuttle Systems (1998): http://www.ksc.nasa.gov/shuttle/technology/

Conolly, C.I. and Grupen, R.A., 1993, "The applications of harmonic functions to robotics", *Robotic Systems*, vol 10, pp 931-946

Dossier (1) 1999, "The UAV Industry", Uninhabited Aerial Vehicles (UAVs), *Air & Space Europe*, vol 1, No 5/6, pp 48-50

Dossier (2) 1999, "Current and Future UAV Military Users and Applications", Uninhabited Aerial Vehicles (UAVs), *Air & Space Europe*, vol 1, No 5/6, pp 51-58

EUROCONTROL and IfEN, 1998, WGS 84 Implementation Manual, Version 2.4, Prepared by the European Organization for the Safety of Air Navigation, Brussels, Belgium, and the Institute of Geodesy and Navigation, University FAF Munich, Germany

Eshelby, M.E., 2000, "Aircraft Performance: Theory and Practice", *AIAA publication*, Education series.

Gesch D. and Greenlee S., 1997, GTOPO30 Documentation, U.S. Geological Survey's EROS Data Center in Sioux Falls, South Dakota, Available from: http://www1.gsi.go.jp/geowww/globalmap-gsi/gtopo30/README.html#h1 [Accessed june 2005]

Goodrich, M. and Tamassia, R., 2001, "Data Structure and Algorithms in Java", Published by Wiley, 2nd edition.

Gundersen, R.W., Olsen, G.C. and Powell, G.E., 1996, "A fuzzy-inference scheme for fully autonomous control of rover class planetary exploration vehicles", *Space Technol.*, vol 16, No 5/6, pp 281-290

Herwitz, S.R., Johnson, L.F., Dunagan, S.E., Higgins, R.G., Sullivan, D.V., Zheng, J. , Lobitz, B.M., Leung, J.G., Gallmeyer, B.A., Aoyagi, M., Slye and R.E., Brass, J.A., 2004, "Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support", *Computers and Electronics in Agriculture*, vol 44, pp 49-61

Hodge, N.E., Zhixia Shi, L. and Trabia, M.B., 2004, "A distributed fuzzy logic controller for an autonomous vehicle", *Robotic Systems*, vol 21, No 10, pp 499-516

Huntsberger, T. and Rose, J., 1998, "BISMARC: a biologically inspired system fir map-based autonomous rover control", Neural Networks, vol 11, pp 1497-1510

Kanakakis, V., Valavanis, K.P. and Tsourveloudis, N.C., 2004, "Fuzzy-logic based navigation of underwater vehicles", *Intelligent and Robotic Systems*, vol 40, pp 45-88

Kassim, A.A. and Vijaya Kumar, B.V.K., 1997, "The wave expansion neural network", *Neurocomputing*, vol 16, pp 237-258

Kong, J., Lihui Gu, D., Gerla M. and Lu S., 2002, "Adaptive security for multilevel ad hoc networks", *Wireless Communications and Mobile Computing*, vol 2, pp 533-547

Kreienbaum B., 2000, "Unmanned Air Vehicles are Taking Off in Nato's Priorities", Uninhabited Aerial Vehicles (UAVs), *Air & Space Europe*, vol 2, No 1, pp 26-30

Le Fort-Piat, N., Collin, I., Meizel, D., 1997, "Planning robust displacement missions by means of robot-tasks and local maps", *Robotics and Autonomous Systems*, vol 20, pp 99-114

Louste, C. and Liegeois, A., 2000, "Near Optimal Robust Path Planning for Mobile Robots: the Viscous Fluid Method with Friction", *Intelligent and Robotic Systems*, vol 27, pp 99-122

Lucas, A., Rönnquist, R., Ljungberg, M., Howden, N., Corke, P. and Sikka, P., 2003, "Teamed UAVs – a new approach with intelligent agents", *2nd AIAA Conference on "Unmanned Unlimited" Systems, Technologies, and Operations – Aerospace*, 15-18 September, San Diego, CA, USA.

Luger, G.F., 2002, "Artificial Intelligence, Structures and Strategies for Complex Problem Solving", Published by Addison Wesley, 4th edition

McLain, T.W., Chandler, P.R., Rasmussen, S., Pachter, M., 2001, "Cooperative Control of UAV Rendez-vous", *IEEE Proceedings of the American Control Conference*, 25-25 June, Arlington, VA, USA, pp 2309-2314

Murda, R. and Douglas, R.J., 2003, "Self-correction mechanism for path integration in a modular navigation system on the basis of an egocentric spatial map", *Neural Networks*, vol 16, pp 1373-1388

Murphy R.R., Hughes, K., Marzilli, A., Noll, E., 1999, "Integrating explicit path planning with reactive control of mobile robots using Trulla", *Robotics and Autonomous Systems*, vol 27, pp 225-245

Naeem, W., Sutton, R., Ahmad S. M. and Burns, R.S., 2003, "A Review of Guidance Laws Applicable to Unmanned Underwater Vehicles", *J. of Navigation*, vol 56, pp 15-29

Nearchou, A.C., 1999, "Adaptive navigation of autonomous vehicles using evolutionary algorithms", *Artificial Intelligence in Engineering*, vol 13, pp 159-173

News 2002, "UAVs in the US and EU", *Photovoltaics Bulletin*, September 2002, pp 2-3

Payton D.W., Rosenblatt, J.K. and Keirsey, D.M., 1990, "Plan guided reaction", IEEE Transactions on Systems, Man and Cybernetics, vol 20, No 6, pp 1370-1382

Podsedkowski, L., Nowakowski, J., Idzikowski, M., and Vizvary, I., 2001, "A new solution for path planning in partially known or unknown environment for nonholonomic robots", *Robotics and Autonomous Systems*, vol 34, pp 145-152

Prestes, E., Engel, P.M., Trevisan, M., Idiart, M.A.P., 2002, "Exploration method using harmonic functions", *Robotics and Autonomous Systems*, vol 40, pp 22-42

Prestes, E., Idiart, M.A.P., Engel, P.M. and Trevisan, M., 2001, "Exploration technique using potential fields calculated from relaxation methods", *IEEE Proceedings of the International Conference on Intelligent and Robotic Systems*, Maui, Hawaii, USA, Oct 29- Nov 03, pp 2012-2017

Prestes, E., Idiart, M.A.P., Trevisan, M. and Engel, P.M., 2004, "Autonomous Learning Architecture for Environmental Mapping", *Intelligent and Robotic Systems*, vol 39, pp 243-263

Rabbath, C.A., Gagnon, E., Lauzon, M., 2004, "On the Cooperative Control of Multiple Unmanned Aerial Vehicles", *IEEE Canadian Review*, Spring, pp 15-19

Ramirez, G. and Zeghloul, S., 2001, "Collision-free path planning for nonholonomic mobile robots using new obstacle representation in the velocity space", *Robotica*, vol 19, pp 543-555

Ratering, S. and Gini, M., 1993, "Robot Navigation in a Known Environment with Unknown Moving Obstacles", *IEEE Proceedings, International Conference on Robotics and Automation*, 2-6 May, pp 25 – 30

Refanidis, I. and Vlahavas, I., 2003, "Multiobjective heuristic state-space planning", *Artificial Intelligence*, vol 145, pp 1-32

Richards, A., Bellingham, J., Tillerson, M. and How, J., 2002, "Coordination and control of multiple UAVs", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 5-8 August, Monterey, CA, USA

Sasiadek, J.Z. and Duleba, I., 2000, "3D Local Trajectory Planner for UAV", *Intelligent and Robotic Systems*, vol 29, pp 191-210

Schmidt, G.K. and Azarm, K., 1992, "Mobile Robot Navigation In A Dynamic World Using an Unsteady Diffusion Equation Strategy", *IEEE Proceedings International Conference on Intelligent Robots and Systems*, vol 1, July 7-10, pp 642 – 647

Singh, L., and Fuller, J., 2001, "Trajectory Generation for a UAV in Urban Terrain using Nonlinear MPC", *IEEE Proceedings of the American Control Conference*, Arlington, VA, USA, June 25-27, pp 2301-2308

Singh, S. N, Chandler, P., Schumacher, C., Banda S., and Pachter, M., 2000, "Nonlinear Adaptive Close Formation Control of Unmanned Aerial Vehicles", *Dynamics and Control*, vol 10, pp179-194

Singh, S. N, Zhang, R., Chandler, P. and Banda, S., 2003, "Decentralized nonlinear robust control of UAVs in close formation", *International J. of Robust and Nonlinear Control*, vol 13, 1057-1078

Singh, S. N., Pachter, M., Chandler, P., Banda, S., Rasmussen, S. and Schumacher, C., 2000, "Input-output invertibility and sliding mode control for close formation flying of multiple UAVs", *International J. of Robust and Nonlinear Control*, vol 10, 779-797

Smierzchalski, R., 1999, "Evolutionary trajectory planning of ships in navigation traffic areas", *Marine Science and Technology*, vol 4, pp 1-6

Soltani, A.R., Tawfik, H., Goulermas, J.Y., Fernando, T., 2002, "Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms", *Advanced Engineering Informatics*. Vol 16, pp 291-303

Stentz, A., 1994, "Optimal and Efficient Path Planning for Partially-Known Environments", *IEEE Proceedings ICRA*, pp 3310-3317

Švestka, P. and Overmars, M.H., 1998, "Coordinated path planning for multiple robots", *Robotics and Autonomous Systems*, vol 23, pp125-152

Tian, L. and Collins, C., 2004, "An effective robot trajectory planning method using a genetic algorithm", *Mechatronics*, vol 14, pp 455-470

Tierno, J.E., 2001, "Distributed autonomous control of concurrent combat tasks", *IEEE Proceedings of the American Control Conference*, 25-27 June, Arlington, VA, USA.

Tsoukalas, L.H., Houstis, E.N. and Jones, G.V.,1997, "Neurofuzzy motion planners for intelligent robots", *Intelligent and Robotic Systems*, vol 19, pp 339-356

Wang, J-S, George Lee, C.S., and Yuh, J., 1999, "An on-line self-organizing neuro-fuzzy control for autonomous underwater vehicle", *IEEE Proceedings of the International Conference on Robotics and Automation*, Detroit, Michigan, May, pp 2416-2421

Wang, J-S, George Lee, C.S., and Yuh, J., 2000, "Self-adaptive neuro-fuzzy systems with fast parameter learning for autonomous underwater vehicle control", *IEEE Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, April, pp 3861-3866

Williams, M. and Jones, D.I., 2001, "A rapid method for path planning in three dimensions for a small aerial robot", *Robotica*, vol 19, pp125-135

Winston, W.L. and Venkataramanan, M., 2003, "Introduction to Mathematical Programming", Operations Research: Volume One, Published by Thomson Brooks/Cole, 4th edition, pp 527, 530-538, 800-814

Yahja, A., Singh, S. and Stentz, A., 2000, "An efficient path planner for outdoor mobile robots", *Robotics and Autonomous Systems*, vol 32, pp 129-143

Yang, S.X. and Meng, M., 2000, "An efficient neural network approach to dynamic robot motion planning", *Neural Networks*, vol 13, pp 143-148

Zheng, C., Ding, M., Zhou, C. and Li, L., 2004, "Coevolving and cooperation path planner for multiple unmanned air vehicles", *Engineering Applications of Artificial Intelligence*, vol 17, pp 887-896

Appendix-A: DEM standard

DEM STANDARD [http://www1.gsi.go.jp/geowww/globalmap-gsi/gtopo30/README.html#h1]

Data for each DTED map are provided in a set of 2 files. The following extensions are used:

Extension	Contents
DEM	digital elevation model data
HDR	header file for DEM

1. DEM File (.DEM)

The DEM is provided as 16-bit signed integer data in a simple binary raster. There are no header or trailer bytes imbedded in the image. The data are stored in row major order (all the data for row 1, followed by all the data for row 2, etc.).

2. Header File (.HDR)

The DEM header file is an ASCII text file containing size and coordinate
information for the DEM. The following keywords are used in the header file:

BYTEORDER	byte order in which image pixel values are stored M = Motorola byte order (most significant byte first) L = Intel byte order (Least significant byte first)
LAYOUT	organization of the bands in the file BIL = band interleaved by line (note: the DEM is a single band image)
NROWS	number of rows in the image
NCOLS	number of columns in the image
NBANDS	number of spectral bands in the image (1 for a DEM)
NBITS	number of bits per pixel (16 for a DEM)
BANDROWBYTES	number of bytes per band per row (twice the number of columns for a 16-bit DEM)
TOTALROWBYTES	total number of bytes of data per row (twice the number of columns for a single band 16-bit DEM)
BANDGAPBYTES	the number of bytes between bands in a BSQ format image (0 for a DEM)
NODATA	value used for masking purposes
ULXMAP	longitude of the center of the upper-left pixel (decimal degrees)
ULYMAP	latitude of the center of the upper-left pixel (decimal degrees)
XDIM	x dimension of a pixel in geographic units (decimal degrees)
YDIM	y dimension of a pixel in geographic units (decimal degrees)

Example header file (Example.HDR):

```
BYTEORDER      M
LAYOUT         BIL
NROWS          6000
NCOLS          4800
NBANDS         1
NBITS          16
BANDROWBYTES   9600
TOTALROWBYTES  9600
BANDGAPBYTES   0
NODATA         -9999
ULXMAP         -99.99583333333334
ULYMAP         39.99583333333333
XDIM           0.00833333333333
YDIM           0.00833333333333
```

Appendix-B: Subsonic Aircraft Dynamic Model

The theory summarized in this appendix is mainly based on *Eshelby 2000.*

B.1. Assumptions

The Global Planner (and the trajectory generator taking care of the UAV's dynamics) uses the *performance equations* described in this appendix. It assumes normal flying conditions. In other words, maneuvers such as loops, extreme turns, intentional stalls or flying upside down, are not considered. Such maneuvers are usually required in "immediate threat" situations which fall under the responsibility of the Local Planner. Therefore, they are not taken in account at the level of the Global Planner.

The Global Planner was designed to plan paths suitable for UAV's with operational constraints similar to those of military transport aircrafts.

B.2. Reference frames

The performance equations use a series of reference frames to simplify the integration of various aerodynamic forces into the final set of equations of motion. These reference frames link the aircraft reference frame to the Earth reference frame which is assumed fixed. The five reference frames are:

- *Earth frame* [F_e], a reference frame attached to a fixed point on the ground.
- *Body frame* [F_b], a reference frame attached to the body of the aircraft.
- *Stability frame* [F_S], a reference frame defined by the wind direction with a zero side slip angle
- *Wind frame* [F_W], a reference frame defined by the true direction of the wind
- *Velocity frame* [F_V], a reference frame defined by the path of the aircraft relative to Earth

These five reference frames are all related by simple rotational transform matrices. The angles establishing the relation between one another are listed in Table B. 1. These angles can be visualized in Figure B. 1.

Table B. 1: *Angles defining the aircraft reference frames*

Axis Transfer	*Earth to Body*	*Body to Stability*	*Stability to Wind*	*Wind to Velocity*	*Velocity to Earth*
Pitch	θ	α	-	-	γ_2
Roll (Bank)	Φ	-	-	γ_1	-
Yaw (Heading)	ψ	-	β	-	γ_3

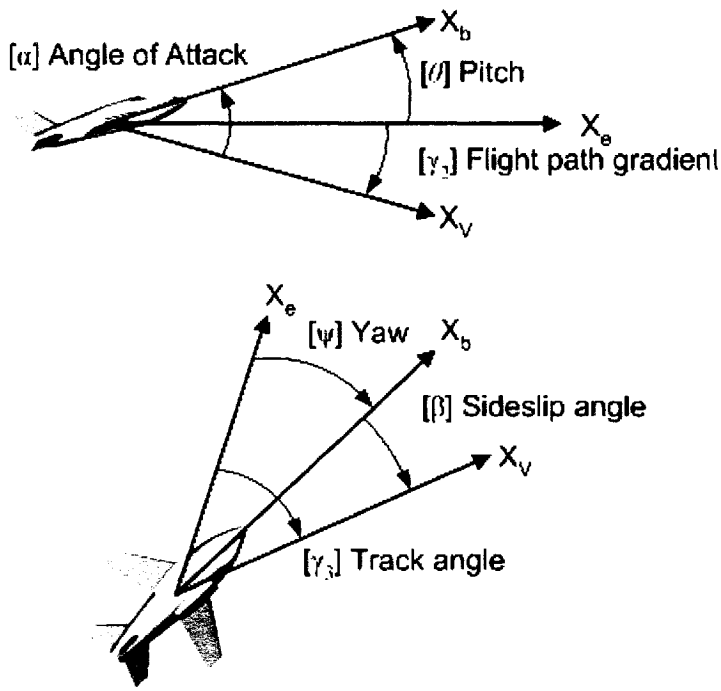


Figure B. 1: *Angles between reference frames*

The attitude of the aircraft is expressed in the Earth-to-Body reference frame as the pitch, roll (or bank) and yaw (or heading). These three parameters are represented in Figure B. 2 with the standard five forces acting on the body of the aircraft.

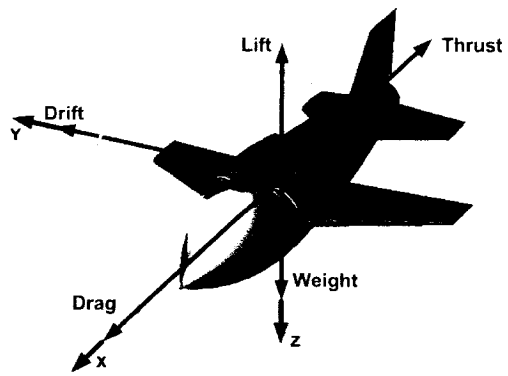


Figure B. 2: *Standard Reference axis and forces for wing-type aircrafts*

The last set of angles to be defined to characterize all parameters within the equations of motion corresponds to the angles defining the thrust direction as shown in Figure B. 3.

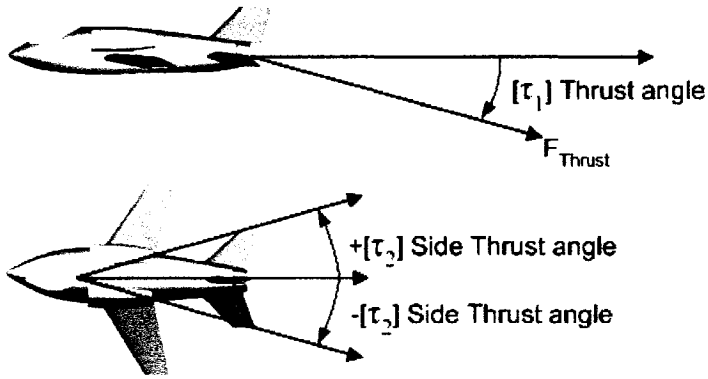


Figure B. 3: *Thrust angles*

The overall side thrust angle is usually assumed to be zero with the angle $+\tau_2$ and $-\tau_2$ canceling out the side component of the resultant force. Thus, the equation of motion used for the performance equation only considers the thrust angle τ_1.

B.3. Equations of motion

The closed form of the equations of motion for a winged aircraft is as follows (see *Eshelby 2000* for a complete description):

$$\begin{bmatrix} -D \\ Y\cos\gamma_1 + L\sin\gamma_1 \\ Y\sin\gamma_1 - L\cos\gamma_1 \end{bmatrix} + \begin{bmatrix} T\cos(\alpha+\tau_1)\cos(\beta) - D_m \\ T\left[-\cos(\alpha+\tau_1)\sin(\beta)\cos(\gamma_1) + \sin(\alpha+\tau_1)\sin(\gamma_1)\right] \\ T\left[-\cos(\alpha+\tau_1)\sin(\beta)\sin(\gamma_1) - \sin(\alpha+\tau_1)\cos(\gamma_1)\right] \end{bmatrix} + \begin{bmatrix} -W\sin\gamma_2 \\ 0 \\ W\cos\gamma_2 \end{bmatrix}$$
$$= m\begin{bmatrix} \dot{V} \\ V\dot{\gamma}_3\cos\gamma_2 \\ -V\dot{\gamma}_2 \end{bmatrix} + \dot{m}\begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix} \qquad \text{B. 1}$$

This equation is given in the velocity reference frame F_V (for which the Y axis is parallel with the one of the Earth reference frame F_e).

B.4. Definition of the basic forces acting on the aircraft

According to Newton's law, the complete set of forces acting on an aircraft can be summarized as:

$$\vec{F}a + \vec{F}p + \vec{F}g = \vec{F}_I \qquad \text{B. 2}$$

with Fa, the aerodynamic forces
Fp, the propulsive forces
Fg, the gravitational forces
F_I, the inertial forces

This section presents an overview of the aerodynamic forces as well as the propulsive force for jet aircrafts.

B.4.1. Lift

The general equation of the lift (force) L can be expressed as:

$$L = \tfrac{1}{2} \rho V^2 S C_L \qquad \text{B. 3}$$

with ρ, the density of the air
V, the air speed
S, the gross wing area
C_L, the lift coefficient

Although the lift coefficient is a function of the wing configuration (i.e. flaps, trailing edge, etc.), the angle of attack and the Mach number, it assumed as constant equal to the maximum reachable C_L under level flight conditions in cruise conditions. This assumption remains reasonable for aircraft operating below Mach 0.8 (i.e. sub-sonic flights). Other parameters can be adjusted dynamically during the flight to maintain this constant lift coefficient.

B.4.2. Drag

Assuming subsonic-sonic flights, the parabolic drag profile can be used as a reasonable approximation of the drag (force) D:

$$D = \frac{1}{2} \rho V^2 S \left[C_{Dz} + K \left(\frac{n \cdot W}{\frac{1}{2} \rho V^2 S} \right)^2 \right] \qquad \text{B. 4}$$

with C_{Dz}, the lift independent drag coefficient
K, the lift dependent drag factor
W, the weight of the aircraft
n, the load factor of the aircraft (which depends on maneuver):

$$n = L / W \qquad \text{B. 5}$$

The lift dependent factor K is a function of the aspect ratio, wing span efficiency, lift of tail plane and the vortex drag. For Mach number less than 0.8, it can be assumed as a constant which can be computed as:

$$K = 1 / (A\, e\, \pi) \qquad \text{B. 6}$$

with A, the wing aspect ratio of the plane

e, the wing span efficiency

The lift independent factor C_{Dz} can also be assumed as constant for $M < 0.8$.

B.4.3. Weight & Thrust

The remaining two forces are the Weight and the Thrust of the aircraft. Since the total mass of fuel required for a mission represents a significant fraction of the total aircraft mass, the weight is a time-dependent force which needs to take in account the fuel consumption rate. In contrast, the Thrust force may remain relatively constant through out a given flight. However, the equations describing the thrust are very different between a propeller aircraft and a jet aircraft. Based on the current literature, no analytical close form solutions are available yet to integrate the thrust equations of propeller aircrafts into the motion equations. Only iterative computer solutions or other complex analysis software are capable of finding solutions for this problem. On the other hand, the thrust of jet aircrafts may be easily modeled using an empirical algebraic formula which depends solely on the relative air density. Since the goal of this work is more related to a proof of concept rather than an attempt to model complex aircrafts dynamics, only jet aircrafts are considered here:

$$F_N = F_{N0} \cdot \sigma^x \qquad \text{B. 7}$$

with F_N, the net thrust

F_{N0}, the net thrust at sea-level

σ, the relative air density

x, an empirical constant (~ 0.7 for troposphere, ~1 for stratosphere)

The fuel consumption C (in kg) of jet aircraft can be approximated using several empirical models with varying level of complexity. The one used in for the system is:

$$C = C_2\ \theta^{1/2}\ M^n \qquad \text{B. 8}$$

with C_2, an empirical constant

θ, the relative atmospheric temperature

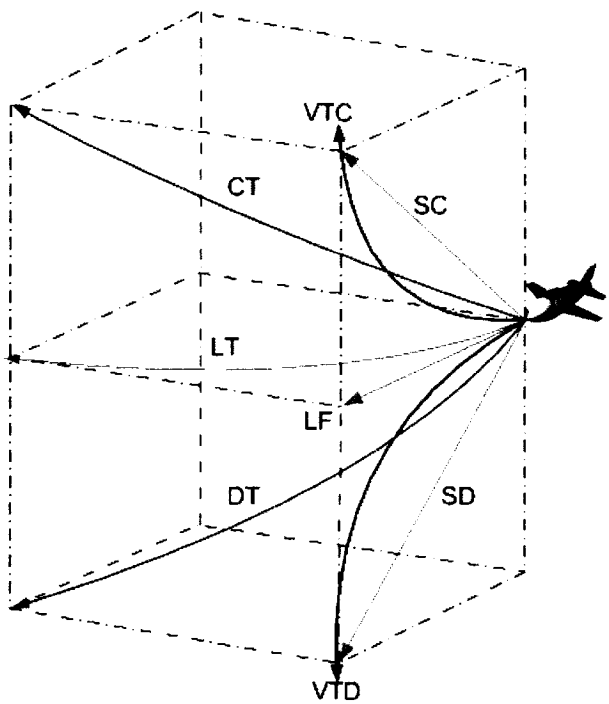
M, the Mach number

n, an empirical constant

B.5. Standard maneuvers

Using a simple 3D representation, the standard maneuvers considered are shown in Figure B. 4. All equations presented in this section are based on two assumptions:

1. The amount of time required to perform each maneuver is considered short enough to neglect the change of mass occurring during it (i.e. fuel consumed is assumed negligible compared to other factors in the motion equations).
2. All maneuvers are considered being performed under steady flight conditions, i.e. at constant speed.



VTC: Vertical Turn & Climb
SC: Steady Climb
CT: Climb & Turn
LF: Level Flight
LT: Level Turn
DT: Descent & Turn
SD: Steady Descent
VTD: Vertical Turn & Descent

Figure B. 4: *Standard manoeuvres*

B.5.1. Level Flight & Maneuver envelope

The level flight is described by the following set of parameters:

Table B. 2: *Level flight parameters*

Conditions	Parameters
Straight flight	$\dot{\gamma}_2 = 0$
	$\dot{\gamma}_3 = 0$
	$\beta = 0$
	$Y = 0$
Steady flight	$\dot{m} = 0$
	$\dot{V} = 0$
Level flight	$\gamma_1 = 0$
	$\gamma_2 = 0$

The equations of motion can then be expressed as:

$$\begin{bmatrix} -D + T\cos(\alpha + \tau_1) \\ 0 \\ -L - T\sin(\alpha + \tau_1) + W \end{bmatrix} = 0 \Rightarrow \begin{bmatrix} F_N \\ 0 \\ L \end{bmatrix} = \begin{bmatrix} D \\ 0 \\ W \end{bmatrix} \qquad \text{B. 9}$$

assuming : $L >> T\sin(\alpha + \tau_1)$

Based on these equations and the basic definition of the aerodynamic forces outlined in section B.4, the following velocities can be defined:

$$V_{Stall(EAS)} = \left(\frac{2W}{\rho_0 S C_{L\max}} \right)^{1/2}$$

$$V_{Critical(EAS)} = \left(\frac{M_{critical}{}^2 \delta p_0 \gamma}{\rho_0} \right)^{1/2} \qquad \text{B. 10}$$

$$V_{md(EAS)} = \left(\frac{2W}{\rho_0 S} \right)^{1/2} \left(\frac{K}{C_{Dz}} \right)^{1/2}$$

with V_{EAS}, the equivalent air speed

ρ_0, the density of the air at sea level

p_0, the air pressure at sea level

δ, the relative air pressure

$M_{critical}$, the maximum Mach number for subsonic conditions (~0.8)

C_{Lmax}, the maximum lift coefficient

γ, a constant equal to 1.4 under subsonic conditions

The EAS is the equivalent speed at which the aircraft would travel if it was flying at sea-level. The conversion between true airspeed and EAS is given as follows:

$$V_{EAS} = V\sqrt{\sigma} \qquad \text{B. 11}$$

Using equations B. 4, B. 6 and B. 9, it is possible to define the flight maneuver envelope. This envelope, represented as the red area in Figure B. 5 determines the range of speeds at which the aircraft can travel while maintaining a thrust force greater or equal to the drag force, given a certain altitude.

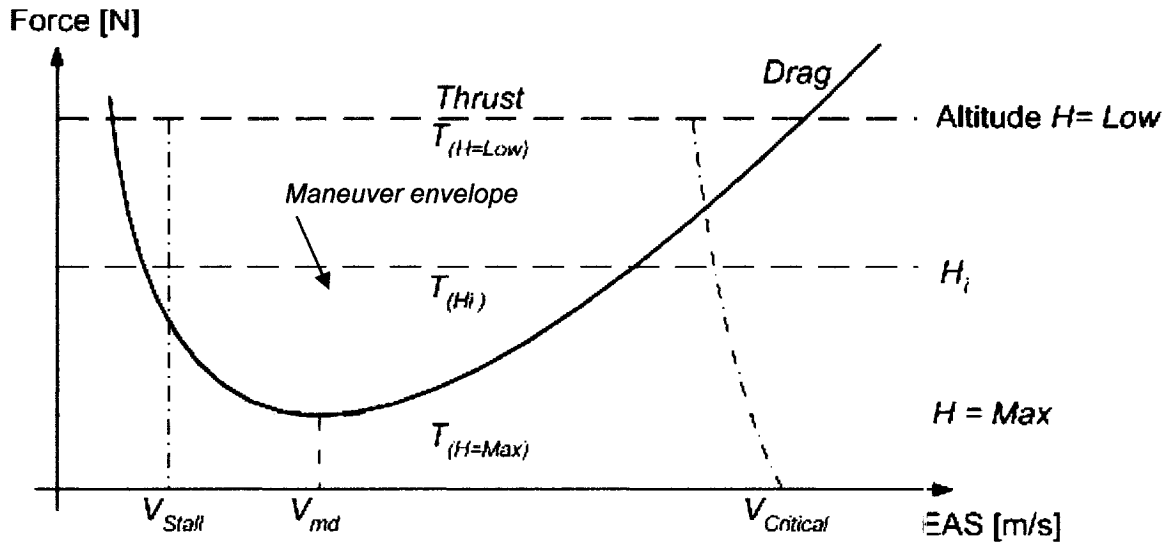


Figure B. 5: *Maneuver envelope for level flight*

The concept of the maneuver envelope can be generalized to any kind of maneuver. In order to do so, the load factor associated with the maneuver of interest must be taken in account in the drag equation. Thus, the maneuver envelope becomes a function of air speed and altitude (as above) as well as a function of the load factor (which is a measure of the acceleration of the aircraft during that maneuver). Figure B. 6 shows the general case of the maneuver envelope with varying load factors (which falls within the normal range in which transport aircraft must operate).

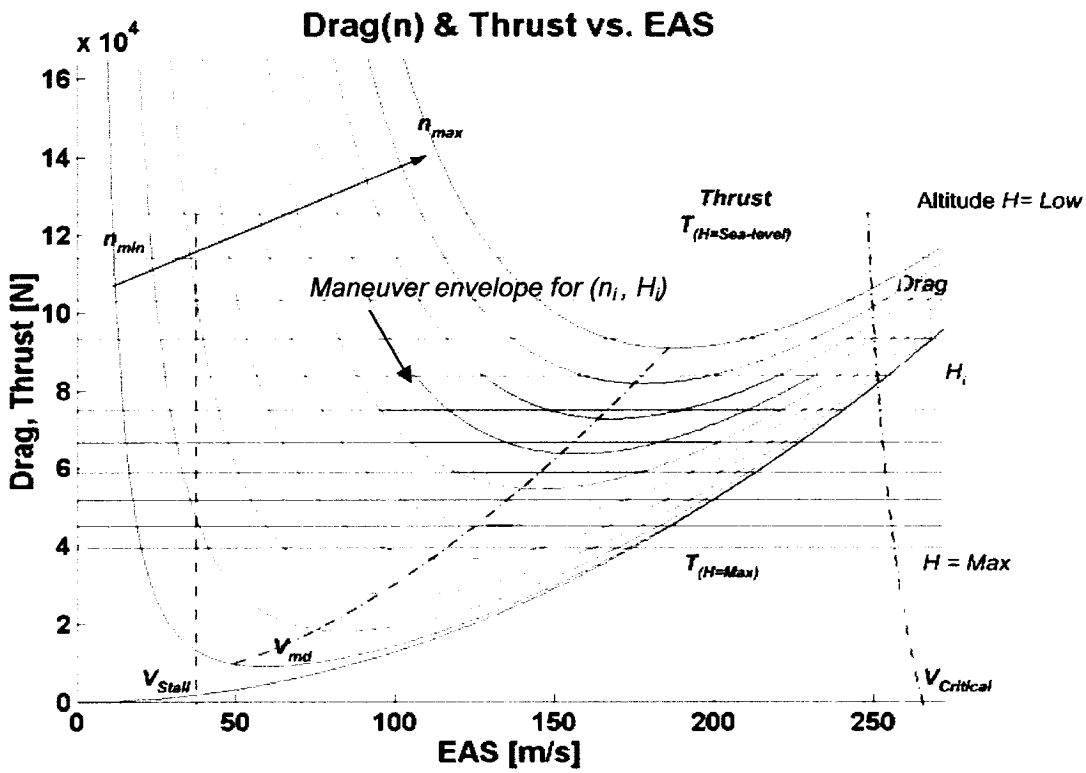


Figure B. 6: *General form of the manoeuvre envelope*

B.5.2. Steady Climb and Steady Descent

Steady Climb is described by the following set of parameters:

Table B. 3: *Steady climb & steady descent parameters*

Conditions	Parameters
Straight flight	$\dot{\gamma}_2 = 0$ $\dot{\gamma}_3 = 0$ $\beta = 0$ $Y = 0$
Steady flight	$\dot{m} = 0$ $\dot{V} = 0$
Constant gradient	$\gamma_1 = 0$ $\gamma_2 =$ Constant

The equations of motion can then be expressed as:

$$\begin{bmatrix} -D+T\cos(\alpha+\tau_1)-W\sin\gamma_2 \\ 0 \\ -L-T\sin(\alpha+\tau_1)+W\cos\gamma_2 \end{bmatrix}=0 \Rightarrow \begin{bmatrix} F_N-D \\ 0 \\ L/W \end{bmatrix}=\begin{bmatrix} W\sin\gamma_2 \\ 0 \\ \cos\gamma_2 \end{bmatrix} \qquad \text{B. 12}$$

assuming : $L >> T\sin(\alpha+\tau_1)$

Based on these equations and the basic definition of the aerodynamic forces outlined in section B.4, the following relation can be defined for the flight path gradient angle:

$$\gamma_2 = \sin^{-1}\left(\frac{F_N - D|_{Hi,Vi}}{W}\right) \qquad \text{B. 13}$$

with $F_N\text{-}D|_{Hi,\,Vi}$, the difference between the net thrust and the drag at altitude Hi and velocity Vi

The maximum value of γ_2 is reached when the velocity is equal to the minimum drag velocity V_{md}. The maximum (i.e. climb) and minimum (i.e. descent) gradient angles are usually expressed in terms of the ratio between the maneuver's velocity V and the minimum drag velocity V_{md}., according to the following relations:

$$\begin{aligned} E\max &= L/D\min = 0.5\left(K C_{D_z}\right)^{-1/2} \\ \tau &= F_N / D\min \\ u &= V/V_{md} \\ u_{critical} &= V_{critical}/V_{md} \end{aligned} \qquad \text{B. 14}$$

The descent angle $\gamma_{2(min)}$ is given as:

$$\gamma_{2(\min)} = \tan^{-1}\left\{\frac{1}{E\max}\left(-\frac{1}{2}\left[u_{critical}{}^{2}+u_{critical}{}^{-2}\right]\right)\right\} \qquad \text{B. 15}$$

The climb angle $\gamma_{2(max)}$ is given as:

$$\gamma_{2(\max)} = \tan^{-1}\left\{\frac{1}{E\max}\left(\tau_{\max}-\frac{1}{2}\left[u_{md}{}^{2}+u_{md}{}^{-2}\right]\right)\right\} \quad with \quad u_{md}=\frac{V_{md}}{V_{md}}=1 \qquad \text{B. 16}$$

The maneuver envelope defining the variation of the gradient angle with respect to the relative velocity u is shown in Figure B. 7.

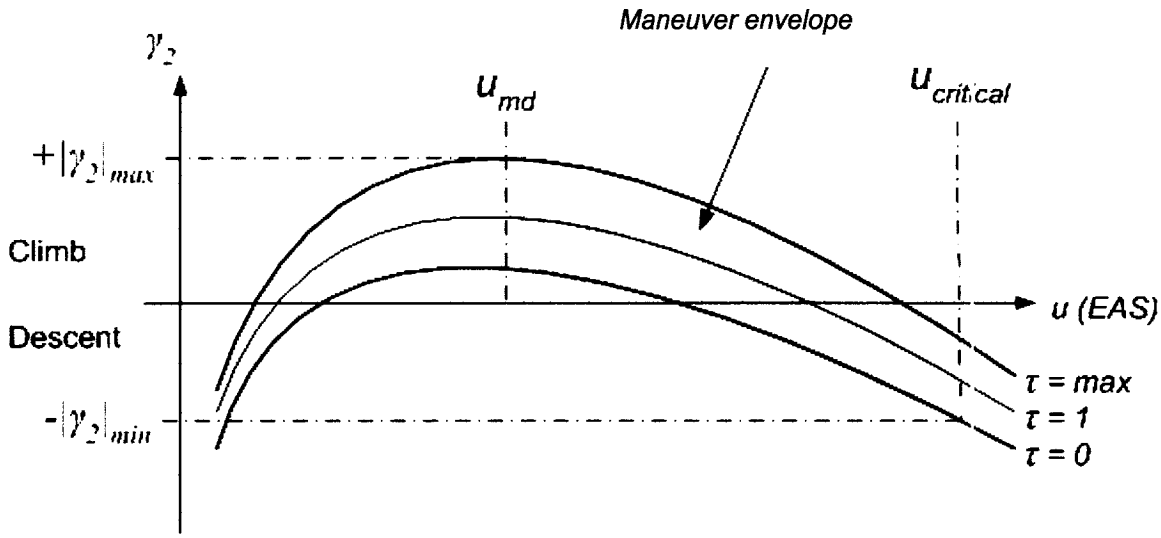


Figure B. 7: *Climb and Descent maneuver envelope*

The lower bound of the maneuver envelope set to the minimum drag speed in order to ensure flight path stability. Elevator control is no longer sufficient to keep the aircraft on its course when flown below that speed.

B.5.3. Level Turn

Steady Level Turn maneuver is described by the following set of parameters:

Table B. 4: *Level turn parameters*

Conditions	Parameters
Constant level turn	$\dot{\gamma}_2 = 0$ $\dot{\gamma}_3$ = Constant $\beta = 0$ $Y = 0$
Steady flight	$\dot{m} = 0$ $\dot{V} = 0$
Constant gradient	γ_1 = Constant $\gamma_2 = 0$

The main assumption is that the roll angle remains constant as well as the altitude of the aircraft. The equations of motion can then be expressed as:

$$\begin{bmatrix} -D+T\cos(\alpha+\tau_1) \\ L\sin\gamma_1 + T\sin(\alpha+\tau_1)\sin\gamma_1 \\ -L\cos\gamma_1 - T\sin(\alpha+\tau_1)\cos\gamma_1 + W \end{bmatrix} = \begin{bmatrix} 0 \\ V\dot{\gamma}_3 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} F_N - D \\ n\sin\gamma_1 \\ n \end{bmatrix} = \begin{bmatrix} W\sin\gamma_2 \\ V^2/(g\cdot R) \\ 1/\cos\gamma_1 \end{bmatrix} \quad \text{B. 17}$$

assuming:

$$L >> T\sin(\alpha + \tau_1),$$
$$n = L/W$$
$$\dot{\gamma}_3 = V/R = \text{Constant}$$

Assuming a maximum load factor n_1 the maximum bank angle is defined as:

$$\gamma_{1\max} = \cos\left(\frac{1}{n_1}\right) \qquad \text{B. 18}$$

This maximum load factor is generally set to a minimum by the FAR / JAR regulation. That minimum is given by the following empirical equations:

$$n_1 = 2.1 + \frac{24000}{0.225 \cdot W + 10\,000} \qquad \text{B. 19}$$

with n_1, the maximum load factor ranging from [2.5; 3.8] for commercial and cargo aircrafts.

The minimum radius of curvature is given by the expression:

$$R_{\min} = \frac{V_{\min}^2}{g \cdot \tan(\gamma_{1\max})} \qquad \text{B. 20}$$

The minimum velocity is given as:

$$V_{\min} = \left(\frac{W \cdot n_1}{\frac{1}{2}\rho_0 S C_{L\max}}\right)^{1/2} \qquad \text{B. 21}$$

with $C_{L\,max}$, the maximum steady flight lift coefficient

B.5.4. Vertical Turn

The vertical turn maneuver includes both the VTC and VTD maneuver shown in Figure B. 4. This maneuver is usually referred as a loop when carried over more than 270°. Assuming that the maximum and minimum pitch angle is limited to ±20° during the maneuver, the radius of curvature of the loop may be considered constant. Since most commercial and cargo aircrafts limit their pitch angle to ±15°, the variability of the loop's curvature will not be taken in account for the application considered herein. Moreover,

the velocity of the aircraft is considered constant throughout the maneuver. The vertical turn can be described by the following set of parameters:

Table B. 5: *Vertical turn parameters*

Conditions	Parameters
Straight flight	$\dot{\gamma}_2 = \text{Constant}$
	$\dot{\gamma}_3 = 0$
	$\beta = 0$
	$Y = 0$
Steady flight	$\dot{m} = 0$
	$\dot{V} = 0$
Constant gradient	$\gamma_1 = 0$
	$\gamma_2 \neq 0$

The equations of motion can then be expressed as:

$$\begin{bmatrix} -D + T\cos(\alpha + \tau_1) - W\sin\gamma_2 \\ 0 \\ -L - T\sin(\alpha + \tau_1) + W\cos\gamma_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -V\dot{\gamma}_2 \end{bmatrix} m \Rightarrow \begin{bmatrix} (F_N - D)/W \\ 0 \\ -n + \cos\gamma_2 \end{bmatrix} = \begin{bmatrix} \sin\gamma_2 \\ 0 \\ -\dfrac{V^2}{g \cdot R} \end{bmatrix} \quad \text{B. 22}$$

assuming :

$L >> T\sin(\alpha + \tau_1)$,

$\dot{\gamma}_2 = V / R = \text{Constant}$

Limiting the pitch angle to small values, it possible to assume that $\cos(\gamma_2) \cong 1$, such that the radius of curvature can be given as:

$$R = \frac{V^2}{g \cdot (n_1 - \cos\gamma_2)} \approx \frac{V^2}{g \cdot (n_1 - 1)} \quad \text{B. 23}$$

The minimum radius of curvature can be found using the minimum drag velocity:

$$R_{\min} = \frac{V_{stall}^2}{g} \quad \text{B. 24}$$

B.5.5. Climb/Descent and turn

The climb or descent and turn maneuvers shown in Figure B. 4 have been implemented in this work as a combination of a level turn and a vertical turn (see section 5.1.2.2.) under the assumption that each one is associated with its own load factor n, and that the norm of these two load factors is always less or equal than the total load factor allowable under the aircraft specifications:

$$n_1 \geq \sqrt{n_{vertical\ turn}{}^2 + n_{horizontal\ turn}{}^2} \qquad \text{B. 25}$$

This assumption is based on the fact that the loads are distributed in two orthogonal planes.

Appendix-C: MOGA – Tunable Parameters

Many parameters can be tuned to modify the Multi-Objective Genetic Algorithm's performance. All of these parameters can be found at the beginning of the *TaskAllocator.h* file. These parameters are briefly presented in this section.

C.1. General parameters

`MAX_RAND_STEPS`

This parameter establishes the maximum number of iteration permitted to find a free cell within the octree when performing the *Insert Transitional* and *Perturb Transitional* genetic operators. It is set to 50 by default.

C.2. Population parameters

`MAX_N_PCHROMOSOMES`

This parameter establishes the maximum number of chromosome that a population can have. Since the number of chromosome can be set by the user during the creation of a population, this limit makes sure that the value remains within system capabilities.

`MIN_N_PCHROMOSOMES`

This parameter complements the limits imposed on the population size. The defaults value is set to three based on the fact that genetic operators (including crossover) can not be applied on smaller populations.

`MAX_N_PGENE`

This indicates the maximum of gene that a chromosome can have This value is limited mainly by the computer's performance in terms of memory a processing speed.

`MIN_GENE_LENGTH, MAX_GENE_LENGTH`

This indicates the minimum and maximum number of transitional waypoints allowed for a gene (i.e. for one objective). The minimum value has great impact on the system's performance. If the minimum number of transitional waypoints is too

high, the final trajectory tends to make unnecessary detours. If the number is too low, the GA cannot find solutions more complex than a relatively straight path to the objective. These limit values must thus be set according to the complexity of the terrain considered for a given mission. As a general guideline, the `MIN_GENE_LENGTH` parameter should be kept below 3 and 5.

C.3. Genetic operators parameters

`PERTURBATION_LIMIT`

This parameter sets the maximum distance that can separate a path's transitional waypoint and a new one generated while performing the *Insert Transitional* genetic operator. This distance must be expressed as a fraction of the octree map (i.e. a number from 0 to 1). Small values in the order of 0.015 are best. Higher ones tend to systematically degrade the optimality of the altered chromosomes by forcing it to take long detour in its course.

`MIN_SHARP_TURN`

This parameter sets the threshold below which an angle is detected as "sharp". This parameter affects the sensitivity of the *Smooth Turn* genetic operator. Since the number of transitional waypoints is best kept low, the number of sharp turn is usually low as well. Thus, limiting the angle to a very small range almost prohibit the genetic operator from making any changes. Angles as high as 60 degrees have been used without negatively impacting the final results of the GA.

C.4. Evolutionary parameters

`POP_PERCENT_RENEW`

This parameter sets the percentage (i.e. a value between 0 and 1) of the population that is replaced by offspring at the end of each generation. It defines to a great extent the rate of convergence of the algorithm. Replacing only a small fraction of the population at a time makes the GA convergence very slow. Setting this fraction too high tends to delete part of the pool of good solutions, making the algorithm

converge rapidly but to a sub-optimal result. Values between 0.5 and 0.7 have showed good results.

`MUTATION_PERCENT`

This parameter sets the percentage (i.e. a value between 0 and 1) of the offspring population on which genetic operators are applied after each generation. Higher values for this parameter increase the level of randomness of the candidate solutions created at each generation. The GA has difficulties approaching a steady optimum if the parameter is set too high. Based on empirical tests, this parameter starts to create undesired effects for values as low as 0.08. Setting it to very low values has the adverse effect of causing premature convergence to sub-optimal solutions.

`ELITISM_PERCENT`

Elitism sets the percentage of the population that will be automatically selected to be part of the next generation. This mechanism selects the fittest individual and skips the *roulette wheel selection* process to insert them, unaltered, into the new population. This value must be kept small to avoid premature convergence of the algorithm. Values below 0.03 and above 0.08 have shown to degrade the performance of the GA.

Appendix-D: Trajectory's sections and geometric equations

Using standard algebraic equations, a trajectory can be established between any two points using five types of section. Each section corresponds to a curve that can be described in 3-D by three parametric equations giving the X,Y and Z components of any point along that curve.. The geometry and underlying aircraft dynamics of these sections are now presented in details, complementing section 5.1. The standard Cartesian axes are used, where the X-Y plane represents the horizontal plane, and the Z axes relates to the altitude.

D.1. Horizontal turn section

This section describes a turn in the X-Y plane based on a circular trajectory with a constant speed and bank angle. The general configuration is shown in Figure 5.3. Any point $S(X, Y, Z)$ on this type of section can be found using the set of parametric equations 5. 1. The constants R_H, $\Delta\gamma$, $\gamma_{r,0}$ and (C_X, C_Y, C_Z) shown in these equations be found based geometric relations as well as standard aircraft dynamic equations.

According to the performance equations described in Appendix-B for a turn at constant altitude, the radius R_H of the turn is a function of the velocity and the bank angle of the aircraft:

$$R_H = V^2 / (g \tan(\gamma_l)) \qquad \text{D. 1}$$

with V, the aircraft velocity

g, the gravity constant

γ_l, the bank angle.

The bank angle is related to the load factor n by the following relation:

$$\gamma_l = \cos^{-1}(1/n) \qquad \text{D. 2}$$

The other variables of the equations set 5. 1 are $\Delta\gamma_r$, $\gamma_{r,0}$, and $C(x,y,z)$, which can be calculated based on the geometric relationships of the configuration shown in Figure 5.3 as given by the set of equations D. 3.

$$\overrightarrow{CP_{T1}} \cdot \overrightarrow{S_1} = \overrightarrow{CP_{T2}} \cdot \overrightarrow{S_2} = 0$$
$$\left\|\overrightarrow{CP_{T1}}\right\| = \left\|\overrightarrow{CP_{T2}}\right\| = R$$
$$\left\|\overrightarrow{P_{T1}P_2}\right\| = \left\|\overrightarrow{P_2P_{T2}}\right\| = R\tan\left(\frac{\Delta\gamma_r}{2}\right) \quad \text{with} \quad \begin{matrix}\overrightarrow{S_1} = \overrightarrow{P_1P_2} \\ \overrightarrow{S_2} = \overrightarrow{P_2P_3}\end{matrix} \qquad \text{D. 3}$$
$$\frac{\overrightarrow{S_1} \cdot \overrightarrow{S_2}}{\|S_1\|\|S_2\|} = \cos(\Delta\gamma)$$

Where the symbol " · " is the dot product.

Given P_1, P_2, P_3 and R_H, the remaining variables can be found as follows:

Angle of the turn

$$\Delta\gamma_r = \cos^{-1}\left(\frac{S_{1x}S_{2x} + S_{1y}S_{2y}}{\sqrt{S_{1x}^{\,2} + S_{1y}^{\,2}}\sqrt{S_{2x}^{\,2} + S_{2y}^{\,2}}}\right) \qquad \text{D. 4}$$

First tangent point

$$P_{T1y} = P_{2y} - \sqrt{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}} \quad with \begin{cases} a = S_{1x}^{\,2} + S_{1y}^{\,2} \\ b = -2RS_{1y}\sqrt{S_{1x}^{\,2} + S_{1y}^{\,2}}\tan(\Delta\gamma_r/2) \\ c = \left(RS_{1y}\tan(\Delta\gamma_r/2)\right)^2 \end{cases} \qquad \text{D. 5}$$

$$P_{T1x} = P_{2x} - \left(\frac{R\tan(\Delta\gamma_r/2)\sqrt{S_{1x}^{\,2} + S_{1y}^{\,2}} - (P_{2y} - P_{T1y})S_{1y}}{S_{1x}}\right)$$

Second tangent point

$$P_{T2y} = P_{2y} + \sqrt{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}} \quad with \quad \begin{cases} a = S_{2x}^{\,2} + S_{2y}^{\,2} \\ b = -2RS_{2y}\sqrt{S_{2x}^{\,2} + S_{2y}^{\,2}}\tan(\Delta\gamma_r/2) \\ c = \left(RS_{2y}\tan(\Delta\gamma_r/2)\right)^2 \end{cases} \qquad \text{D. 6}$$

$$P_{T2x} = P_{2x} + \left(\frac{R\tan(\Delta\gamma_r/2)\sqrt{S_{2x}^{\,2} + S_{2y}^{\,2}} - (P_{T2y} - P_{2y})S_{2y}}{S_{2x}}\right)$$

Center of turn

$$C_y = \frac{P_{T1x}S_{1x} + P_{T1y}S_{1y} - (P_{T2x}S_{2x} + P_{T2y}S_{2y})(S_{1x}/S_{2x})}{S_{1y} - S_{2y}(S_{1x}/S_{2x})}$$
$$C_x = \frac{P_{T1x}S_{1x} + P_{T1y}S_{1y} - C_yS_{1y}}{S_{1x}} \qquad \text{D. 7}$$

D.2. Vertical turn section

This section describes a turn in the vertical plane (i.e. a partial loop) based on a circular trajectory with a near-constant speed and a zero bank angle. The general configuration is shown in Figure 5.4. Assuming that this vertical plane is defined by two axes, the vertical Z axis and the horizontal Γ axis, any point $S(\Gamma, Z)$ on this type of section can be computed using the set of parametric equations 5. 2. The constants R_V, $\Delta\gamma_2$, $\gamma_{2,0}$ and $C(\Gamma,Z)$ of this set of equation can be found based on the geometric relations and the standard aircraft dynamic equations shown below.

According to the performance equations described in Appendix-B, the radius R_V of the turn is a function of the velocity, load factor and pitch angle of the aircraft:

$$R_V = \frac{V^2}{g(n-\cos(\gamma_2))} \qquad \text{D. 8}$$

with V, the aircraft velocity

g, the gravity constant

n, the load factor

γ_2, the pitch angle.

For relatively small pitch angles, the cosine factor will be much less than the load factor, allowing the following standard simplification (*Eshelby 2000*):

$$R_V = V^2 / (g \cdot (n-1)) \qquad \text{D. 9}$$

This assumption remains valid for the majority of sub-sonic aircrafts.

Given P_1, P_2, P_3 and knowing R_V, the remaining variables can be found as follows:

Angle of the turn

$$\Delta\gamma_2 = \cos^{-1}\left(\frac{S_{1\Gamma}S_{2\Gamma}+S_{1Z}S_{2Z}}{\sqrt{S_{1\Gamma}^{\ 2}+S_{1Z}^{\ 2}}\sqrt{S_{2\Gamma}^{\ 2}+S_{2Z}^{\ 2}}}\right) \quad \text{with} \quad \begin{matrix}\overrightarrow{S_1} = \overrightarrow{P_1P_2} \\ \overrightarrow{S_2} = \overrightarrow{P_2P_3}\end{matrix} \qquad \text{D. 10}$$

First tangent point

$$P_{T1Z} = P_{2Z} - \sqrt{\frac{-b\pm\sqrt{b^2-4ac}}{2a}} \; with \begin{cases} a = S_{1\Gamma}^{\ 2}+S_{1Z}^{\ 2} \\ b = -2RS_{1Z}\sqrt{S_{1\Gamma}^{\ 2}+S_{1Z}^{\ 2}}\tan(\Delta\gamma_2/2) \\ c = (RS_{1Z}\tan(\Delta\gamma_2/2))^2 \end{cases} \qquad \text{D. 11}$$

$$P_{T1\Gamma} = P_{2\Gamma} - \left(\frac{R\tan(\Delta\gamma_2/2)\sqrt{S_{1\Gamma}^{\ 2}+S_{1Z}^{\ 2}} - (P_{2Z}-P_{T1Z})S_{1Z}}{S_{1\Gamma}}\right)$$

Second tangent point

$$P_{T2Z} = P_{2Z} + \sqrt{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}} \quad with \quad \begin{cases} a = {S_{2\Gamma}}^2 + {S_{2Z}}^2 \\ b = -2\,R\,S_{2Z}\sqrt{{S_{2\Gamma}}^2 + {S_{2Z}}^2}\tan(\Delta\gamma_2/2) \\ c = (R\,S_{2Z}\tan(\Delta\gamma_2/2))^2 \end{cases} \qquad \text{D. 12}$$

$$P_{T2\Gamma} = P_{2\Gamma} + \left(\frac{R\tan(\Delta\gamma_2/2)\sqrt{{S_{2\Gamma}}^2 + {S_{2Z}}^2} - (P_{T2Z} - P_{2Z})S_{2Z}}{S_{2\Gamma}} \right)$$

Center of turn

$$C_Z = \frac{P_{T1\Gamma}\,S_{1\Gamma} + P_{T1Z}\,S_{1Z} - (P_{T2\Gamma}\,S_{2\Gamma} + P_{T2Z}\,S_{2Z})(S_{1\Gamma}/S_{2\Gamma})}{S_{1Z} - S_{2Z}\,(S_{1\Gamma}/S_{2\Gamma})}$$

$$C_\Gamma = \frac{P_{T1\Gamma}\,S_{1\Gamma} + P_{T1Z}\,S_{1Z} - C_Z\,S_{1Z}}{S_{1\Gamma}} \qquad \text{D. 13}$$

D.3. Log section

This section combines a turn in the vertical and horizontal planes. Figure 5.6 shows an example of a log section and the set of equations 5. 6 give its parametric representation. The parametric equation giving the Z-component of the points along a log section is based on two main assumptions which are presented here with the derivation of this equation. The first assumption consists of taking the rate of change of the pitch angle as constant along the section. Hence, the parametric equation of the pitch angle along the section becomes linear and can be written as follows:

$$\gamma_2 = v\,(\gamma_{2f} - \gamma_{2,0}) + \gamma_{2,0} = v\,\Delta\gamma_2 + \gamma_{2,0} \qquad \text{D. 14}$$

with $\gamma_{2,f}$, the final pitch angle

$\gamma_{2,0}$, the initial pitch angle

$\Delta\gamma_2$, the difference between the initial and final pitch angles (same as for the vertical turn section)

v, the parametric variable varying between 0 (the first point of the section) to 1 (the last point of the section)

The definition of the pitch angle γ_2 is given as:

$$\tan\gamma_2 = \frac{dz/dv}{R_H\Delta\gamma_r} \qquad \text{and thus} \qquad \tan\left[v\,\Delta\gamma_2 + \gamma_{2_0}\right] = \frac{dz/dv}{R_H\Delta\gamma_r}$$

The parametric expression of the Z-component with respect to v can then be found as:

$$\int_{Z_0}^{Z(v)} dz = R_H \, \Delta\gamma_r \int_0^v \tan\left[v\,\Delta\gamma_2 + \gamma_{2,0}\right] dv$$

which gives after integration:

$$Z(v) - Z_0 = \frac{R_H \, \Delta\gamma_r}{\Delta\gamma_2} \ln\left|\sec\left(v\,\Delta\gamma_2 + \gamma_{2,0}\right)\right| \Big|_0^v \qquad \text{D. 15}$$

The second assumption is that the aircraft is not doing any aerobatic maneuvers and thus, has its pitch angle bounded between [-π/2; π/2]. The logarithm is then computed for values varying only between [0; 1]. Thus, the absolute value of the logarithm in equation D. 15 can be dropped, and the resulting expression for Z is given as:

$$S_Z = \frac{R_H \Delta\gamma_r}{\Delta\gamma_2} \ln\left(\frac{\cos(\gamma_{2,0})}{\cos\left(v\,\Delta\gamma_2 + \gamma_{2,0}\right)}\right) + Z_0 \qquad \text{D. 16}$$

Equation D. 16 is the same as the third equation of the set of equations 5. 6 which defines the X, Y and Z components of the all the points along a log section. The variables R_H, $\Delta\gamma_r$, $\gamma_{r(C1)}$ and C_1 of the equations giving the X and Y components are calculated based on the equations for a standard horizontal turn with a load factor set to n_H. The variables $\gamma_{2,0}$ and Z_0 of the parametric equation giving the Z-component of the points along the log section are found based on the initial conditions of the section. The variable $\gamma_{2,0}$ is set to the pitch angle of the aircraft at the beginning of the section, i.e. the pitch angle of the last point of the preceding section, or the pitch of the starting position of the aircraft. The altitude Z_0 of the first point of the log section is also found using either the last point of the preceding section, or the initial position of the aircraft if the section is the first one of the trajectory. The remaining variable that needs to be found is $\Delta\gamma_2$.

Using the assumption (*Eshelby 2000*) regarding relatively small pitch angles ($\cos(\gamma_2) << n$), the variable $\Delta\gamma_2$, can be calculated as follows:

$$\begin{aligned} \dot{\gamma}_2 &= \frac{V}{R_V} \quad \text{with} \quad R_V = \frac{V^2}{g\left(n_V - \cos(\gamma_2)\right)} \cong \frac{V^2}{g \cdot \left(n_V - 1\right)} \\ \dot{\gamma}_2 &= \frac{\Delta\gamma_2}{\Delta t} = \frac{\Delta\gamma_2 \, V_{xy}}{R_H \, \Delta\gamma_r} \quad \text{with} \quad \begin{cases} V_{xy} = \sqrt{R_H \cdot g \cdot \tan(\gamma_1)} \\ \gamma_1 = \cos^{-1}(1/n_H) \end{cases} \end{aligned} \qquad \text{D. 17}$$

with V_{xy}, the near-constant horizontal component of the velocity.

Equations D. 17 leads to the following expression which can be solved to find the maximum $\Delta\gamma_2$ (i.e. the maximum variation in pitch) as a function of the horizontal component of the log section:

$$\Delta\gamma_2 = \frac{(n_V - 1)\,\Delta\gamma_r}{V}\sqrt{\frac{g \cdot R_H}{\tan(\gamma_1)}} \qquad \text{D. 18}$$

D.4. Tangent line segments

There are four possible tangents between the two circles corresponding to the horizontal turns of the pair of VoA configuration. These tangents are represented in Figure 5.16 by the segments connecting the pairs of points PT_{en} PT_{ex} tangent to the circles C1 and C2. The geometric relations describing this configuration is given as:

1. $\begin{cases} PT_{en,X} = R1\cos\theta_1 + C1_X \\ PT_{en,Y} = R1\sin\theta_1 + C1_Y \end{cases}$
2. $\begin{cases} PT_{ex,X} = R2\cos\theta_2 + C2_X \\ PT_{ex,Y} = R2\sin\theta_2 + C2_Y \end{cases}$
3. $\overrightarrow{PT_{en}PT_{ex}} \cdot \overrightarrow{C1\,PT_{en}} = 0$ (the two vectors being perpendicular)
4. $\overrightarrow{PT_{en}PT_{ex}} \cdot \overrightarrow{C2\,PT_{ex}} = 0$
5. $\theta_1 \equiv \theta_2$
6. $\theta_1' \equiv \pi + \theta_2'$

D. 19

The four sets of possible tangent can be found based on these relations:

- *Tangent points PT1 ($PT1_{en}$, $PT1_{ex}$):*

$$\theta_1 = \sin^{-1}\left(\frac{R1 - R2}{\left\|\overrightarrow{C1C2}\right\|}\right) - \tan^{-1}\left(\frac{C2_X - C1_X}{C2_Y - C1_Y}\right) \qquad \text{D. 20}$$

$$\begin{cases} PT1_{en,X} = R1\cos\theta_1 + C1_X \\ PT1_{en,Y} = R1\sin\theta_1 + C1_Y \end{cases} \quad \textit{and} \quad \begin{cases} PT1_{ex,X} = R2\cos\theta_1 + C2_X \\ PT1_{ex,Y} = R2\sin\theta_1 + C2_Y \end{cases}$$

- *Tangent points PT2 ($PT2_{en}$, $PT2_{ex}$):*

$$\begin{cases} \overrightarrow{C1PT2}_X = \overrightarrow{C1C2}_X \cdot\cos\theta_2 - \overrightarrow{C1C2}_Y \cdot\sin\theta_2 \\ \overrightarrow{C1PT2}_Y = \overrightarrow{C1C2}_Y \cdot\cos\theta_2 + \overrightarrow{C1C2}_X \cdot\sin\theta_2 \end{cases} \quad \textit{with} \quad \theta_2 = -\left(\overrightarrow{C1C2} \,\hat{\bullet}\, \overrightarrow{C1PT1}_{en}\right) \qquad \text{D. 21}$$

$$\Rightarrow PT2_{en} = \overrightarrow{C1PT2} \cdot \frac{R1}{\left\|\overrightarrow{C1C2}\right\|} + C1 \quad \textit{and} \quad PT2_{ex} = \overrightarrow{C1PT2} \cdot \frac{R2}{\left\|\overrightarrow{C1C2}\right\|} + C2$$

- *Tangent points PT3 ($PT3_{en}$, $PT3_{ex}$):*

$$\theta_3 = \sin^{-1}\left(\frac{R1+R2}{\left\|\overrightarrow{C1C2}\right\|}\right) - \tan^{-1}\left(\frac{C2_X - C1_X}{C2_Y - C1_Y}\right) \qquad \text{D. 22}$$

$$\begin{cases} PT3_{en,X} = R1\cos\theta_3 + C1_X \\ PT3_{en,Y} = R1\sin\theta_3 + C1_Y \end{cases} \quad \textit{and} \quad \begin{cases} PT3_{ex,X} = R2\cos(\theta_3+\pi) + C2_X \\ PT3_{ex,Y} = R2\sin(\theta_3+\pi) + C2_Y \end{cases}$$

- *Tangent points PT4 ($PT4_{en}$, $PT4_{ex}$):*

$$\begin{cases} \overrightarrow{C1PT4}_{en,X} = \overrightarrow{C1C2}_X \cdot\cos\theta_{4,en} - \overrightarrow{C1C2}_Y \cdot\sin\theta_{4,en} \\ \overrightarrow{C1PT4}_{en,Y} = \overrightarrow{C1C2}_Y \cdot\cos\theta_{4,en} + \overrightarrow{C1C2}_X \cdot\sin\theta_{4,en} \end{cases} \textit{ with } \theta_{4,en} = -\left(\overrightarrow{C1C2} \,\hat{\bullet}\, \overrightarrow{C1PT3_{en}}\right)$$

$$\begin{cases} \overrightarrow{C1PT4}_{ex,X} = \overrightarrow{C1C2}_X \cdot\cos\theta_{4,ex} - \overrightarrow{C1C2}_Y \cdot\sin\theta_{4,ex} \\ \overrightarrow{C1PT4}_{ex,Y} = \overrightarrow{C1C2}_Y \cdot\cos\theta_{4,ex} + \overrightarrow{C1C2}_X \cdot\sin\theta_{4,ex} \end{cases} \textit{ with } \theta_{4,ex} = -\left(\overrightarrow{C1C2} \,\hat{\bullet}\, \overrightarrow{C1PT3_{ex}}\right) \qquad \text{D. 23}$$

$$\Rightarrow PT4_{en} = \overrightarrow{C1PT4_{en}} \cdot \frac{R1}{\left\|\overrightarrow{C1C2}\right\|} + C1 \qquad \textit{and} \qquad PT4_{ex} = \overrightarrow{C1PT4_{ex}} \cdot \frac{R2}{\left\|\overrightarrow{C1C2}\right\|} + C2$$