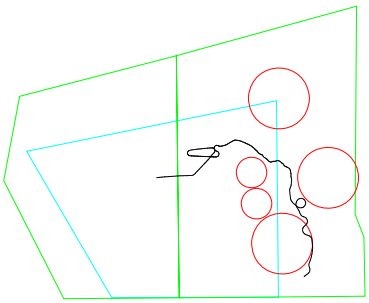
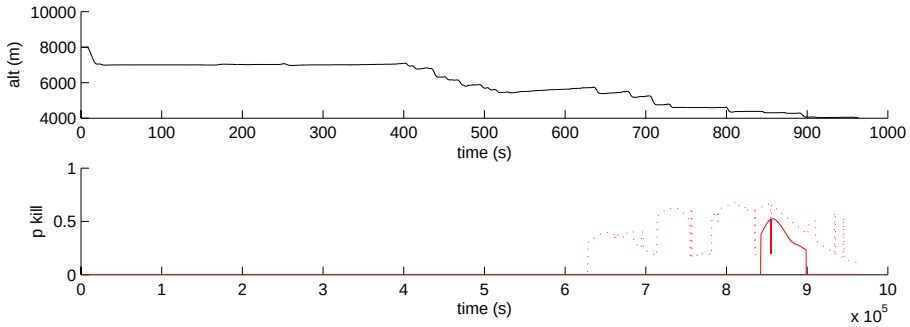
Path Planning and Autonomous Navigation for use in Computer Generated Forces

RICHARD HALL





FOI-R-- 2310 --SE
ISSN 1650-1942

Scientific report
June 2007

Combat Simulation, FLSC

Richard Hall

Path Planning and Autonomous Navigation for use in Computer Generated Forces

FOI-R--2310--SE Scientific report Combat Simulation, FLSC
ISSN 1650-1942 June 2007

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Combat Simulation, FLSC SE-164 90 Stockholm	FOI-R--2310--SE	Scientific report
	Research area code 2. Operational Research, Modelling and Simulation	
	Month year June 2007	**Project no.** E52303
	Sub area code 24 Air Combat Simulation Centre	
	Sub area code 2	

Author/s (editor/s)	Project manager
Richard Hall	Karina Waldemark
	Approved by Anders Borgvall
	Sponsoring agency
	Scientifically and technically responsible Örjan Ekeberg

Report title

Path Planning and Autonomous Navigation for use in Computer Generated Forces

Abstract

In this Master's thesis project, the use of Unmanned Combat Aerial Vehicle, UCAV, in the fighter role was studied. The mission of the agent was to guard and supervise a No-Fly Zone (NFZ) and to, upon violation, intercept a single intruder while minimizing the risk of being shot down. Methods for on-line path planning on a discretized version of the search space were evaluated and, in order to further enhance paths, post processing using virtual forces was applied.

Simulations showed that threat-constrained path planning, based on a tradeoff between path length and risk estimate, can be performed in satisfyingly short time, suitable for on-line use. Further considerations have yet to be made in order to meet the demands of an uncertain environment with dynamic threats and targets maneuvering in order to achieve diversion.

As part of the project, a prototype of the agent was implemented as an add-on to a fighter aircraft entity running in the Mission Training simulator at the Swedish Air Force Combat Simulation Centre (FLSC).

It is finally suggested that path planning procedures, like the ones studied in the project, should be an integrated part in future tools for Computer Generated Forces (CGF) as they could cover some of the shortcomings in pure rule-based systems.

Keywords

Path Planning, UAV, UCAV, Threat optimization, SAM, Graph search, Virtual forces, flight simulation

Further bibliographic information	Language English
ISSN 1650-1942	**Pages** 73 p.
	Price acc. to pricelist

Utgivare FOI - Totalförsvarets forskningsinstitut Avdelningen för stridssimulering, FLSC 164 90 Stockholm	**Rapportnummer, ISRN** FOI-R--2310--SE	**Klassificering** Vetenskaplig rapport
	Forskningsområde 2. Operationsanalys, modellering och simulering	
	Månad, år Juni 2007	**Projektnummer** E52303
	Delområde 24 Luftstridssimuleringscenter	
	Delområde 2	
Författare/redaktör Richard Hall	**Projektledare** Karina Waldemark	
	Godkänd av Anders Borgvall	
	Uppdragsgivare/kundbeteckning	
	Tekniskt och/eller vetenskapligt ansvarig Örjan Ekeberg	

Rapportens titel

Vägplanering och autonom navigering för datorgenererade stridskrafter

Sammanfattning

Detta examensarbete har studerat UCAV (Unmanned Combat Aerial Vehicle) i jaktrollen. Agentens uppdrag var att övervaka en No-Fly Zone (NFZ) genom att möta en inkräktare innan denna nått gränsen och samtidigt minimera risken att bli nedskjuten. Metoder för vägplanering och diskretisering av sökrymden har utvärderats och, som ett efterbehandlingssteg även, virtuella kraftfält.

Simuleringar visade att vägoptimering baserad på en avvägning mellan väglängd och riskskattning kan göras tillräckligt snabbt för att lämpa sig för planering under färd. Det återstår fortfarande att utforska situationer med manövrerande mål och dynamiska hot.

Som en del i projektet implementerades en prototyp av en UCAV som påbyggnad till en flygmodell i Flygvapnets Luftstridssimuleringscentrums (FLSC) simulatoranläggning.

I slutsatsen rekommenderas metodiken för vägplanering som ingående modul i verktyg för datorgenererade stridskrafter (CGF), för att täcka in områden som inte modelleras tillräckligt bra av befintliga, regelbaserade system.

Nyckelord

vägplanering, UAV, UCAV, hot, optimering, luftvärn, grafsökning, virtuella kraftfält, flygsimulering

Övriga bibliografiska uppgifter	**Språk** Engelska
ISSN 1650-1942	**Antal sidor:** 73 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Contents

List of Figures

List of Tables

Acronyms

AAW	Anti-Aircraft Warfare
ACO	Air Coordination Order
AFRL	Air Force Research Laboratory
AOR	Area Of Responsibility
ATO	Air Tasking Order
AWACS	Airborne Warning And Control System
BVR	Beyond Visual Range
C^2	Command & Control systems (e.g. AWACS)
C^3	Command, Control & Communication
C^4ISTAR	Command, Control, Communication & Computers + Intelligence, Surveillance, Target Acquisition & Reconnaissance
CAP	Combat Air Patrol
CGF	Computer Generated Forces
CONOPS	Concept of Operations
ELINT	Electronic Signals Intelligence
FLSC	Swedish Air Force Combat Simulation Centre (Swedish acronym)
FOI	Swedish Defence Research Agency (Swedish acronym)
GCI	Ground Controlled Intercept
HARM	High speed Anti Radiation Missile
HLA	High Level Architecture
M&S	Modeling and Simulation
MALE	Medium Altitude, Long Endurance
NBG	Nordic Battle Group
NFZ	No-Fly Zone
OTW	Out of The Window
PSO	Peace Support Operations
ROE	Rules Of Engagement
RTI	Run-Time Implementation
RWR	Radar Warning Receiver
SAM	Surface to Air Missile
SBA	Simulator Based Acquisition
SEAD	Suppression of Enemy Air Defense
SNNS	Stuttgart Neural Network Simulator
STOW	Synthetic Theater Of War
T3SIM	Training and Tactical/Technical Development Simulation System)
TACSI	TACtical Simulation
UAV	Unmanned Aerial Vehicle
UCAV	Unmanned Combat Aerial Vehicle
VID	Visual Identification
WVR	Within Visual Range

Preface

This report is the written record of a master's thesis project at the School of Computer Science and Communication at the Royal Institute of Technology (KTH). The project was carried out at the division of combat simulation, Swedish Air Force Combat Simulation Centre (FLSC), at the Swedish Defence Research Agency during the winter semester of 2006/2007.

I would like to express my sincerest thanks to my supervisors Dr. Karina Waldemark at FOI, FLSC and Dr. Örjan Ekeberg at CSC, KTH for guidance and support.

I thank all the coworkers at FLSC for support and for making my time there so memorable.

Thanks also to Martin Castor, Petter Ögren and all others who have helped me in my research.

Stockholm, May 2007
Richard Hall

Chapter 1

Introduction

Following the rapid evolution of computers, the abilities within computerized simulation has skyrocketed in the last decades, much thanks to the game industry which constantly raises demands on graphical hardware and software. Computerized simulations can be applied in a variety of disciplines, but in this report we will refer to the domain of combat flight simulation, where the purpose is mission training for military pilots. In this field, Computer Generated Forces (CGF's) have become an important factor in simulating the complex and scattered battlefields we see in current conflict zones.

CGF's can be described as entities in a simulation that are controlled by computer software. They take actions based on knowledge gathered from the simulated environment, often by mimicking human behavior, and their purpose is to create a higher resemblance of reality in the outcome of the simulation.

1.1 Background

During 2006 the Swedish Air Force performed a concept study regarding the future use of unmanned combat aircraft. The purpose was not primarily to evaluate technologies but rather to identify tasks where removing the human operator would result in a tactical advantage. Further, the command structure was emphasized and especially regarding the human operator that would be in control of one or several unmanned aircraft.

As part of the study, simulations were done during two weeks at the Swedish Air Force Combat Simulation Centre (FLSC). The first week concentrated on Unmanned Combat Aerial Vehicles (UCAV's) performing ground attack missions, while during the second week, a more protective role was considered where supervising a restricted airspace was set into focus. The scenario took place in a fictive Peace Support Operation (PSO) and the task to be solved was to perform Combat Air Patrol (CAP) over a No-Fly Zone (NFZ). This is a task that might need to be maintained over a long period of time, thereby becoming repetitive and fatiguing for human operators. The level of conflict was assumed to be low making large scale

violations to the NFZ unlikely.

Although simulations were successful, the UCAV model required much attention from the operator while intercepting an object that was on its way to violate the no-fly zone. One reasons for this was the way surface threats, e.g. Surface to Air Missiles (SAM's), were handled. The UCAV would simply turn away from threats it discovered, even if it conflicted with its intercept mission or put it on another potentially dangerous path. It was obvious that improvements could be done to make the model act more autonomously on information it already possessed and herein arose the problem that is dealt with in this master's thesis. A scenario similar to the one used in the second week of simulations together with the control software and operator interface were used as base.

1.2 Purpose and Goals

The purpose of this thesis is to evaluate what kind of capabilities are needed in Computer Generated Forces used for simulating unmanned aircraft. Focus is set on autonomous path planning which is an important issue commonly addressed in the domain of Unmanned Aerial Vehicles (UAV) and Unmanned Combat Aerial Vehicles (UCAV). The problem addressed is how the vehicle is to move itself from A to B, taking into consideration obstacles and threats. This type of problem is also often addressed in the field of autonomous systems and mobile robotics.

The goal of this thesis is to suggest methods how to automatize the behavior of a simulated unmanned aircraft on a mission where it is to guard and supervise a no-fly zone (NFZ). A NFZ is a restricted geographical area where no movements, except for allied forces, is allowed in the airspace. Should the zone be violated by someone, the guard is to intercept the intruder at the border or as close to the border as possible. The guard, however, should not expose itself to more threats than necessary. Hence, there is a need to optimize its trajectory towards the target. There should also be a higher level decision making that can determine whether a proposed trajectory is feasible or if there is need for replanning. Threats consist, in this scenario, of surface to air missiles (SAM) and hostile combat aircraft.

In the beginning of the project the following list of subgoals was created.

1. Identify what parameters to use for decision making and optimization
2. Evaluate how to best make use of planning algorithms
3. Develop a prototype running in FLSC
4. Make use of recent methods and research results

1.3 Scope and Limitations

This thesis project lies mainly within the scope of Autonomous agents and Computer Generated Forces, concentrating on the planning needed in an Unmanned Combat

Aerial Vehicle on a patrol mission.

Since CGF's generally try to model human behavior, the simulated UCAV that is referred to in this study, which is controlled by an operator, is not by definition a CGF. However, UAV's and CGF's are both to some extent autonomous agents that acts on inputs from the environment. The distinction between CGF and UAV is not always unambiguous. For instance, a CGF could receive new commands during an ongoing simulation, resulting in changed behavior as well as a UAV could be charged to operate completely autonomously during a mission.

The task that is dealt with is well restricted within the scenario. One UCAV is to set CAP over a NFZ and, when assigned an incoming target, plan and perform an interception, preferably before the target enters the NFZ. In this context interception has the meaning of position the UCAV within a certain distance of the intruder where further actions could be to visually identify and/or to deter.

One limitation that was made was to restrict the number of simultaneously incoming targets. This is because normally, to maintain a NFZ there is need for a large combined force, covering a wide range of capabilities, where a CAP mission would be conducted by a team of at least two combat aircrafts. To achieve coordinated actions within the scope of UCAV, one would have to widen the perspective and include areas as distributed target assignment and communication issues.

1.4 Outline of Thesis

In the next chapter, the task is presented and broken down. In chapter 3, a brief background to combat simulation and the the main topics covered in this report is presented. Chapter 4 explains the methods and techniques used for planning a path in order to intercept a given target. In chapter 5 the implementation phase of the UCAV prototype is described. In chapter 6, simulation results are discussed and finally, chapter 7 provides a discussion and conclusions part.

In appendix A, diagrams from the simulations are displayed and in in appendix B a background study provides the reader with references to related research and previous work.

1.5 Previous Work

Before the goals of this thesis were fully defined, a literature study was conducted in the fields of Combat Simulation, Computer Generated Forces, and Autonomous airborne systems (A summary is available in appendix B). During this study, a number of interesting publications were evaluated. Closely related to the thesis and worth pointing out here, is an article written for the preceding Air Force UCAV study (Ögren *et al.*, 2006), a FOI article about path planning and target assignment (Ögren and Winstrand, 2005), and an article proposing a virtual forces based approach to path planning (Bortoff, 2000).

Chapter 2

Task Break Down

As the projects first phase, a search and study of literature and research was initiated. The background study aimed at finding feasible methods and to reuse work already done in the field.

The first approach to be evaluated was Neural Networks. The net was supposed to output control commands given input on target and threats. Similar work had been done, (Rosander and Walther, 1996; Ehlert *et al.*, 2003), but very little research was found that included planning. It was decided that the task should incorporate planning and the neural network approach was then dropped since it was deemed that it would not provide the traceability that was requested. Once trained, a NN is too much of a black box in some of these respects.

Next, rule-based systems were studied. Much of the CGF research is done using these systems, e.g. Soar and TACSI to which references can be found in appendix B, which do produce a traceable behavior. However, no material was found on using rule-based systems together with planning so, for that reason it was decided to look closer into this area. The hypothesis being that it could possibly be integrated into a CGF system.

When the task was this much defined it was time to look into the howto's. Path planning algorithms and threat representations had to be evaluated. Probabilistic Path Planning and Randomly exploring Rapid Trees (RRT's) were both considered as planning algorithms but since much reference work on UAV path planning was using Voronoi graphs, this method was chosen. Had there been time, a RRT planner would also have been evaluated.

To complete the planner, a graph search algorithm had to be selected. Since the A* heuristic search is somewhat of a standard in graph search, this method was tried first. It worked out well and could also be proven optimal.

2.1 Execution Phase

After most of the background research was completed, a phase dedicated to tool and method evaluation took place. The task was broken down in the following sub

tasks so that some of them could be evaluated separately.

- Find a representation of the search space
- Find representation/estimation of threats
- Find the proper search algorithm
- Translate output into flyable trajectories
- Evaluate output

Following the method evaluation, implementation and testing of the entire system was begun. The project was ended with evaluation and reporting.

The original project week plan is displayed in figure 2.1. The ordering was roughly followed although more tasks were done in parallel than indicated in the figure. The implementation and the testing phases turned out much more time consuming than estimated from the beginning.

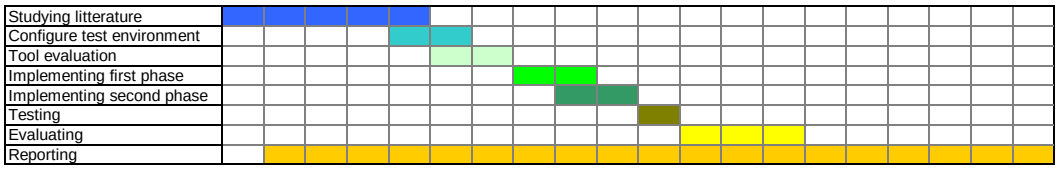


Figure 2.1: Rough GANTT week chart

Chapter 3

Combat Simulation

> "Once the command of the air is obtained by one of the contended armies, the war must become a conflict between a seeing host and one that is blind."
>
> H. G. Wells

Humans are valuable assets today and few military powers would deploy troops that have not received adequate training. Performance in the field is strongly correlated to the amount and quality of the training and since the development has resulted in more and more advanced technical systems, more skills are needed in the operators and this means more training. Combat simulation ranges from live exercises with humans playing all roles, to pure virtual reality where computer generated forces control all participants. The purpose is multifold, e.g. training and validating forces, mission rehearsal, developing new tactics and methodology or performing Simulation Based Acquisition (SBA).

Although Wells had no insight in network centric warfare and these days importance of electronic sensors, the quote above reflects the purpose of air power very well. In military context, the term Dominant Battlefield Awareness (DBA) is used to point out the need for information superiority which, due to the wast technological development of military forces, has gotten the meaning of adequately filtered information rather than complete information, which is more likely to be synonym with information overflow (Owens and Offley, 2000). In the modern aerial arena there is a great number of participants, allied from different nations, enemy and non combatants, all moving at high speeds making the operational theater large and complex. There is a great dependence on electronic sensor information gathering and exchange which has led to the simultaneous development of electromagnetic warfare and countermeasures. All in all, to be superior in the battlefield it is necessary to have the right information but never the less to know how to act upon it. This is why extensive training and high fidelity simulation is so important in lifting the "fog of war" on the battlefield.

3.1 Peace Support Operations

> "On peace support operations, the use of military force is normally controlled by restrictive rules of engagement. The purpose is to limit freedom of action on lower levels in order to achieve maximal control and freedom of action on a political level. This is a point of view that is particularly apparent in large scale international operations."
>
> (Försvarsmakten, 2005)

Over the last decades we have seen an increased political will to use our military forces in coalition with other nations on international peace support operations proceeding UN resolutions. This is closely related to the new world order, resulting from the fall of the east block, where threat assessments points at a lesser need for the large national defense. The reorganization of our military forces have many different meanings wherein flexibility seems to be a key concept.

In an international perspective, aerial forces are often a component in peace keeping missions, contributing with a large range of action, high speed and maneuverability. To protect ground and maritime forces, it is crucial to achieve air superiority, which means that joint forces have obtained full control of the airspace. However, in order to guarantee air superiority one needs to stay in the area and supervise, a task that is typically solved by proclaiming a no-fly zone. A NFZ needs to be maintained 24 hours a day, 7 days a week, possibly during long periods of time[1], performing repetitive and dull tasks. These kinds of missions, that have the potential of wearing out a human operator, are put in focus in this thesis because it is a field where automation have proven very successful, with respect to UAV systems operative today, and where autonomous aircraft will probably play an even bigger role in the near future.

3.2 Flight Simulation

For many years, flight simulators have been employed in the training of both commercial and military pilots. In the beginning, pneumatical/mechanical models were used, with very good results, providing the user with immersion to some extent even without a graphical display. Because of its wide range of benefits, e.g. in improving pilot skills and reducing costs, flight simulation is a research area which has contributed and, to a large extent, driven the VR development. Today commercial flight simulators involve features from most niches of Virtual Reality i.e. visual, audio and haptics.

Immersion in flight simulation is achieved by providing the user with a mixture of real hardware and virtual feedback. The pilot is in most cases constrained to a chair

[1]The NFZ that was proclaimed over Iraq after the gulf war of 1991 lasted for over a decade (Wikipedia, 2007a)

Figure 3.1: Simulator pilot station at FLSC

surrounded by real buttons and controls. In some simulators, as in FLSC, even parts of the instrumentation can be virtual in the sense that it is shown on interactive displays, which makes it easy to change their appearance and functionality. The visual feedback is given to the pilot by projections of the environment outside the aircraft but also by virtual and real instruments inside. Audio is used for simulating radio traffic and warning signals e.g. missile alert or ground collision warning. In some simulators the haptic feedback is provided through pneumatical actuators changing the orientation of the simulator. It is thereby possible to simulate the pitch and roll that the pilot would experience in a real maneuver. Acceleration can be simulated by tilting the simulator backward and deceleration by leaning it forward. Although this method can be sufficient in commercial aviation, heavy acceleration and rapid maneuvers are more difficult to achieve and that is sometimes needed in military simulations. Haptic feedback can also be given by force feedback in the rudder controls. Since many aircrafts today make use of fly-by-wire systems, force feedback is implemented similarly in the simulator as in a real aircraft.

Simulators are widely used in both military and civilian aviation but there are some general differences that needs pointing out. Commercial aircraft simulators are constructed mainly to train pilots in order to increase flight safety. Therefore the simulator must be able to put the pilot in critical situations, that are either

too dangerous or too hard to recreate in plain air. In military applications, there is also a need for training coordination while flying in a formation. Therefore it becomes much more important with network capacities and real-time performance. Every pilot must have his view of the environment presented at the same time as the others get theirs. This form of VR is called Distributed Virtual Reality. Besides the real-time aspect military simulators also has to deal with weapon systems and hostile entities which can be autonomous, then called Computer Generated Forces (CGF) which will be explained next.

3.3 Computer Generated Forces

CGF's have been of interest to military forces for quite some time due to a high potential in e.g. training, development and acquisition. In brief they make use of AI techniques to model and simulate military units ranging from large and complex vehicle systems down to the individual soldier. In many applications even group behavior and formations are considered.

Thanks to the computer evolution, simulation has gained a higher level of fidelity and the capabilities of real-time computations has largely increased. Further, VR applications makes it possible for humans to achieve a deeper level of immersion and gives them a better interface to interact with the simulation. At the same time complexity and scope of tasks is increasing, rendering a constantly growing set of possibilities.

When CGF's are discussed in this report there is a focus on aircraft agents and surface-to-air missile platforms (SAM's). In appendix B some existing software for handling CGF's are presented.

3.4 UAV and UCAV

> "Today, there is a lack in the development of CONOPS (Concept of Operations) regarding the possible use of UCAV, individually and in cooperation with existing systems. A more evolved development of CONOPS would bring more evolved demands on technique, thus making the future more predictable"
>
> (Försvarsmakten, 2006)

The concept of Unmanned Aerial Vehicles is in no way a novelty. Following World War I, experiments with "aerial torpedoes" were conducted which was sort of predecessor to the cruise missile. It is however within the last two decades that we have seen a technological break-through which has resulted in the deployment of a number of intelligent and reusable systems. An evident example of this is the development of mini-UAV's that are the result of electronic components getting smaller, lighter and less expensive.

In recent years, interest has also grown regarding Unmanned Combat aerial Vehicles to overtake tasks from human pilots. UCAV's distinguish themselves from UAV's mainly by the capability of carrying arms. Today there are already such systems in service, e.g. the MQ-1 Predator which carries two Air-to-ground Hellfire missiles. The predator MALE system (Medium Altitude, Long Endurance) was however primarily built for surveillance tasks and the UCAV-term is mostly associated with aircraft that are developed for performing similar to a manned jet fighter. The main advantages with these kinds of systems is that they will be able to perform missions that, for a human, could be considered dirty, dull and/or dangerous.

In the picture (figure 3.2), you see an illustration of Neuron, a joint European project which aims at flying with a UCAV demonstrator in 2010.

Figure 3.2: nEUROn, © Dassault Aviation - A. Ernoult

3.5 Simulating Fixed-wing Aircraft

Fixed-wing aircraft are, as the rest of the world, restricted by the laws of motion. A propulsion unit provides a forward force which in turn, through Bernoulli's principle, creates a lift that opposes the force of gravity. Once in the air, the aircraft can make changes to a number of control surfaces which will change its current course.

When planning and executing a simulated mission it is important to know the dynamics of the model. Unlike rotory-wing, airplanes need to be in motion in order to maintain lift and this affects the minimum possible speed aswell as the angle of attack. If we for instance request a long path to be travelled at a slow speed we will force the aircraft to enter a high angle of attack, wasting a lot of fuel just to stay in

the air. The economic alternative would then be to stay at high altitude, and hold until the target can be reached at cruise speed.

3.5.1 Dynamic Model

The simplest model of an aircraft moves forward, and maneuvers within the three rotational degrees of freedom, i.e. pitch, roll and yaw. These reflect the main effect from changing the rudder, elevator and ailerons on a traditional aircraft and together with thrust they constitutes the primary flight controls or actuators.

Figure 3.3: The three rotational degrees of freedom in an aircraft. This image is licensed under the GNU Free Documentation License, `http://en.wikipedia.org/wiki/Image:Flight_dynamics.jpg`

Even if this model is a sufficient, although coarse, for fixed-wing aircraft, all high fidelity simulators make use of a model with six degrees of freedom that also incorporates the three translational degrees of freedom. This way, it is possible to model the attitude of the aircraft relative to its forward motion. In fact it is the attitude that determines the lifting force generated by the wings but also the drag component.

Based on the shape of the aircraft, the aerodynamic force contribution is calculated and added to the system of equations that make up the equations of motion that apply to the vessel. The system is then solved and the state updated with a certain frequency.

3.5.2 Interoceptive Sensors

When referring to sensors on an autonomous vehicle, there is a distinction between interoceptive and exteroceptive. The interoceptive constitutes all sensors that in some way measures the current state of the craft. There is a big number sensors of this kind in a modern aircraft and they range all from velocity, altitude and positional meters to rudder positions and engine status. Since regulation of the aircraft, in the simulation is performed based on sensor data, these values has to be derived from the state equations.

3.5.3 Exteroceptive Sensors

Exteroceptive sensors are the ones that measures everything that has to do with the environment outside the aircraft. For instance it gathers knowledge on vessels in its surroundings by using a RADAR or finds the heat signature of another aircraft with a Forward Looking Infra Red camera (FLIR).

On-Board RADAR

A RADAR onboard a modern combat aircraft has capability to position targets in three dimensions. With a Doppler RADAR it is also possible to measure the velocity of an object. By changing mode, the RADAR can use different search patterns, i.e. to intensify search in a certain sector or update a specified target more often. The RADAR is the primary sensor of a combat aircraft and loss of it would result in blinding, which is why electronic warfare and jamming devices have been developed hand in hand with the RADAR.

Data Link

Since most missions in the air arena are conducted by combined forces with different functionality, there is a need to communicate within the group to control and exchange information. Either the communication is vocal over radio, or it is data packages sent over a radio link. For instance, an airborne RADAR, like the E-3 Sentry, can scout a very wide area, linking targets to fighters in the force which then can shut down their own RADAR, thus not reveling themselves to foes.

Radar Warning Receiver

A radar warning receiver (RWR) is a passive device present in most fighters for protective reasons. The main purpose is to detect active sensors in the environment that could possibly constitute a threat. Many different data can be gathered from a radar signal e.g. frequency, signal strength (amplitude), pulse repetition frequency (PRF) and bearing. When the signal has been recorded it is matched against an emitter library in order to determine the type of the source. RWR's could for instance warn for radar lock-on, radar guided missiles or approaching SAM sites. Since the radar beam has to bounce off its target and return to the emitter, thus traveling the double distance, it is often much easier to passively detect the emitter than for the emitter to detect a target. The rule of thumb is therefore that the emitter can be revealed on two times the detection range (Siggelin, 2006).

When receiving signals from other airborne platforms, it is often only possible to position them with bearing, even if the signal strength gives us some boundaries regarding the distance. However, when measuring stationary emitters, such as SAM sites, we have the ability to use several measurements, separated by time and position, to triangulate the position of the emitter. In figure 3.4 two triangulation cases are displayed which pinpoints the role of uncertainty propagation when measuring

from different angles. In reality, the positioning is done with far more than two measurements in order to increase precision.

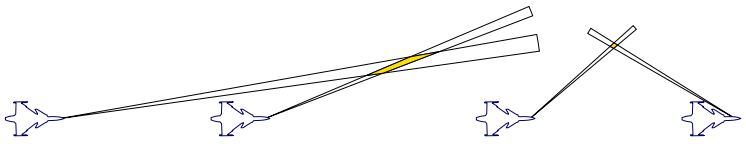


Figure 3.4: In the first case the uncertainty region (yellow area) is larger due to the relative angle of the observed source. The best estimation is made when measurements are perpendicular. The uncertainty for each measurement is here illustrated by a 2°cone

Chapter 4

Target Intercept

A typical task for a combat aircraft on a PSO mission is to intercept any vessel inside the Area of Responsibility (AOR) moving toward the No-Fly Zone. By intercept, we mean to position ourselves at a position where we have the tactical advantage. This is typically in a sector behind the target where we see him with our sensor and close enough so that he is inside the effective range of our weapons (Lavén, 2006). The intercept could have the purpose to identify the vessel (VID), to deter it from advancing any further, and/or to, ultimately, engage it in combat. What can be done is strictly specified in the rules of engagement (ROE) for the operation and these are in this scenario left for an operator to interpret.

4.1 Chasing a Dynamic Target

In order to acknowledge a new target, we need to make a detection with at least one of the available sensors, in our case the on-board radar or the airborne AWACS radar. The detection, or radar plot, specifies a location in three dimensions with some associated uncertainty. However, to make estimations on where the plot will be after a certain amount of time we need to compare consecutive detections in a process called tracking.

A tracker merges noisy measurements with estimates from previous updates so that we constantly get an as accurate as possible state information on the objects we are tracking. One of the most common tracking filters is the Kalman filter, (Becker *et al.*, 2007) which make use of a linear model of motion. More advanced trackers use the extended Kalman filter (EKF) with nonlinear models or a particle filter.

4.1.1 Estimated Rendezvous

If we assume that we have obtained position data and a velocity vector on the specified target, we can also estimate where and when it will enter our NFZ polygon. We start by iterating the segments of the NFZ to find their intersection point, if existing, with the target velocity vector.

If an edge is given by a start point and a unit direction vector, $\vec{e}_{pos}$ and $\vec{e}_{dir}$, and the target is given by $\vec{t}_{pos}$ and $\vec{t}_{dir}$, we can state the following equation:

$$\vec{e}_{pos} + k\vec{e}_{dir} = \vec{t}_{pos} + l\vec{t}_{dir} \tag{4.1}$$

where k and l are coefficients corresponding to the length of the direction vectors. Next we use the cross product with $\vec{t}_{dir}$ to get rid of the l-term.

$$k\vec{e}_{dir} \times \vec{t}_{dir} = (\vec{t}_{pos} - \vec{e}_{pos}) \times \vec{t}_{dir} + l\vec{t}_{dir} \times \vec{t}_{dir} \tag{4.2}$$

which, in 2D, can be simplified to

$$k = \frac{(\vec{t}_{pos} - \vec{e}_{pos}) \times \vec{t}_{dir}}{\vec{e}_{dir} \times \vec{t}_{dir}} = \frac{(\vec{t}_{pos} - \vec{e}_{pos})\mathbf{R}_{-\pi/2}\vec{t}_{dir}}{\vec{e}_{dir}\mathbf{R}_{-\pi/2}\vec{t}_{dir}} \tag{4.3}$$

where $\mathbf{R}$ is the 2D rotation matrix,

$$\mathbf{R}_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

If the divisor of equation (4.3) is zero, the lines are parallel and we will not find any intersection. Further, for all NFZ segments, we must check whether k is within the length of that segment. If it is, we can calculate l from (4.2) similarly and thus find the closest segment that intersects with the target vector.

Since we do not want the target to reach the border of the NFZ, we add an offset vector to the intersection point in the direction toward the target. In order not to approach the target straight forward but from the side, we also add an offset perpendicular to the target velocity vector.

Target data is updated with a certain frequency (depending on the sensor) and since the above linear equations have negligible time complexity, O(nr NFZ edges), the same update rate can be used for calculating the estimated point of intercept, i.e. the position where we would like to position ourselves when the target is at a predefined distance from the NFZ.

4.2 Finding the Shortest Path

When operating over a large surface it becomes important to maintain a high mobility in order to respond to actions in the area. It can also be important to reach a certain position within an exact time frame, which is fairly easy if we can determine the exact path distance. Although aircraft have the opportunity of moving along the straight line of sight, obstacles as mountains, bad weather or threats might force them to take a detour. Pathfinding is thus done in order to find the shortest path that best respects all constraints. There are hard constraints, like obstacles or the earth surface, that we are forbidden to break but there are also soft constraints, like threats that can be modeled as continuous functions. For these constraints, which will be considered in this report, we will have to minimize the exposure.

4.2.1 Threats in the Environment

When operating in an uncertain environment, it seems reasonable to weigh the risk of performing a task against the achievable effect. To do so, we need to gather information on where the threats are located and how they can act in order to prevent us from accomplish our task.

When it comes to surface threats, SAM sites (Surface-to-Air-Missiles), it is known that many such systems use a radar to detect targets, although exceptions do exist. The radar is an emitter of electromagnetic waves and therefore, anyone who is scanning the right frequency can gather information from it. Systems capable of doing this, often referred to as Electronic Signals Intelligence (ELINT), compares the received signal against a signal library in order to determine the type of the emitter. In joint aerial operations, ELINT is typically performed by dedicated platforms, such as the EA-6B Prowler, but in the scope of this thesis, the signal intelligence that is considered is performed with an on-board Radar Warner Receiver (section 3.5.3).

4.3 Discretization

In order to search the complex environment of the aircraft efficiently for an optimal path, the continuous reality must first be modeled as a discrete, finite set of points or cells. In mobile robotics, a common method for discretizing is to use occupancy grids wherein the environment is mapped onto an, often 2 dimensional, grid of cells. The method is well suited for keeping track of a static environment in which the robot is collecting data from exteroceptive sensors. A drawback with the grid-based approach is the amount of storage and and processing capacity needed for a grid with high resolution and this is especially a problem with UCAV, where the operational range is very large, rendering a huge grid. References on occupancy grids are found in (LaValle, 2006) and (Becker *et al.*, 2007).

4.3.1 Voronoi Tessellation

A common approach to discretization within the field of UAV path planning is to subdivide the search space into a Voronoi graph, references can be found in (Beard *et al.*, 2002; Ögren and Winstrand, 2005; Winstrand, 2004). The Voronoi graph is built up by regions. These regions, or cells, are created around a set of points so that each region contains only one point and all other points within that region are closer to their generating point than any other point. Together, these regions create a Voronoi tessellation and such a tessellation can be found for any discrete set of points in n-space. The regions are convex but not necessarily bounded (closed). In this context we will consider a Voronoi tessellation in the plane, where the regions can be called polygons.

A walk along any border of a polygon is guaranteed to maintain an equal distance to the two closest generating points. The nodes of the Voronoi graph are found at

all positions that lies on the border of three adjacent polygons. The borders are edges in the graph that connect the nodes. In figure 4.1 and 4.2 a Voronoi Diagram is created, using points in $\mathbb{R}^2$. The points represent SAM sites, UCAV position and target position.

The geometric dual to the Voronoi graph is the Delaunay triangulation. It connects the generating points of a Voronoi Diagram whose polygons share a common border. Alternatively the triangulation can be defined as a connection of all three points whose circumcircle does not include any other point.

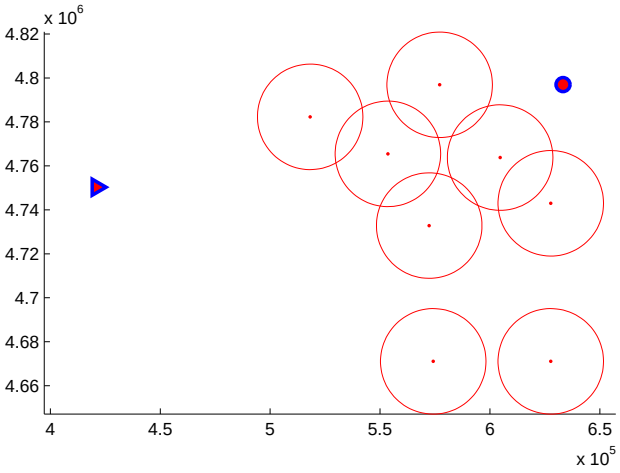


Figure 4.1: plot of UCAV (triangle), target (filled circle) and SAM sites (red dots with circular ranges.)

4.4 Graph Search

There are many different methods of searching a graph and the efficiency is largely dependent upon the size and shape of the graph. We can divide the different methods into the two major sub categories *breadth-first*, which stride to explore all edges of a node before descending, and *depth-first*, which strides to descend as far as possible before exploring the breadth. Of these two algorithms, only breadth-first is complete in the sense that it is guaranteed to find the shortest path to the goal, if existent, since it explores all paths of a certain length in increasing order. Depth-first on the other hand may be a lot faster but it suffers from the risk of getting caught in circles. Assuming depth-first does not expand previously visited nodes, the worst case time complexity for both methods is exponential, $O(b^n)$, where b is the maximum branching factor (number of successors to a node) and n the maximum path length.

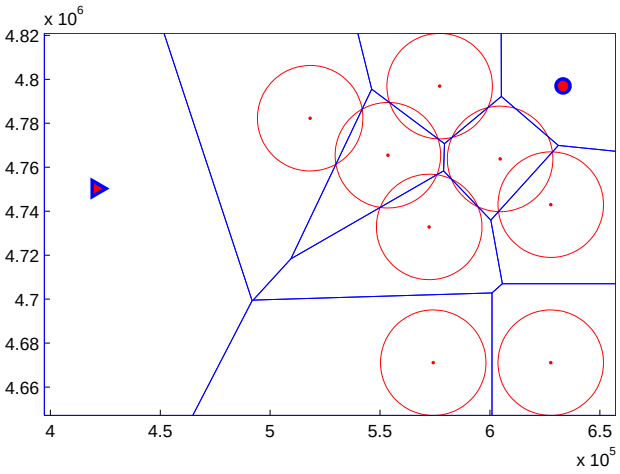


Figure 4.2: Voronoi tessellation constrained by an outer, rectangular, bound. Most lines that intersect with the bound would otherwise continue into infinity.

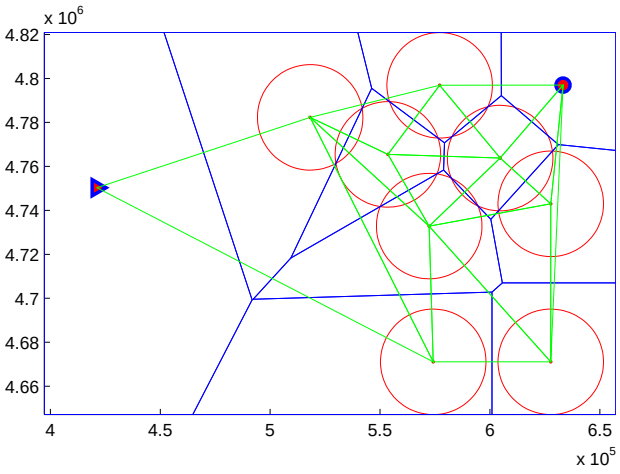


Figure 4.3: In this picture, the Delaunay triangulation is added to the graph.

There are many specializations of these methods, designed to achieve speedup and to reduce the storage space. A very well known one is named by its inventor, the Dutchman Edsger Dijkstra (Dijkstra, 1959). Dijkstra's algorithm is based on breadth-first but with the difference that it expands the most promising node first.

This approach requires that there exists an order of preference amongst the paths, i.e. path length (a cost associated with each edge). A generalization of Dijkstra that make use of a heuristic to enhance search is introduced next.

4.4.1 A* search

With A* a heuristic function is introduced which can estimate the cost from any given node to the goal node. The heuristic cost, $h(n)$, of a node, n, is added to the actual cost of getting to that node, $g(n)$, to form a priority function, $f(n) = g(n) + h(n)$.

Beginning at the start node, the priority function is explored for all of the neighbors. The nodes are then stored in a priority queue by their f-values so that the most promising node can be explored in each iteration. The A* algorithm is presented in pseudo code below.

The main benefit with A* is that it is optimal given that the heuristic function never overestimates the cost of reaching the goal.

$$h(n') \leq g(n') - g(n) + h(n) : \qquad n' \text{ succeeds } n \tag{4.4}$$

Even if an optimal solution is feasible to find for a limited graph, we might face the problem of exponential growth. The following criteria needs to be fulfilled in order to guarantee polynomial growth:

$$|h(n) - h^*(n)| \leq O(\log(h^*(n))) \tag{4.5}$$

where $h^*(n)$ is the exact optimal heuristic from n to the goal. (Russell and Norvig, 2003)

4.4.2 Cost Function

An aircraft that is flying through uncertain territory is always facing the risk of being shot down. Hostile units could be in possession of missiles or ballistic weapons fired from ground based or aerial platforms. Detection is not possible for all of these threats and there is also the risk of electronic warfare which can limit our awareness or ultimately, cause loss of control by electromagnetic pulse weapon (EMP). Since most of these threats are nondeterministic, man-in-the-loop systems, estimating risks with little uncertainty is not easy, if even possible. Therefor, in this report, estimations have been done roughly from a worst-case scenario based on our present knowledge of missile sites.

A straight forward approach is to define the risk of flying along an edge from point A to point B as the probability of being killed, p_k, while doing it. The probability of survival, p_s, is then equal to $1-p_k$. We can assume that the risk propagates, so if the edge is divided into a number of waypoints, the conditional probability of survival at one point is dependent on the previous survival probabilities (Ögren and Winstrand, 2005).

Algorithm 1 A* Search

```
1: priority queue Q
2: closed set C
3: Node s ⇐ startnode
4: Node t ⇐ targetnode
5: add s to Q
6: while Q not empty do
7:    n ⇐ remove node with lowest f(n) from Q
8:    if n = t then
9:       return success
10:   end if
11:   for n' in successors(n) do
12:      f' ⇐ g(n') + h(n') = g(n) + cost(n, n') + h(n')
13:      if n' is unseen or in C or Q with f(n') > f' then
14:         place/promote n' on PQ with f(n') ⇐ f'
15:      end if
16:   end for
17:   add n to C
18: end while
19: return failure
```

Let S_i denote the event that the UCAV survives at position i along the edge. Then $p(S_i) = p(S_i|S_{i-1}) = p(S_i)p(S_{i-1}) \cdots p(S_1)$. If we choose to look at N points on the edge, the overall probability of survival is estimated by $p_s = p(S_N)$.

The instantaneous risk at any position along the edge is a weighted function of several variables, i.e. altitude, velocity, nearby threats and distances to them. To accurately estimate the risk, we would have to gather statistical data from as many different situations as possible, but even this would be hard due to the high dimensionality. However, for our mission it is sufficient if we could obtain a risk estimate that roughly returns a higher value for a dangerous situation than for a safe one even if it does not reflect the actual risk of failure.

Taking as an example a SAM site, the risk can be divided into two portions. First there is the probability that a missile is fired which is likely to be dependent on the distance to the SAM site, i.e. if we are flying within its effective range. Secondly, there is the probability of hit given that a missile was fired. Since missiles are not capable of maintaining maximum speed throughout their whole trajectory, this probability is dependent on with what speed the target is moving from or toward the launcher. Regarding the launch probability, we disregard the human decision factor and assume that the SAM site will fire a missile with zero probability when the UCAV is just outside the range. The probability will then smoothly step up to a 0.5 probability when we get inside the range, and continue increasing until it is close to one at about half the range. Due to mechanical limitations in most launch ramps the missile cannot be fired when the target is too close and therefore we have

a smooth transition to zero when approaching the launch site. The probability function, figure 4.4, is realized with three smooth functions, equation (4.6). In reality there is also a fourth step function that sets the probability to zero when the maximum altitude is passed, but this is not displayed in the picture.

$$p_{fire} = (1 - f_{step}(r, d, 5/r))(1 - f_{step}(r, d, 1))(1 - \exp^{-10\frac{d}{r}}) \tag{4.6}$$

where d is the distance, r is the SAM range and f_{step} is the sigmoid step function:

$$f_{step}(x_{mean}, x, \sigma) = \frac{1}{1 + e^{-(x - x_{mean})\sigma}} \tag{4.7}$$

where σ is a parameter that determines the steepness of the step.

The probability of hit, is based on whether we are approaching or escaping the SAM site. Simply put, we assume there is a greater risk when we are decreasing the distance than when we try to escape. A simple way to model this is by measuring the angle, ϕ, between the velocity vector of the UCAV and the vector pointing at the SAM site. The risk is then expressed as a cosine function

$$p_{hit} = \frac{a}{2}\cos(\phi) + \frac{1}{2} \tag{4.8}$$

where a is a scaling factor between 0 and 1.

The risk at a specified position can now be expressed as $p_k = p_{fire} \cdot p_{hit}$.

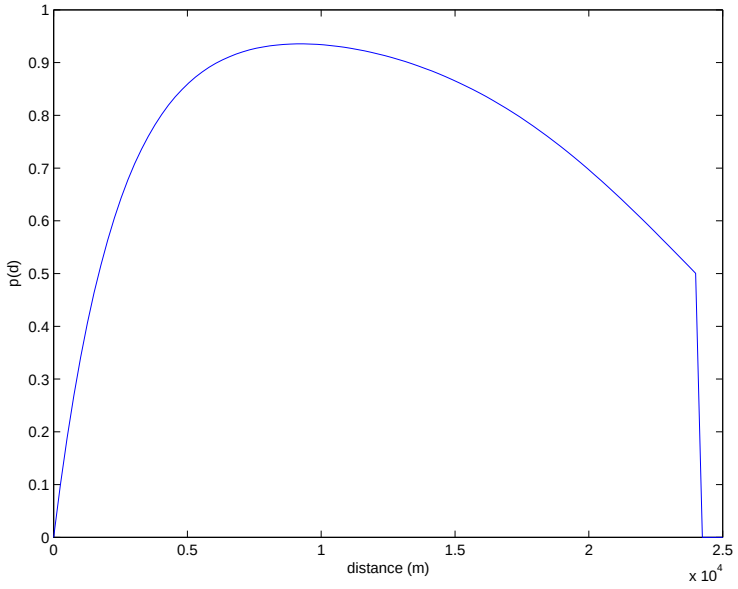


Figure 4.4: Probability function for SAM fire, p_{fire}, when target is at a certain distance.

4.5 Post-processing

A path, or a set of waypoints, generated from a discrete Voronoi graph might be pretty coarse and possibly make unnecessary long detours around threats. Angles between path segments may also be steep and difficult to follow since the UCAV is restricted by its maximum g-force restricted turning radius. It is therefore desired to optimize the path by making shortcuts where possible and by smoothing steep turns in the path.

4.5.1 Virtual Forces

One method of smoothing a coarse path is provided by Virtual Forces, references can be found in (Bortoff, 2000). In this approach, waypoints are considered to be masses in a mechanical system connected to each other with springs and dampers. Since most edges in the original path are long, it is preferable to first place additional waypoints evenly along the path so that no two neighboring waypoints exceed a predefined distance. The start and target positions are fix and hence the springs exert a contracting force that pulls the masses toward each other. Each mass, except for the start and the target, is affected by two springs whose forces are proportional to their lengths. Hence the spring force on a mass is equal to:

$$F_{spring} = \kappa(x_{j-1} - x_j) + \kappa(x_{j+1} - x_j) \tag{4.9}$$

where κ is a spring coefficient and j is the number of the current mass. The dampers are there to prevent the system from ending up in self induced oscillations. A simple damper can be created by adding a derivative force,

$$F_{damp} = b\ddot{x} \tag{4.10}$$

where b is a damping coefficient. Without further forces involved this system would have a steady-state solution that places the masses on a straight line between start and target.

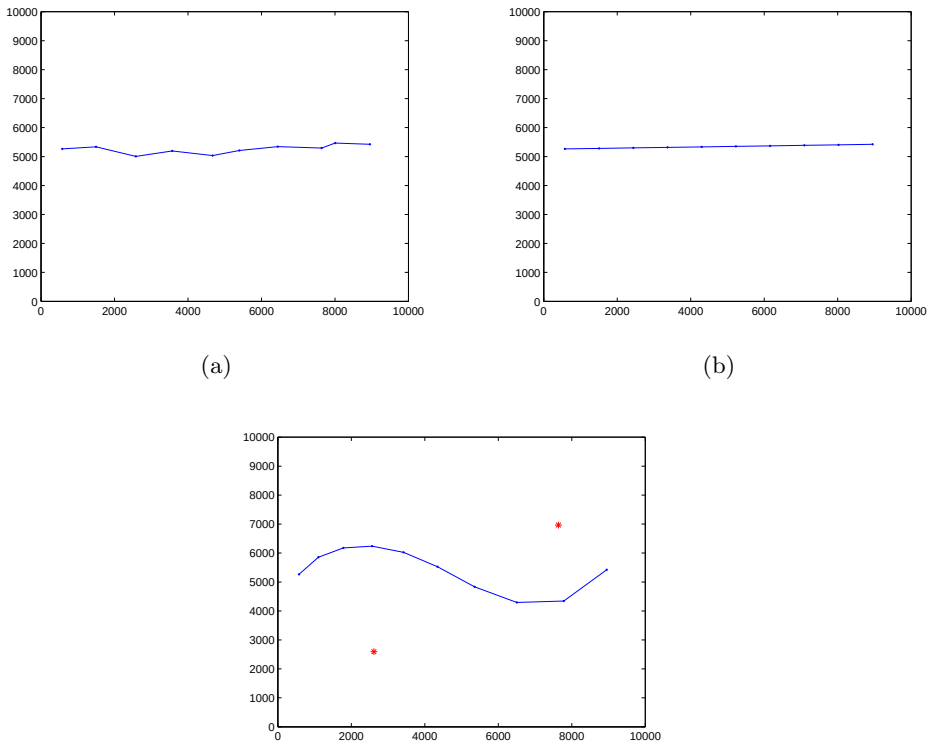
In the second step, threats are inserted into the system. They are simulated as virtual force fields, each exerting a repelling force on the masses. The forces are made inversely proportional to $1/d^4$, where d is the distance to the threat. It is important to note that these force fields are non-directional with infinite force at the center. The force from one field acting on a mass is:

$$F_{field} = \frac{\vec{n}}{d^4} \tag{4.11}$$

where d is the distance from the source and $\vec{n}$ is a normal vector pointing from the source to the mass. Using the distance to the fourth power is inspired by the radar equation (Wikipedia, 2007b) which in practice means that we are locally minimizing exposure to received radar energy. It is possible, although it requires some manual work, to create non-homogeneous force fields that better represent the risk function

of the threats. However this function still needs to be smooth so that convergence is not disturbed when the system is simulated.

When all forces are defined, we can describe the mechanical system, which in fact is a Lagrangian system, with a well known equation, namely Newton's second Law of motion: $F = ma$. Roughly this means that the movement of a mass is determined by the combination of forces that is exerted on it. In figure 4.5 a simple spring and damper system is simulated.



(a) (b)

(c)

Figure 4.5: In figure (a) an initial course path is given. In (b) the waypoints are evenly distributed between start and target using springs and dampers. In (c) two force fields are added which forces the path to bend around the sources.

If the system of virtual forces is simulated in three dimensions, the state vector, $\vec{x}$, of a mass can be written as:

$$\vec{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.12}$$

From Newtons law, we can derive the differential equation that describes the

mechanical system.

$$m_j \ddot{x}_j = F_{spring,j} + F_{damp,j} + F_{field,j} \tag{4.13}$$

where m_j is the j-th mass, F_{spring} is the the force of the springs acting on that mass, F_{damp} is the damping force and F_{field} is the force exerted by the virtual force fields. Expanding this function yields:

$$m_j \ddot{x}_j = \kappa(x_{j-1} - x_j) + \kappa(x_{j+1} - x_j) - b\dot{x}_j + \sum_{k=1}^{N} \frac{Q_k}{d_{jk}^4} \vec{n}_{jk} \tag{4.14}$$

where Q_k is a scaling factor that determines the magnitude of each force field and $\vec{n}_{jk}$ is a normal unit vector pointing from force field k to mass j.

To obtain the final solution, the system of ordinary differential equations (ODE) is evolved with a numerical solver, e.g. RK4, until the residual gets small enough and the system has reached steady-state.

Chapter 5

Implementation

From the beginning of this project it was a goal to implement an UCAV model and get it airborne in the relatively complex simulator facility at FLSC. Even though the simulator has a modular structure and contains generic aircraft models, it is a rather extensive task to construct a control system from scratch. Luckily, I was able to extend a system developed for the 2006 UCAV study (Försvarsmakten, 2006, appendix 5), which is based on the JAS 39 Gripen aircraft model.

A great advantage with having a model fly in the simulator is the high level of fidelity that is obtained, and that is hard to match by other means than by live tests. On the other side, each test run takes time to set up and cannot be performed any faster than real-time. Therefore, in order to test algorithms and techniques that were to be implemented, some simulations was set up in Matlab at first. This, of course, resulted in some extra work when it was time to port the system to C++ on IRIX which runs on the simulator servers.

5.1 Plant Description

The facilities available at FLSC were originally built with the primary purpose of training pilots in Beyond Visual Range (BVR) combat. Over the years, the tasks have increased to also include e.g. PSO scenarios with elements of Within Visual Range (WVR) combat, but also assessment and acquisition studies. To achieve the flexibility required when performing this variety of tasks, FLSC has been built as a modular system where each module in itself is a simulator, which communicates with other modules through message passing. This way it is possible to add or remove parts of the simulation without interfering to much with the rest of the system.

The main software of the simulator is called T3SIM. In figure 5.1, a schematic chart over the facility is shown. From a pilot centric point of view, the flight simulation module is the vital part, running separately on a machine dedicated to a pilot station. In parallel, on the same machine, a number of additional simulation processes, such as radar, weapons and visualization simulations, are run. At the

moment, FLSC has four 6-channel dome stations and four 3-channel stations. The domes provides a 180° horizontal field of view and a vertical field of view about 90°.

5.1.1 Hardware

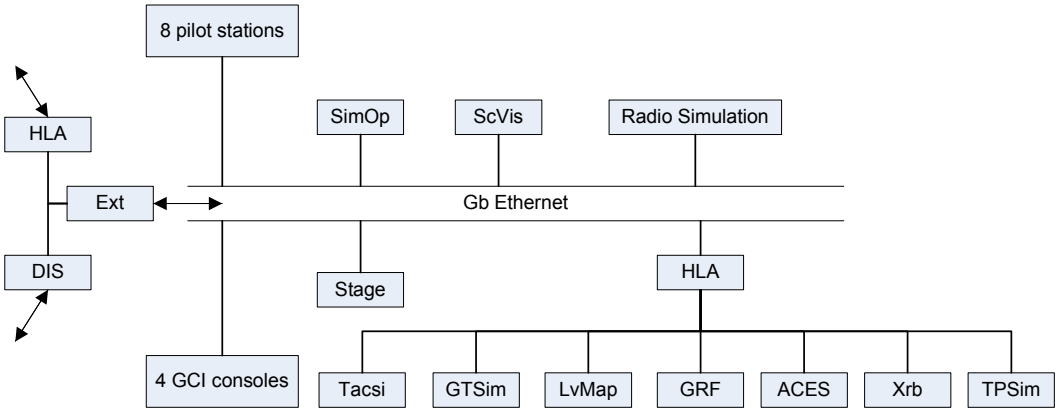


Figure 5.1: Chart over FLSC simulator

In the next section, the framework used for standardization in distributed simulation is briefly explained.

5.1.2 HLA Framework

The High Level Architecture (HLA), is an IEEE standard (IEEE, 2003) for Modeling and Simulation (M&S) systems. Its main purpose is to define a design that makes it possible for simulations, running on different platforms to interact. As implied by the name, HLA is a standard on a level above the actual implementation. For lower levels, there exists several Runtime Infrastructures (RTI) which implements communication between different simulations, referred to as federations in the HLA standard.

Through the HLA interface, a number of additional simulations, such as the CGF tool TACSI or Xrb for simulating cruise missiles, can be included in the FLSC simulation (see figure 5.1).

5.1.3 T3SIM and Tools

To initiate a simulation in FLSC, an operator interface, SimOp, is used to coordinate all participating T3SIM modules. SimOp runs on a separate server from where it connects and starts processes on dedicated computers.

An important tool for visualizing the simulations is SCVIS or, its predecessor, HawkEye. These graphical 3D "Gods eye view" representations plots the positions of all participating entities during real-time simulation or thereafter as a playback

feature. Figure 5.2 shows a BVR simulation where missiles are fired and their trajectories visualized.

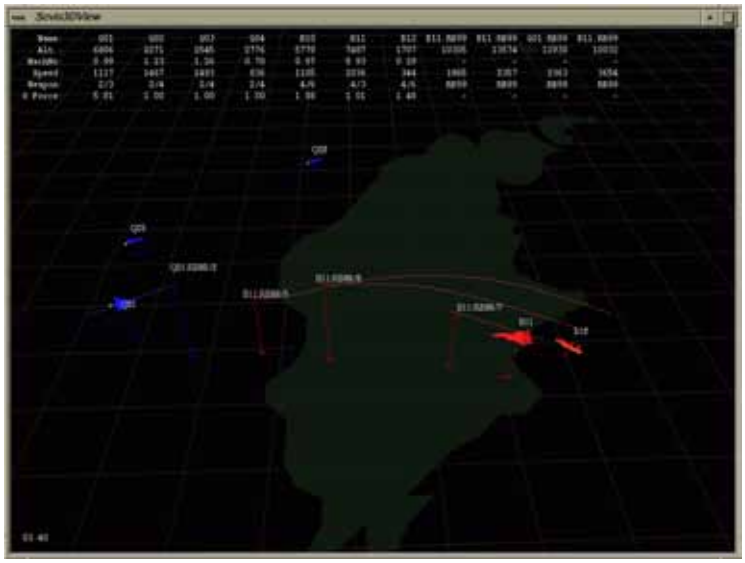


Figure 5.2: SCVIS simulation visualizer. Allows scenarios to be recorded and played back later on.

The CGF tool that was used in this thesis is called Stage, by Engenuity technologies inc. Stage was chosen because it is already integrated in T3SIM and was sufficient for the purpose. It provides a relatively simple interface for including airborne, but also some ground and sea entities. It is possible to build scripted CGF's within Stage but in the scenarios studied in this thesis, entities were either stationary or set into motion, following a predefined trajectory. In figure 5.3, a screenshot from on of the test scenarios is shown in Stage. All entities besides the blue UCAV to the left are generated by Stage.

5.1.4 UCAV Model

As stated before, a UCAV model was built on top of a JAS 39 Gripen flight model for an Air Force study in 2006. The flowchart in figure 5.4 shows the modules concerning the UCAV and how they communicate.

Associated with the UCAV model, is an operator interface (5.5). This GUI lets the operator choose between six different modes, of which the *follow mode* was the one that was expanded with the new planning prototype.

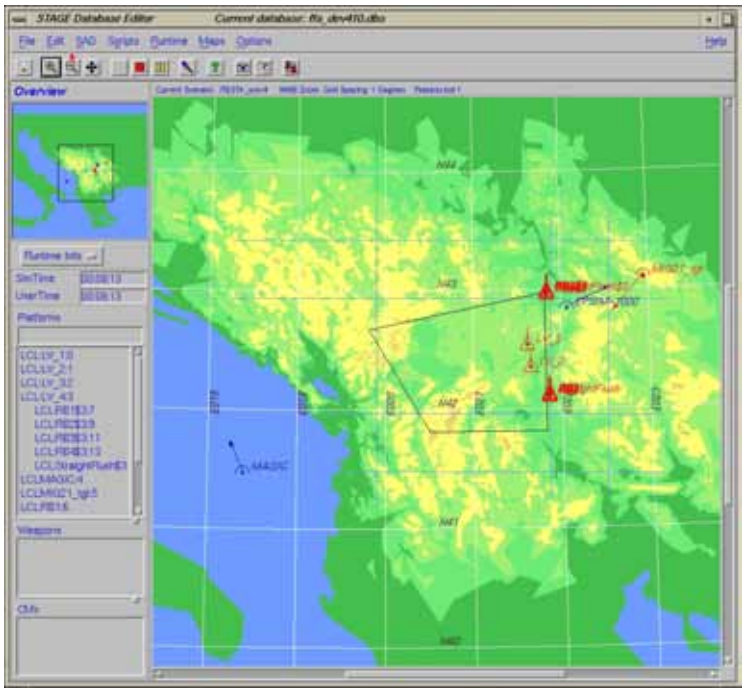


Figure 5.3: Stage, a scenario and CGF tool.

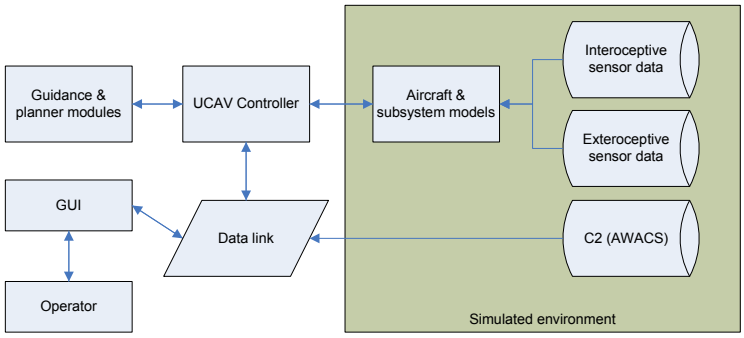


Figure 5.4: Flowchart for the UCAV control system

5.2 Matlab Simulations

Before work began with developing a prototype for running in T3SIM, the suggested methods were implemented and tested in Matlab. This was done since Matlab is a useful tool for obtaining fast results, especially when working with data on matrix form. Although powerful, Matlab was not suitable for all parts of the process. An example of this was graph search, where recursion and linked graph structures would

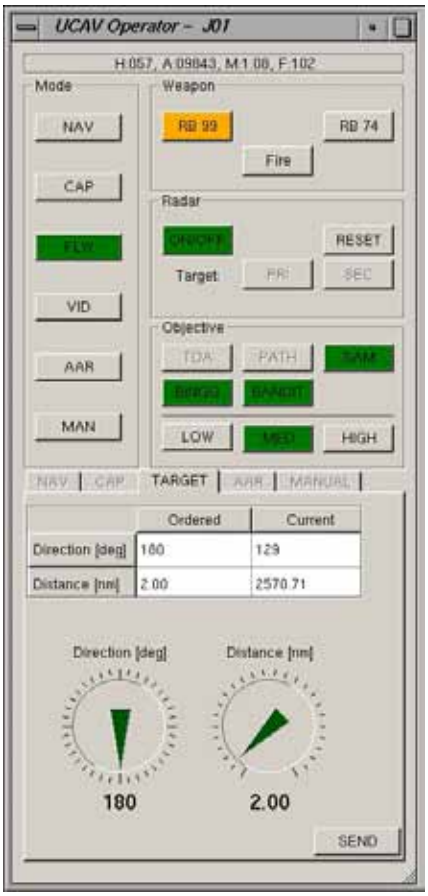


Figure 5.5: UCAV GUI/Operator interface for a UCAV in the fighter role.

have been tricky to implement. The solution to this was to implement those parts in C/C++ and make use of Matlabs mex-library to call them from inside Matlab. This was also an advantage since most of the code were later to be converted into C++ for integration with the UCAV model. To summarize, the following was tested in Matlab:

- Voronoi graph creation
- cost/threat function calculations
- A* search (through mex c++ functions)
- Virtual Forces simulation
- interpretation and visualization of data recorded in the real-time simulations

5.3 Simulator Integration

After deciding what functionalities to include in the prototype, the implementation phase started, even if some parts of the code were developed for evaluation (Matlab) and deployment (T3SIM) in parallel. In figure 5.6, a simplified UML diagram for the UCAV module is put on display. The executable class, which communicates with the aircraft model by sockets, is simply called ucav. The main class has a control class, UcavControl, which is stepped by the frequency of the simulation[1]. The logic for guiding the UCAV in one of the six available modes, navigation, combat air patrol, follow, visual identification, air-to-air refueling and manual guidance, is located in the Guidance class. This class was modified with functionality in the follow mode and equipped with a new PathPlanner class.

5.3.1 Programming Environment

Most parts of T3SIM and its associated components are composed in C++ running on Silicon Graphics machines with IRIX®6.5, except for the virtual environment visualization which runs on Windows®XP workstations.

Development was in fact not performed directly on the real FLSC simulator but in a secluded development environment. This made it possible to run and perform test without affecting ongoing live simulations. This environment was also sufficient for the testing described in the next chapter. The limitation is that it is not possible to visualize the environment as seen from the cockpit OTW (Out of The Window), but since it was unmanned aircraft that was studied, this did not affect the result at all. Another difference between the development and live environments is that the live simulator has more computational power, but that did not become a limitation during this study.

5.3.2 Planner Module

The PathFinder class was built to encapsulate the planner module. A flowchart explaining its functionality is shown in figure 5.7. Data from the aircraft model and subsystems are updated in each simulation cycle and kept in the planner. When a target is selected by the operator, its position and velocity are also reported together with data from the radar warner. The operator will then request a search for an optimal path by putting the UCAV in follow mode. This process is run in a separate process which returns a waypoint path when done. If conditions changes over a certain threshold, e.g. a new SAM sites is discovered or an old one moves more than 500 meters, a new path search is requested automatically. The pathfinder is then run in parallel and its result polled by the guidance module once produced. This process is continued until a) the goal position is reached, b) the target comes too close to the NFZ or c) the follow mode is deselected by the operator.

[1]The simulator time management server steps all participating entities with a certain frequency, e.g. 60 Hz, which requires that all updates are performed within the given time slot

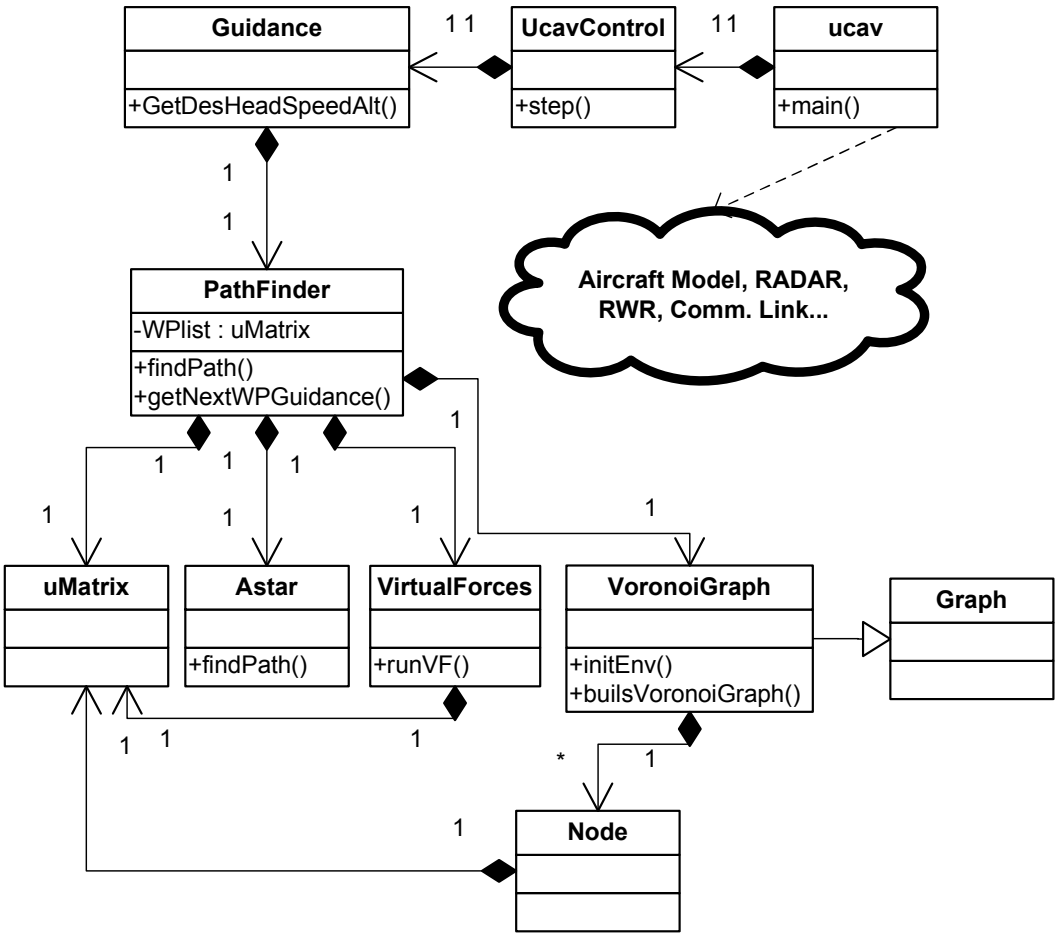


Figure 5.6: UML diagram showing relations between implemented modules

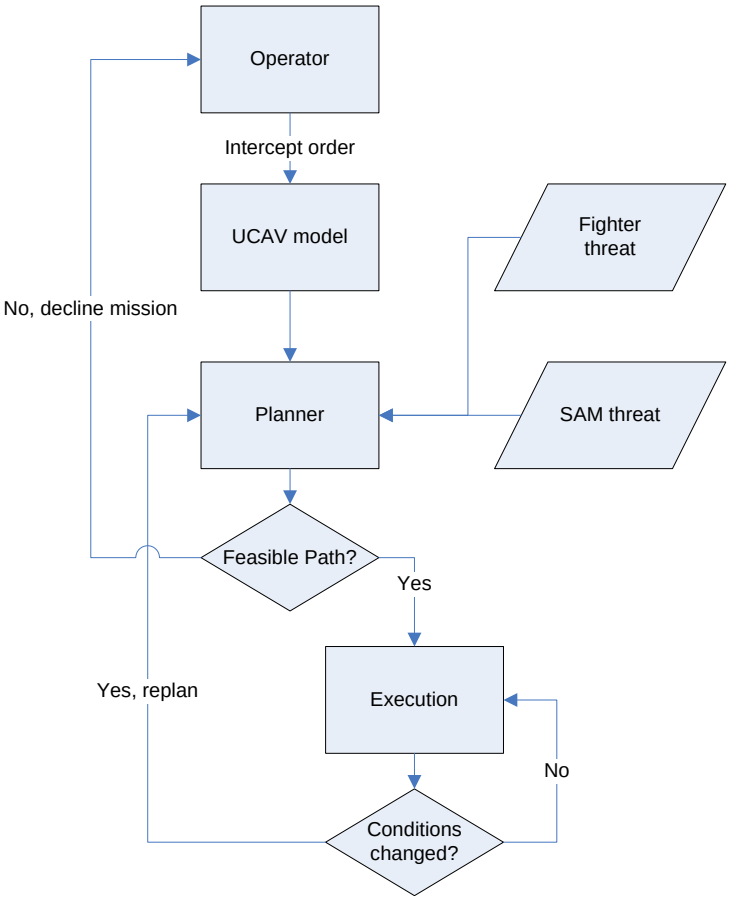


Figure 5.7: Flowchart for the UCAV planner

Chapter 6

Simulations

In an effort to verify the prototype and control its performance, a couple of test runs were done with alternating complexity. During the development phase, tests were also done in parallel but in the final testing, data on participating entities was logged for the purpose of evaluation. The test series were run in the mission training facility's development environment. This time, focus was not on the man-in-the-loop as it had been in the Air Force study, and therefore there was no need to man a cockpit with real operators. Instead, the author controlled all participating modules listed below:

1. Aircraft entity running the UCAV software
2. Airborne C2 station (AWACS)
3. CGF target (Stage object)
4. CGF SAM sites (Stage objects)
5. Visualization and recording (SCVIS)

6.1 Test Scenario

A general introduction to the chosen test scenario was given i section 3.1, Peace Support Operations. It involves a no-fly zone and a wider area of responsibility (AOR). The airspace that is controlled with the NFZ lies over the west part of country B that has been struck by internal conflicts. A warlord in the east of the country performed a coup d'état and proclaimed himself president. The international intervention commenced as a response to actions taken by the new government in order to scare away an ethnic minority which mostly inhabits the western part of the country.

To the north of B lies country A and to the south country C, both still neutral to the conflict although the presidents paramilitary forces are known to use the territory of A when they are performing operations in the west of B. The military

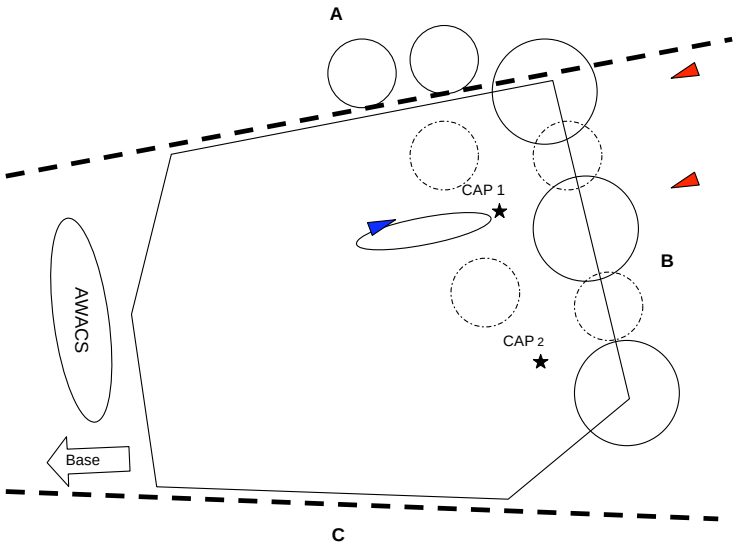


Figure 6.1: General map of the scenario and NFZ.

of B has been badly maintained for a large number of years but, as a result of the coup d'état, the new president gained access to a large quantity weapons and ammunitions. An old Air Force base was also acquired and reconnaissance have reported that two MiG 21 "Fishbed" and a number of smaller propeller driven aircraft are operational. A number of surface to air missile systems, mainly SA6 "Gainful" and SA8 "Gecko" are threatening the coalitions aerial forces. The main purpose of the NFZ is to prevent hostile aircraft to threaten the international ground forces that are operating in the western part of B. In figure 6.1 a simple map is provided.

In all test runs the UCAV was started in midair, inside the NFZ. It would then go to a predefined CAP point in order to scout along the east border of the NFZ. AWACS resources are operating from outside the eastern border of the country providing a good coverage of the airspace, at altitudes over 1000 meters. The NFZ, white polygon in figure 6.2, holds an area of approximately 16,000 km^2, the distance from the eastern to the western border is approximately 150 km and the AOR, red polygons, stretch about 50 km outside the NFZ.

6.2 Test Results

In the following tables, data from the test runs are presented. In total, five scenarios were executed with varying number of executions. The level of difficulty was raised incrementally between scenarios, mostly by altering number and placement of SAM's, but within some scenarios there were also some minor alterations of preconditions between executions. It is important to note that the scenario was not

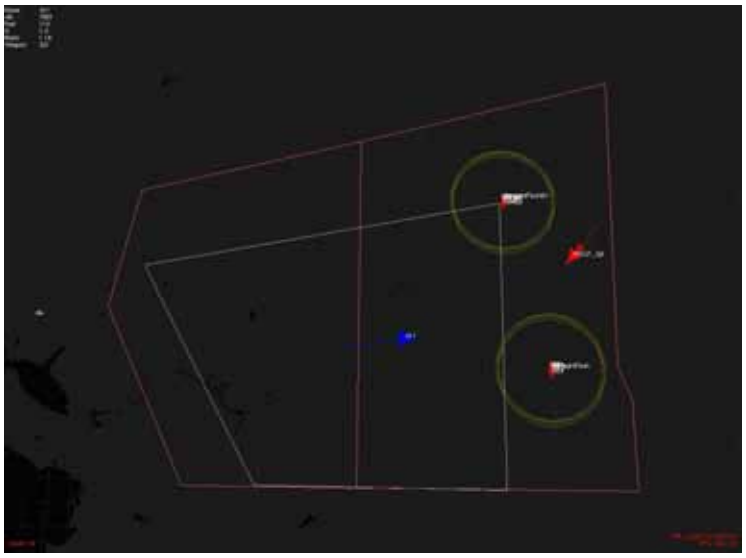


Figure 6.2: Gods eye view as seen with Hawkeye

primarily altered to reflect a realistic conflict buildup, but rather to challenge the system. After all, it exists yet no doctrine telling right from wrong regarding this kind of unmanned missions.

Since the simulations are nondeterministic, especially with regards to sensor models, the same preconditions would not guarantee the same result. Also, since there is a human operator involved, the UCAV would not get the intercept order at the exact same instant in time.

The first table (table 6.1) shows two reference executions, put here for comparison, performed with the old UCAV model, used in the Air Force study. In this model, the behavior is implemented without planning and the UCAV steers right towards the target on a tractrix trajectory, placing itself at a predefined distance behind the target. The reference runs were performed with the same preconditions as in scenario 2 and 5 respectively. In appendix A, trajectory diagrams with threat curves are displayed for all simulations.

In the tables below SA6 and SA8 signifies the participating number of respective SAM system, d_{SAM} is the closest distance on which a SAM site was passed in the scenario, v_{tgt} is the velocity of the target, which was constant during all missions, and a_{tgt} is the altitude of the target. The target was coming from North East in all executions. The maximal risk, see section 4.4.2, is given as well as an integral mean value. Since the UCAV was completely outside the range of the threats in some scenarios, a second risk value, r_2, where ranges were multiplied with 1.5, was recorded for comparison.

Table 6.2 shows results from the first scenario. In the first two missions, the intercept order is given from start, which is why the UCAV goes straight to the

scenario	SA6	SA8	d_{SAM} (km)	v_{tgt} (m/s)	a_{tgt} (m)	r (max/mean)	r_2 (max/mean)
2	2	1	29.1	250	7000	0/0	0.37/0.35
5	3	3	12.4	250	4000	0.64/0.29	0.81/0.42

Table 6.1: Reference runs

mission	SA6	SA8	d_{SAM} (km)	v_{tgt} (m/s)	a_{tgt} (m)	r (max/mean)	r_2 (max/mean)
A	2	0	24.2	250	7000	0/0	0.40/0.25
B	2	0	30.4	250	7000	0/0	0.09/0.09
C	2	0	22.4	250	7000	0/0	0.62/0.50

Table 6.2: Scenario 1

mission	SA6	SA8	d_{SAM} (km)	v_{tgt} (m/s)	a_{tgt} (m)	r (max/mean)	r_2 (max/mean)
A	2	1	37.3	290	7000	0/0	0/0

Table 6.3: Scenario 2

mission	SA6	SA8	d_{SAM} (km)	v_{tgt} (m/s)	a_{tgt} (m)	r (max/mean)	r_2 (max/mean)
A	2	2	24.4	250	7000	0/0	0.62/0.26
B	2	2	24.4	250	7000	0/0	0.62/0.26
C	2	2	21.1	250	7000	0/0	0/0
D	2	2	20.6	250	7000	0/0	0.29/0.22
E	2	2	14.0	250	7000	0/0	0/0

Table 6.4: Scenario 3

intercept point. The threats are never close enough to make an impact on the path. In mission C, the target enters the scene later which is why the UCAV have time to enter CAP mode. The UCAV stays outside of SAM range in all missions.

In scenario 2, only one test run was made. In this scenario the target moved at a higher velocity which led to an intercept inside of the NFZ.

Table 6.4 presents the third test scenario in which the SAM threat has been with two SA8 (one inside the NFZ). The UCAV stays outside of SAM ranges in all missions. Note that mission B was aborted due to a bug in the software.

Table 6.5 displays data from scenario 4, which was similar to scenario 3 but with both SA8's inside the NFZ. In mission C, the target was mistakenly flown at a very high altitude, at which the SAM's were out of range. In mission D, the target was

mission	SA6	SA8	d_{SAM} (km)	v_{tgt} (m/s)	a_{tgt} (m)	r (max/mean)	r_2 (max/mean)
A	2	2	12.4	250	7000	0/0	0/0
B	2	2	11.8	250	7000	0/0	0/0
C	2	2	9.6	250	15000	0.0/0.0	0.03/0.0
D	2	2	10.6	250	3000	0/0	0.30/0.22

Table 6.5: Scenario 4

mission	SA6	SA8	d_{SAM} (km)	v_{tgt} (m/s)	a_{tgt} (m)	r (max/mean)	r_2 (max/mean)
A	3	2	13.9	250	4000	0.57/0.39	0.78/0.34
B	3	2	5.5	250	4000	0.83/0.46	0.84/0.46

Table 6.6: Scenario 5

lowered to 3000 meters which increased the threat factor.

The last test scenario, shown in table 6.6, was created in order to overload the system. An additional SA6 was put right outside the NFZ, leaving only small openings to get through. In mission A, the UCAV managed to get through although it went inside a SA6 zone after reaching the intercept point. In mission B, one of the SA8's was moved in order to close one of the gaps and this time performance degraded, which resulted in a higher risk taking behavior.

Chapter 7

Discussion

In the previous chapter, results from real-time simulations were presented, for the purpose of evaluating the overall performance of the prototype. At first, the idea was to measure performance with pure binary values saying whether the mission succeeded and if the UCAV survived. However, if the CGF SAM sites were to get permission to fire, there would be a need for accurate missile models, and since no such non-restricted SAM models were available, it was decided to instead use the momentaneous risk function. This performance indicator could be calculated in the same manner as seen in section 4.4.2.

A drawback with testing in the mission training simulator was the fact that it is time consuming for several reasons. Firstly, the simulations must be run in real-time. Even if the scenarios were created to cut as much "idle" time as possible, it takes time to fly a desired path. The mean length of all test missions was 8.9 minutes, but to this we have to add quite some time for setting up the scenario and to extract and save the data afterward. It was also necessary to supervise the simulations and to play the role of UCAV operator and Ground Controlled Intercept, (GCI), operator. The structure of the simulator makes it hard to make scripted test runs, which could help in generating a more statistically significant amount of data. Instead, it was decided to make a qualitative analysis, based on a number of interesting situations.

7.1 Performance Evaluation

In most of the simulations, the prototype can be said to have accomplished the mission, which is to reach the intruder in time without exposing itself to a high risk. Exceptions were scenario 2A, where the UCAV did not manage to intercept the target outside the NFZ, 4D, where the UCAV was accidentally shot down since the wrong SAM model was used and scenario 5B, where extensive replanning took the aircraft on a path through a SAM region.

When comparing the results from the planner with the reference runs, it seems that the planner usually selects a path that takes smother detours around dangerous

areas. The logic in the reference model struggles with contradictory aims; Escape SAM and chase target and that yields a trajectory that "hesitates" close to the SAM range (see figure A.2). It could also get stuck in a local minimum between two sites, and that cannot happen to the planner version.

A fact that becomes evident when reviewing the results of the simulations is that, if replanning occurs very frequently, the resulting path would typically be less optimal than when the plan was allowed to be executed for a longer period of time. An example of this can be observed in appendix A.17 where the path takes alternating left and right turns resulting in more time spent inside a potentially dangerous zone. In this simulation, replanning was performed in average each fourth second, which in this scenario must be considered way to often. One explanation for this behavior is uncertainty in sensor data which will be discussed in the next section.

7.1.1 Radar Warner

A key factor in generating the safest path is to have a clear and complete picture on where threats and own forces are located. In the simulations that were made, the only source used to localize SAM positions was the on-board Radar Warner (RWR). Although it is a useful and efficient system, its main purpose is to classify and warn the pilot of nearby radar emitters such as other fighter radars, active missiles and SAM radars, and for this purpose smooth tracking is not requested. As seen in chapter 3.5.3, the triangulated position of for example a SAM site is associated with less uncertainty if the different measurements are taken at a large relative angle. Due to the placement of antennas on the fuselage, the RWR has different effective ranges at different azimuth's, affected by both roll and pitch. Uncertainty is also dependent on the distance to the threat.

In figure 7.1, We see a SAM detection that is pretty far away from its actual position although it is on a correct bearing from the UCAV (triangle mark). When the UCAV closes in and receives additional measurements, the positioning will be more accurate.

A consequence of poorly tracked threats becomes obvious when comparing figures 7.2b-c. In (b), the original path (light colored dots) takes a course south of the center SAM site, whereas it in (c) takes on a northerly path. These two plans are made closely after each other in time, resulting in completely contradictory steering orders. Figure 7.2 also shows many examples of SAM sites represented by double detections and sites being detected far away from their actual positions.

7.1.2 Virtual Forces

The post processing step was necessary to create a feasible trajectory within the safe proximity of the coarsely planned path. The method infers a number of adjustable parameters, see section 4.5.1, amongst which the virtual force constant, Q, was the one that weighted path length against how far away from threats the UCAV

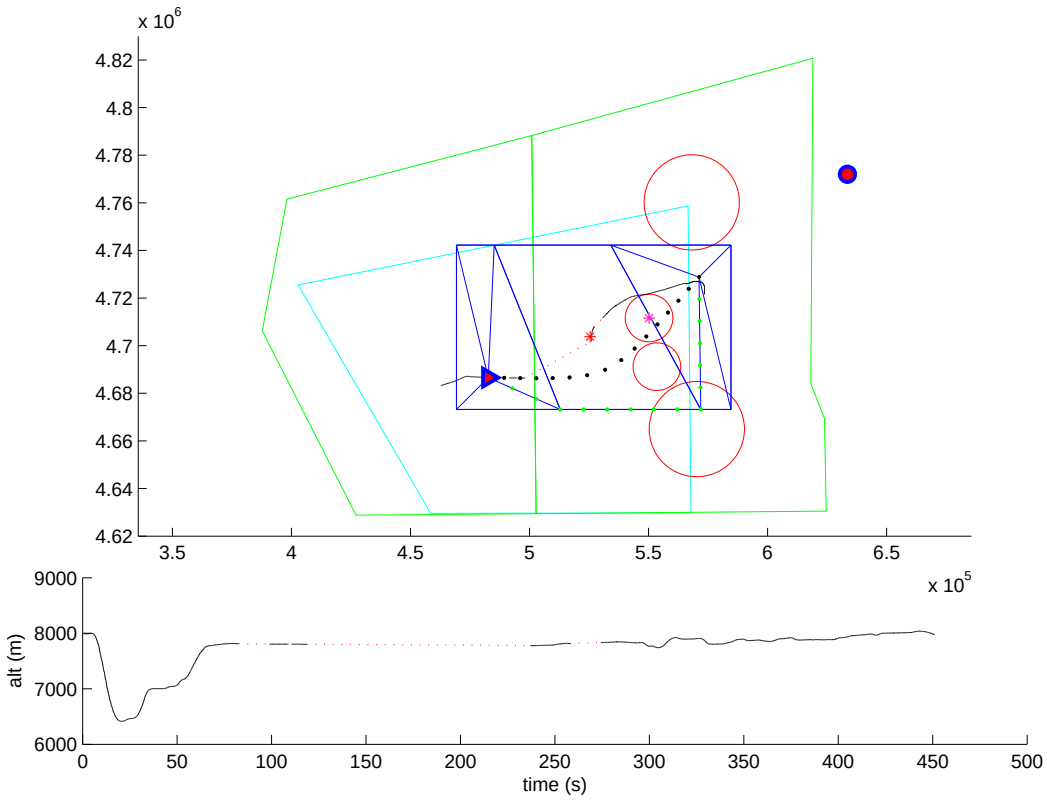


Figure 7.1: Snapshot of plan during target intercept mission. The left (red) asterisk marks the only SAM site currently detected by the RWR. As seen, the detection is on the right bearing although it is far away from the actual position (right asterisk). Note also how the very coarse planned path is smoothed with virtual forces.

was to be pushed. During testing, different values of Q were tested until results looked reasonable for the chosen scenario. The effect of this parameter is however highly dependent on the number of waypoints and their start positions and therefore a global optimal value for it does not exist. A numerical optimization of Q by minimizing the threat function would enhance output but this would have to be payed for in extra time consumption since each calculation of virtual forces is done by solving a system of ODE's. A cheaper solution would be to test a small number different Q values in the proximity of the last one used and choose the best of them.

Although Virtual Forces has proven a very useful tool, a few problems was observed, most of them related to the value chosen for the virtual forces weight, Q. Figure 7.2 shows examples of some virtual forces problems. In all of the examples, some or several of the waypoints are even placed within the SAM sites. The Q value has also been given a too large weight for threats, which results in that the path is pushed too far away. In 7.2b, the path has been pushed away in another direction

than its original A* planned path. In 7.2c, virtual forces has exceeded its maximum allowed number of iterations to reach an equilibrium state, resulting in something that does not resemble a path. The reason for this is that when waypoints starts too close to a threat, the repulsing force acting on it pushes it away far enough to induce oscillations that are not damped within the given time. In 7.2d, we see a path that, though it leads right, also is repelled further away than necessary. Remember that the path does not end at the target, marked with the big circle, but at a calculated meet-up position outside the NFZ.

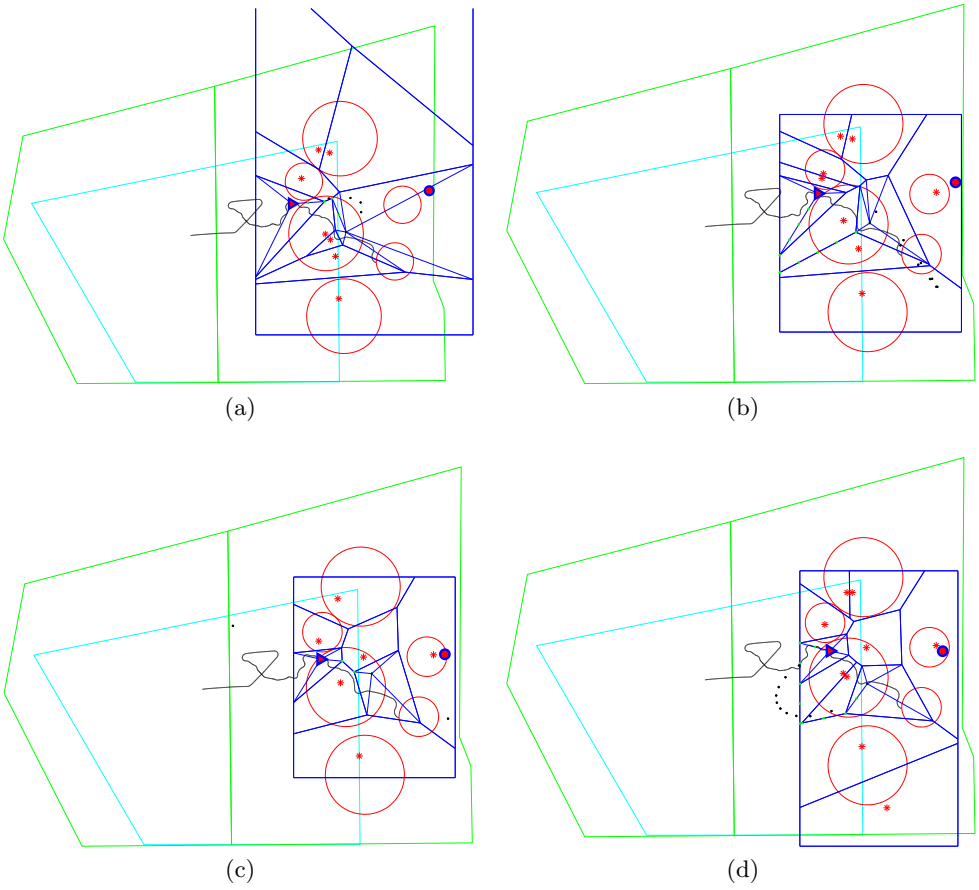


Figure 7.2: Figures showing the effect of unsuccessful Virtual Forces computations. The plans originates from simulation 5B which was somewhat of an overload test. Note that the detected location of SAM sites (asterisks) alters between plans and that, in some cases, multiple SAM's are detected where there should be only one. Again, the circles depict the actual SAM location

7.1.3 Trajectory Following

Paths that are generated by the planner are returned as lists of waypoints. This format is a convenient representation deriving from the discretization needed in the planning algorithms but it is not necessarily the most efficient representation to use for guidance of the aircraft. The regulation that was used to control the prototype was a built-in auto pilot that could set either speed and altitude or speed and heading. For evaluating the planner, this was considered enough but should the system be used for a higher fidelity simulation, a controller that follows a path rather than sets a direction would enhance the model. In that case, the UCAV could follow an interpolated path, thus giving it a more natural and efficient behavior.

7.2 Conclusions and Further Work

Throughout this thesis, the problem of navigation by path planning for aerial vehicles in dangerous environments has been studied. A number of methods were evaluated of which one approach was implemented as a prototype in a simulated environment.

An important lesson from developing the prototype was that creating an agent for modeling combat aircraft is an extensive task, even with a very limited field of operation, that requires contributions from many different disciplines ranging from automation to cognitive science.

Even if focus has been set on planning and optimization, the computer generated forces aspect has been kept in mind in order to highlight the often fuzzy responsibility zone between human operator and computer. It has become clear that even if the system were to make more of the decisions it has to account for them visually in order for a human to comprehend and for the sake of traceability.

As seen earlier, planning by graph search can potentially be associated with an exponential time complexity. However, with discretization by Voronoi regions the search space could be limited and by heuristic search an optimal path was found in $n \log n$ time. During testing, path finding was executed in its own thread, since the time slot given by the simulation frequency, 60 Hz, was not sufficient. The delay in the planner was on the other hand never big enough to influence the performance and thereby we have shown that the methods are suitable for on-line use. Many considerations regarding parameters and weights has still to be done by the human operator, but since we have not yet reached the limit of computational capacity, some additional optimizations could probably be done to relieve the man-in-the-loop.

7.3 Goal Fulfillment

In the beginning of the project, a set of sub goals were stated and this is my opinion on how they were met.

1. *Identify what parameters to use for decision making and optimization*
   This question might be one of the more challenging when approaching these kinds of optimization problem. It was chosen to consider only the SAM threat in this study. A threat function was constructed, based on distance and angle towards the SAM sites. All measured parameters used derived from simulated sensor data.

2. *Evaluate how to best make use of planning algorithms*
   Different methods were studied during the first phase of the project. Two approaches of building search spaces were tested, of which the Voronoi approach was considered best suited. The graph search worked out well and could be proven optimal, although enhancements can be done to the search space and the post processing steps.

3. *Develop a prototype running in FLSC*
   This phase took up a much greater part of the total project time than what was estimated, much because it felt like an important step that enriched the author in knowledge regarding real-time simulation. The prototype performed its task correctly in most test cases and can hopefully be used in future CGF/UCAV development at FLSC.

4. *Make use of recent methods and research results*
   During the background study, several recent projects with similar aims were compared. Since the concept of Unmanned Combat Aerial Vehicles is rather new in itself, most research is up to date even if it makes use of older well tested algorithms.

7.3.1 Proposed Enhancements

It very likely that Computer Generated Forces and simulation will play a bigger role in military aviation in the near future, and this development is already observable for example at FLSC. I believe that it is necessary not only to build advanced agents but even more to gain knowledge on how to use them and how to interpret their behavior. For this reason, and since agents are to act independently, traceability becomes a very important issue in proving an adequate behavior. Only by extensive use, testing and a deeper understanding can trust be built and advantages be fully achieved.

In order to achieve this comprehension for CGF behavior, an agent should be rule-based, providing a simple and intuitive presentation of current state and what decision it makes. Although this is in itself a big design problem, I find it likely that such a system would gain from having the possibility of choosing from operations that optimizes behavior during a predefined interval of time or during a specific maneuver. An example of this could be the path planning operation that is dealt with in this report.

Focusing only on the UCAV model, a list of enhancement is proposed for future use below.

UCAV model

- persistent tracking of SAM sites with Kalman filter (less toggling)
- incorporate threats from mission planning (Military Intelligence)
- consider maneuvering threats (other fighters)
- alternative path evaluation
    - variable threat weight
    - variable virtual forces strength
    - different altitude patterns
- evasive actions when receiving radar warning

It would also be wise to upgrade the operator interface in order to study more realistic human to UCAV communication. The following enhancements are proposed.

Operator Interface

- Include a tactical display (map indicator) to present:
    - threats
    - friendly forces
    - current path
    - alternative paths
- risk level presentation
- risk management (choose acceptable level)
- path selection
- path speed selection

Figure 7.3: nEUROn, © Dassault Aviation - A. Ernoult

Bibliography

Randal W. Beard, Timothy W. McLain, Michael Goodrich, and Erik P. Anderson. 2002. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922. URL `citeseer.ist.psu.edu/beard02coordinated.html`.

Marcelo Becker, Richard Hall, Björn Jensen, Sascha Kolski, Kristijan Macek, and Roland Siegwart. 2007. The use of Obstacle Motion Tracking for Car-like Mobile Robots Collision Avoidance in Dynamic Urban Environments. In P. S. Varoto and M. A.Trindade, editors, *The XII International Symposium on Dynamic Problems of Mechanics (DINAME 2007)*, Ilhabela, SP, Brazil, February 2007.

Scott A. Bortoff. 2000. Path planning for UAVs. In *American Control Conference*, Chicago, IL, USA, June 2000.

Silvia Coradeschi, Lars Karlsson, and Anders Törne. 1996. Intelligent Agents for Aircraft Combat Simulation. In *Proceedings of the 6th Computer Generated Forces and Behavioral Representation Conference*, Orlando, Florida.

E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Mark Edwards, Eugene Santos Jr., Sheila B. Banks, and Martin R. Stytz. 1996. Computer Generated Intelligent Companions for Distributed Virtual Environments. In *Eighth IEEE International Conference on Tools with Artificial Intelligence*. IEEE. ISBN 0-8186-7686-7. URL `citeseer.ist.psu.edu/10244.html`.

Patrick A.M. Ehlert, Quint M. Mouthaan, and Leon J.M. Rothkrantz. 2003. A Rule-Based And A Probabilistic System For Situation Recognition In A Flight Simulator. URL `citeseer.ist.psu.edu/728139.html`.

David Eppstein. 1998. Finding the k shortest paths. *SIAM J. Computing*, 28(2): 652–673. URL `http://dx.doi.org/10.1137/S0097539795290477`.

Dave Ferguson and Anthony Stentz. 2005. The Delayed D* Algorithm for Efficient Path Replanning. In *European Control Conference, CDC-ECC*, Seville, Spain, April 2005. IEEE.

Försvarsmakten. 2005. Doktrin för luftoperationer. Swedish Armed Forces publication 09833:60610. URL `http://www.mil.se/article.php?id=10833`.

Försvarsmakten. 2006. SLUTRAPPORT UCAV och UAV. Final report in the UAV/UCAV concept study conducted by the swedish Air Force. (Unpublished).

IEEE. 2003. IEEE Std. 1516.3. ISBN 0-7381-3584-4. URL `http://ieeexplore.ieee.org/servlet/opac?punumber=8526`.

Randolph M. Jones, John E. Laird, Paul E. Nielsen, Karen J. Coulter, Patrick G. Kenny, and Frank V. Koss. 1999. Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine*, 20(1):27–41. URL `citeseer.ist.psu.edu/jones99automated.html`.

Sascha Kolski, Dave Ferguson, Mario Bellino, and Roland Siegwart. 2006. Autonomous Driving in Structured and Unstructured Environments. In *IEEE Intelligent Vehicles Symposium*.

Steven M. LaValle. 2006. *Planning Algorithms*. Cambridge University Press, Available at: `http://planning.cs.uiuc.edu/`, Cambridge, U.K.

Patric Lavén. 2006. Fighter pilot and mission trainer at FLSC. interview.

Qiuxia Liang, Patrick Ehlert, and Leon Rothkrantz. 2004. Towards A Neural Control Artificial Pilot. URL `citeseer.ist.psu.edu/729380.html`.

M. W. McConley, M. D. Piedmonte, B. D. Appleby, E. Frazzoli, E. Feron, and M. A. Dahleh. 2000. Hybrid control for aggressive maneuvering of autonomous aerial vehicles. In *Proceedings of the 19th Digital Avionics Systems Conference, vol 1*, pages 1E4/1 – 1E4/8. DASC.

R.K. Mehra, J.D Boskovic, and Sai-Ming Li. 2000. Autonomous formation flying of multiple UCAVs under communication failure. In *Position Location and Navigation Symposium*, pages 371–378. IEEE.

Paul Nielsen, Don Smoot, and JD Dennison. 2000. Participation of TacAir-Soar in Road Runner and Coyote Exercises at Air Force Research Lab, Mesa, AZ. In *9th Annual Conference on Computer Generated Forces and Behavioral Representation (CGFBR-00)*.

Petter Ögren, Adam Backlund, Tobias Harryson, Lars Kristensson, and Patrik Stensson. 2006. Autonomous UCAV Strike Missions using Behavior Control Lyapunov Functions. In *Guidance, Navigation and Control Conference*. AIAA.

Petter Ögren and Maja Winstrand. 2005. Combining Path Planning and Target Assignment to Minimize Risk in SEAD Mission. In *Guidance, Navigation and Control Conference*. AIAA.

Per-Magnus Olsson. 2002. Intelligent Agents For Aircraft Handling. Master's thesis, Linköping University.

William A. Owens and Ed Offley. 2000. *Lifting the Fog of War*. Farrar, Straus & Giroux, Incorporated. ISBN 0374186278.

Roland Philippsen. 2006. A Light Formulation of the E* Interpolated Path Replanner. Technical report, Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne.

Yao-Hong Qu, Quan Pan, and Jian-Guo Yan. 2005. Flight path planning of UAV based on heuristically search and genetic algorithms. In *Industrial Electronics Society, IECON 2005*. IEEE.

Helen Rosander and Jonas Walther. 1996. Användning av artificiella neurala nät vid simulering av piloters taktiska beslut under luftstrid. Master's thesis, Umeå universitet, Institutionen för datavetenskap.

Stuart Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*, 2nd edition edition. Prentice-Hall, Englewood Cliffs, NJ. ISBN 0-13-080302-2.

N. Sadati and J. Taheri. 2002. Solving robot motion planning problem using Hopfield neural network in a fuzzified environment. In *International Conference on Fuzzy Systems*, pages 1144 – 1149. IEEE.

Lars-Åke Siggelin. 2006. Electronic warfare operator, Swedish Air Force. interview.

Wikipedia. 2007a. Iraqi no-fly zones. URL `http://en.wikipedia.org/wiki/Iraqi_no-fly_zones`.

Wikipedia. 2007b. Radar. URL `http://en.wikipedia.org/wiki/Radar`.

Maja Winstrand. 2004. Mission planning and control of multiple UAVs. Master's thesis, Swedish Defence Research Agency, Stockholm.

U. Zengin and A. Dogan. 2004. Probabilistic Trajectory Planning for UAVs in Dynamic Environments. In *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, Chicago, Illinois, September 2004. AIAA.

Appendix A

Simulation Diagrams

In the following pages, scenario graphs for all recorded simulations are displayed. For each simulation, the path of the UCAV is plotted within a scenario graph. The inner and outer polygons are NFZ and AOR respectively. In all scenarios the UCAV starts inside the west side of the NFZ. In some simulations it continues up to the CAP point in the upper East of the NFZ but in others it goes straight on the target. The target, which is not displayed in the graphs, is approaching the AOR from North East in all simulations.

Beneath the scenario graph, an altitude diagram displays the altitude of the UCAV during the simulation and beneath that, a risk diagram is shown. The dotted line in the instantaneous risk diagram shows the risk, given a 50 % augmentation of SAM ranges.

Due to errors in the sampling process, some parts of certain simulations were not recorded. These parts are dotted in the scenario and altitude graphs. In figures A.5, A.14 and A.15, the altitude curve goes down to 200 meters. This occurred when the UCAV was performing CAP and was nothing but an error in the mission data file.

A.1 Reference Simulations

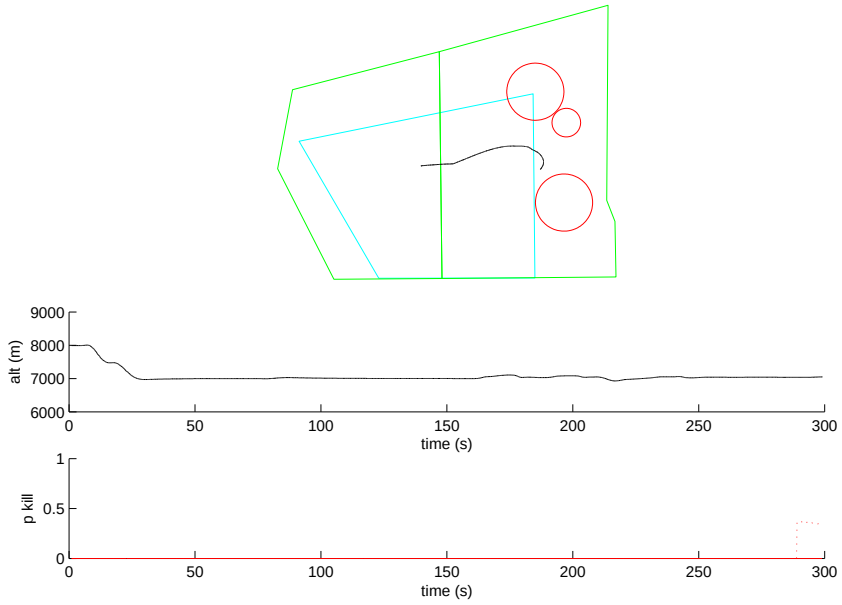


Figure A.1: Reference 1 (scenario 2)

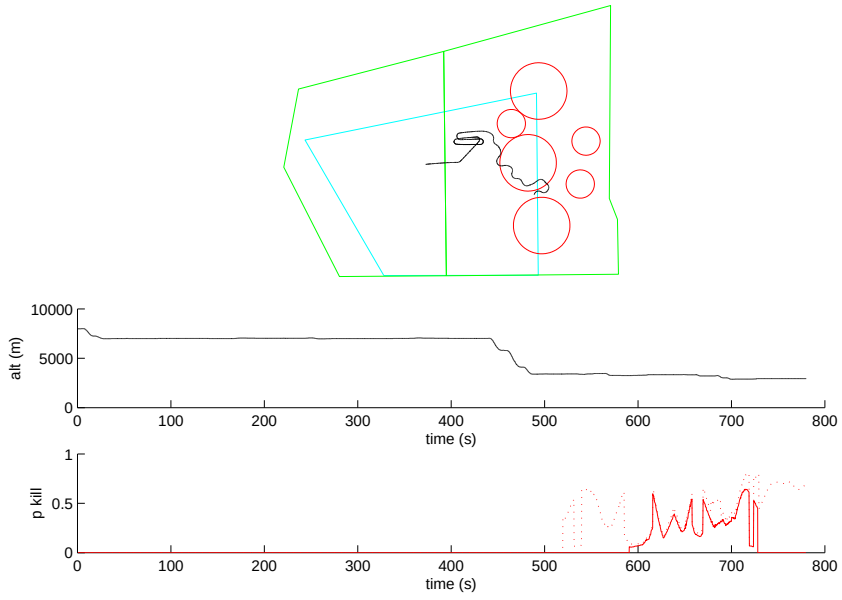


Figure A.2: Reference 2 (scenario 5)

A.2 Simulation Serie 1

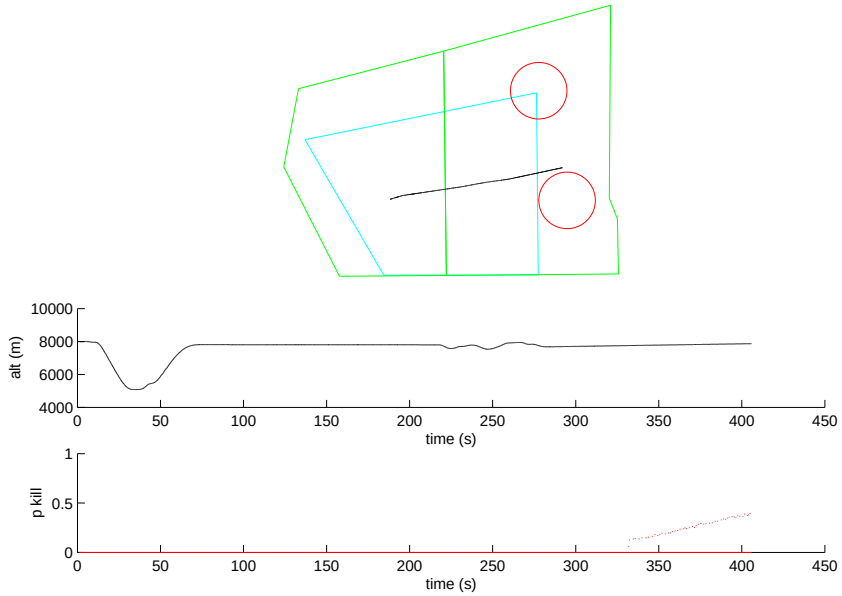


Figure A.3: Test scenario 1A

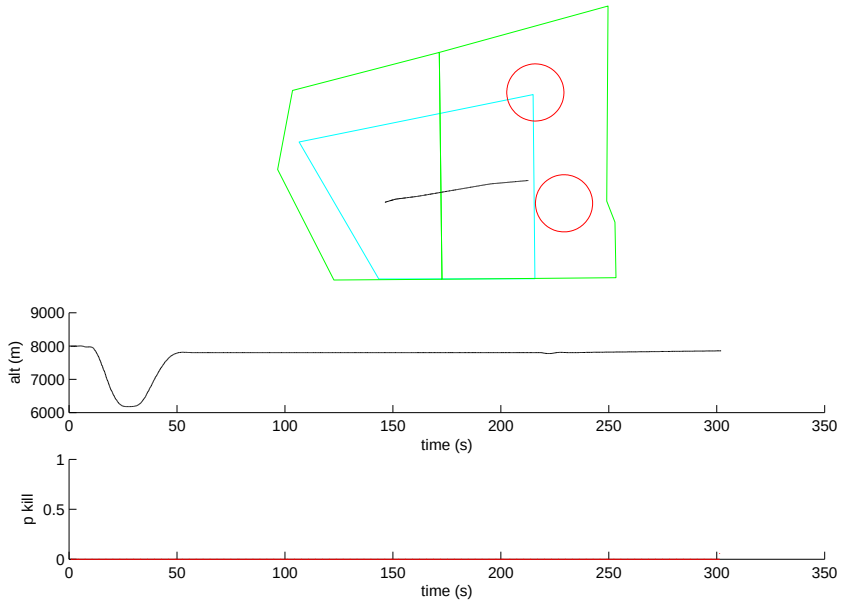


Figure A.4: Test scenario 1B

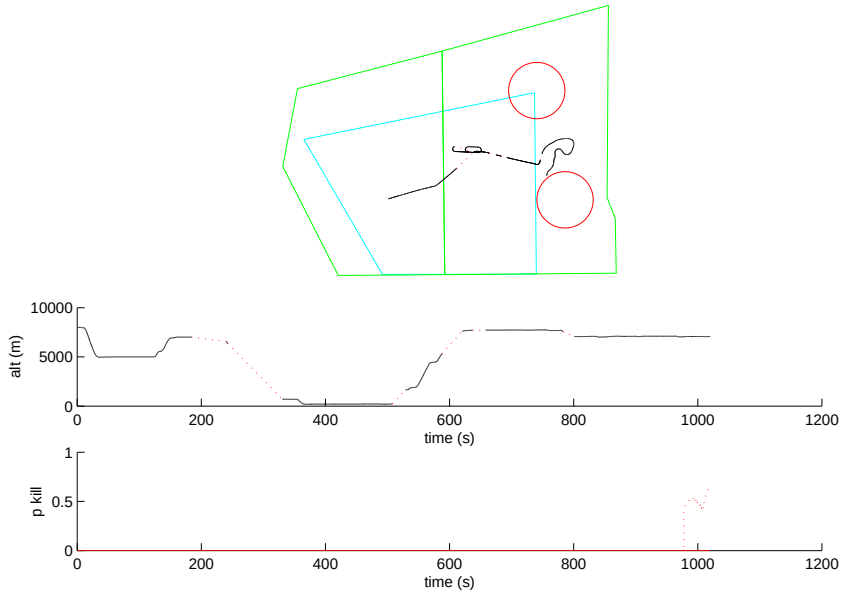


Figure A.5: Test scenario 1C

A.3 Simulation Serie 2

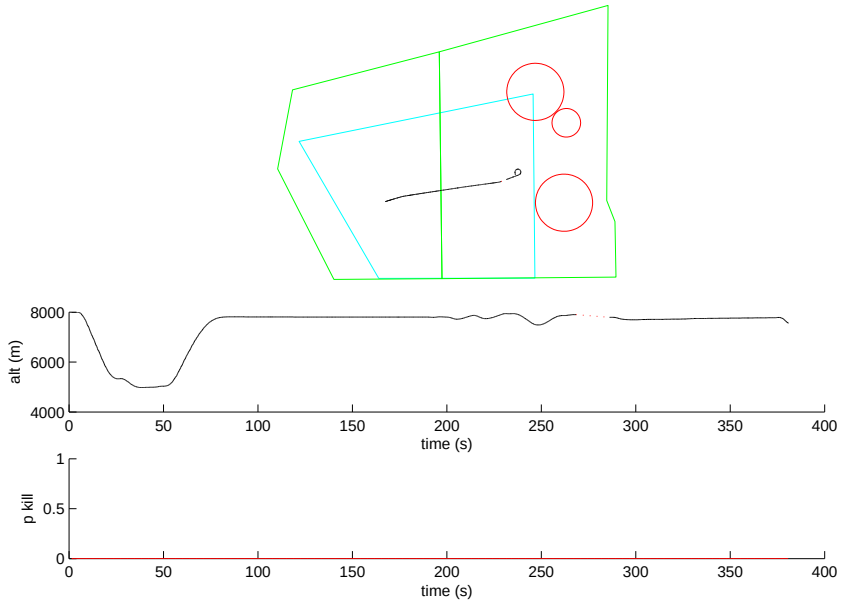


Figure A.6: Test scenario 2A

A.4 Simulation Serie 3

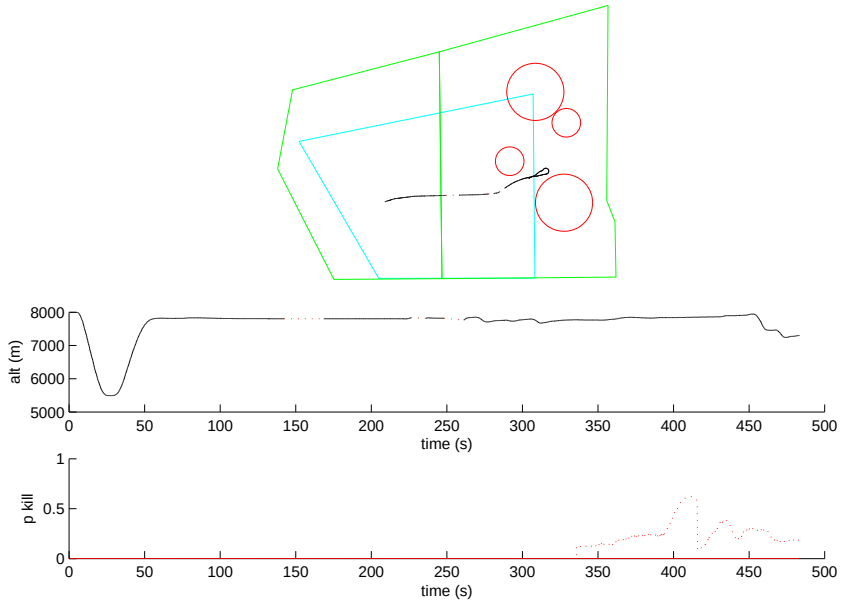


Figure A.7: Test scenario 3A

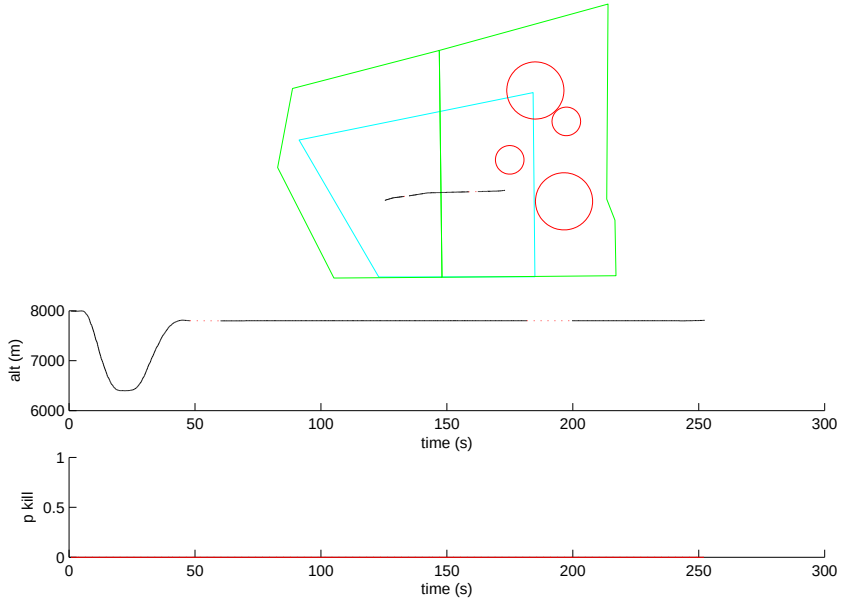


Figure A.8: Test scenario 3B

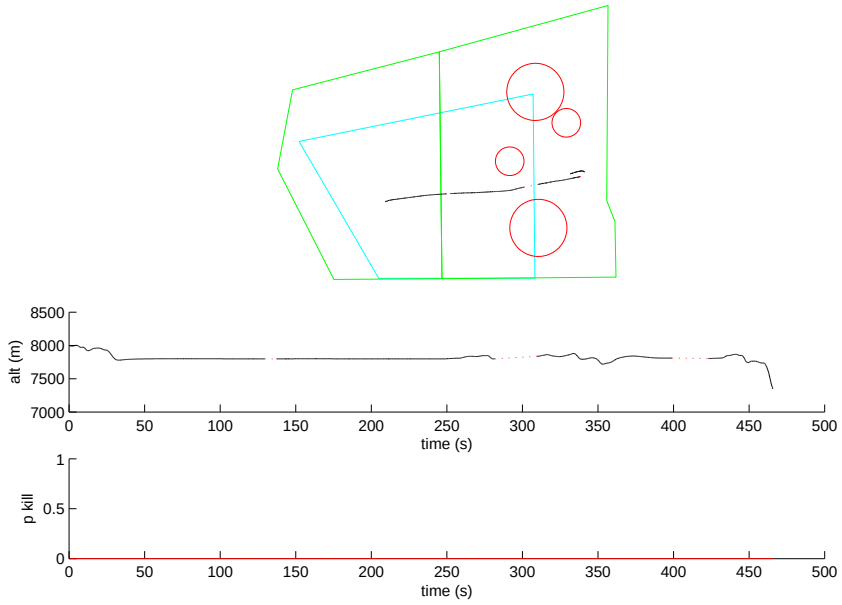


Figure A.9: Test scenario 3C

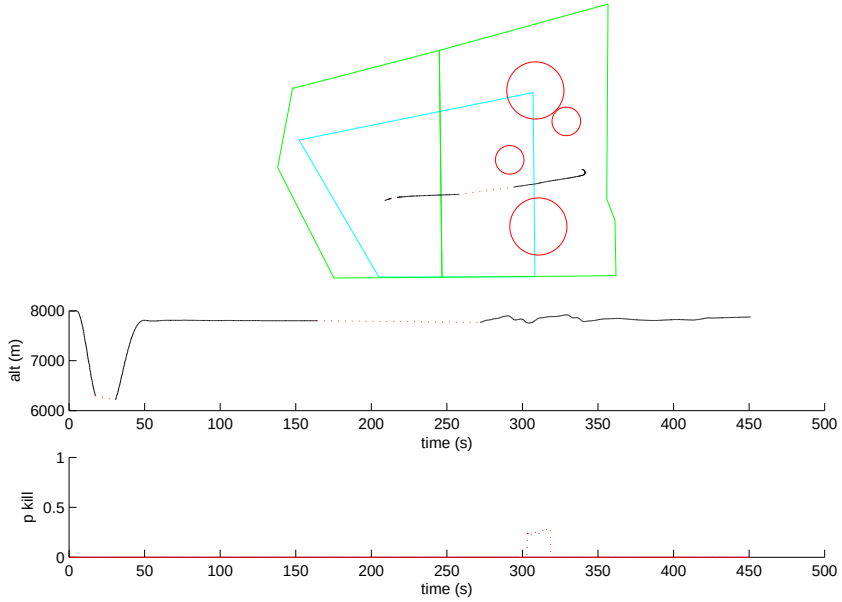


Figure A.10: Test scenario 3D

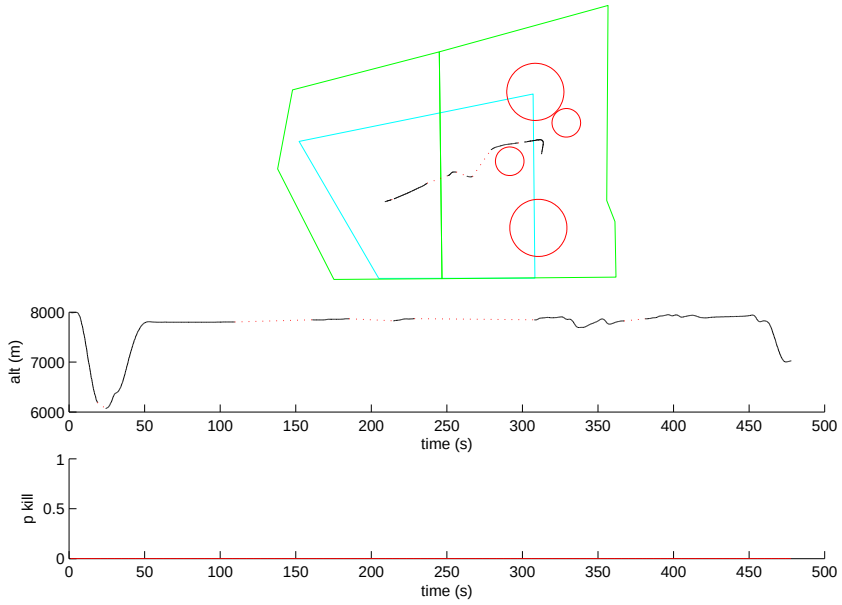


Figure A.11: Test scenario 3E

A.5 Simulation Serie 4

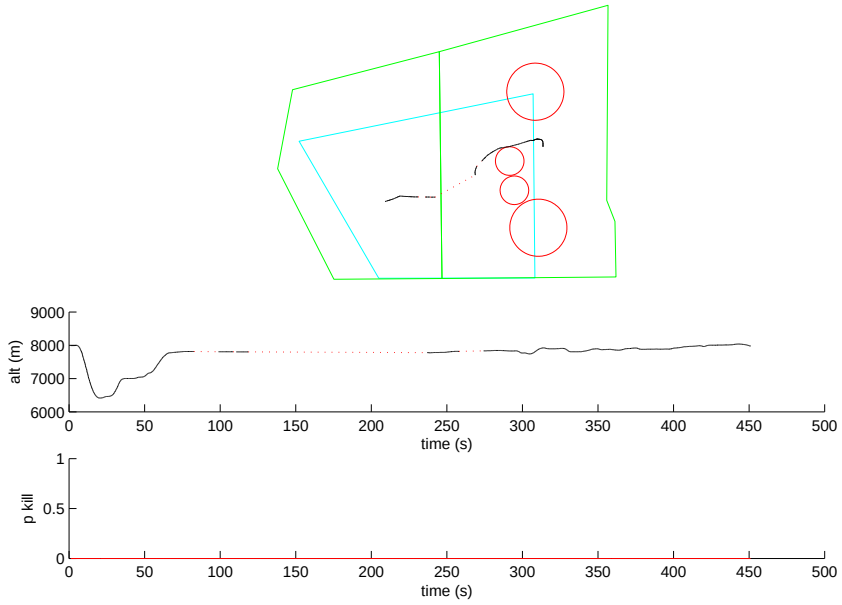


Figure A.12: Test scenario 4A

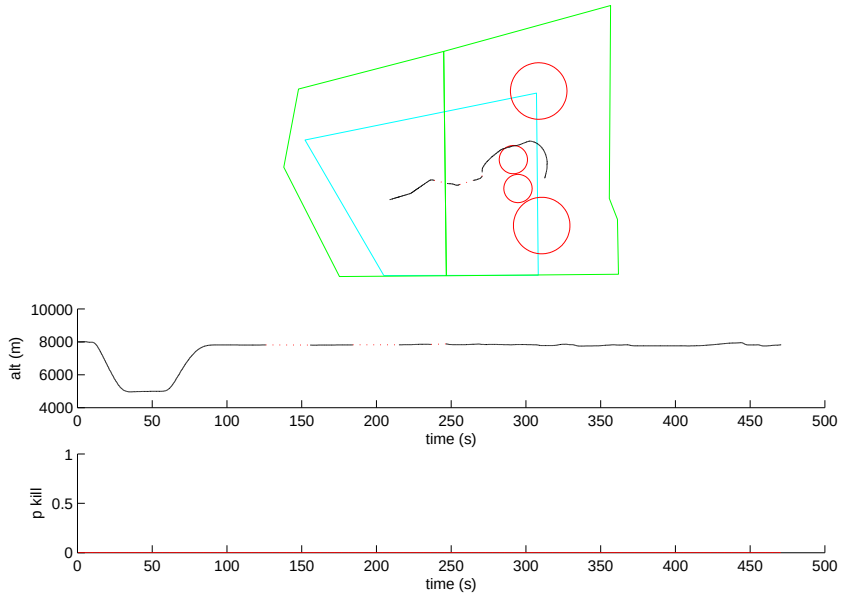


Figure A.13: Test scenario 4B

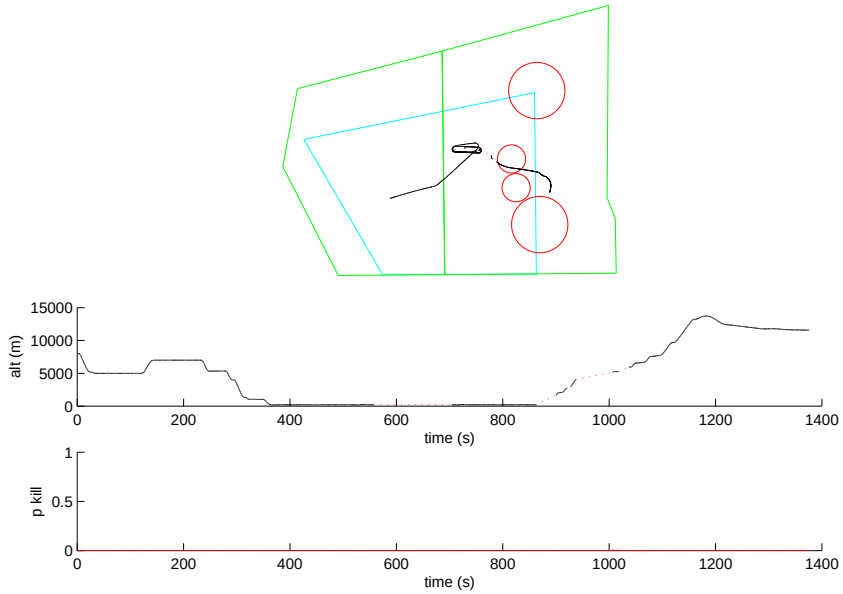


Figure A.14: Test scenario 4C

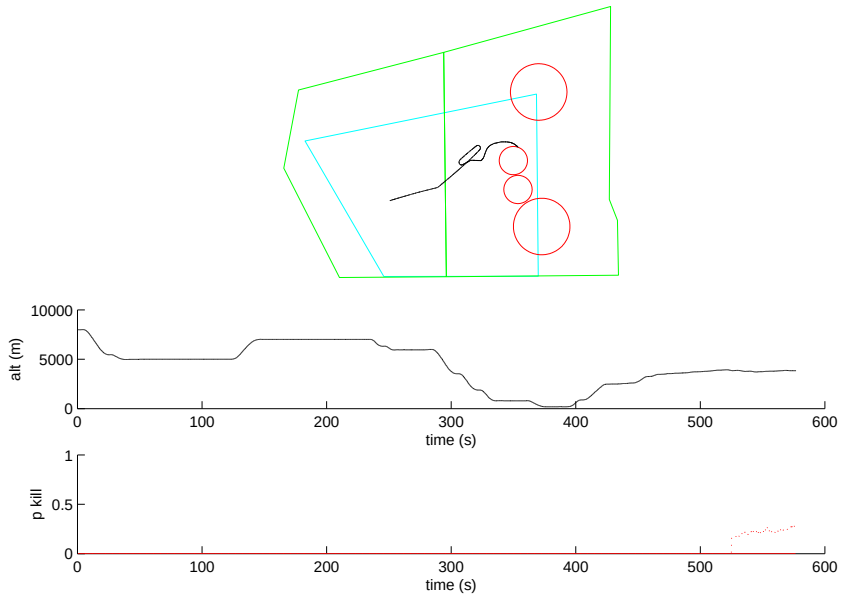


Figure A.15: Test scenario 4D

A.6 Simulation Serie 5

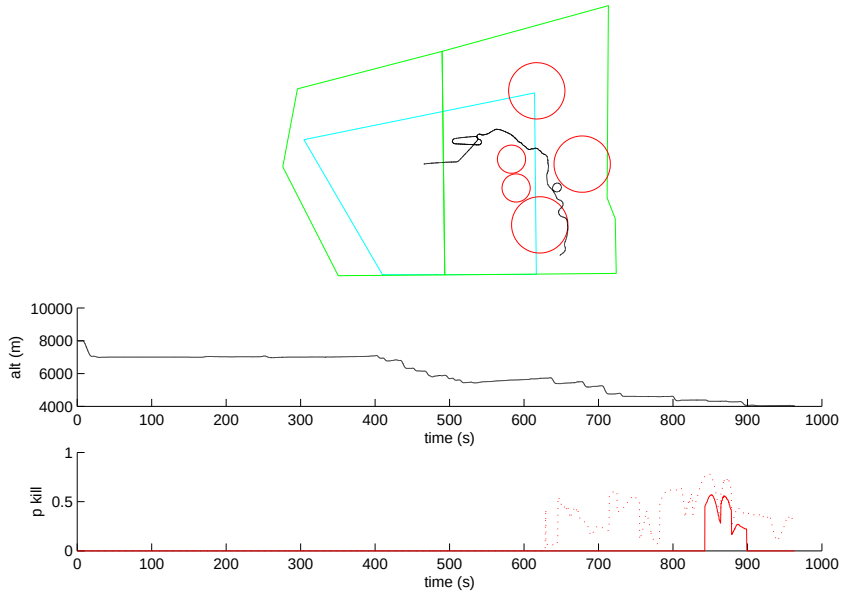


Figure A.16: Test scenario 5A

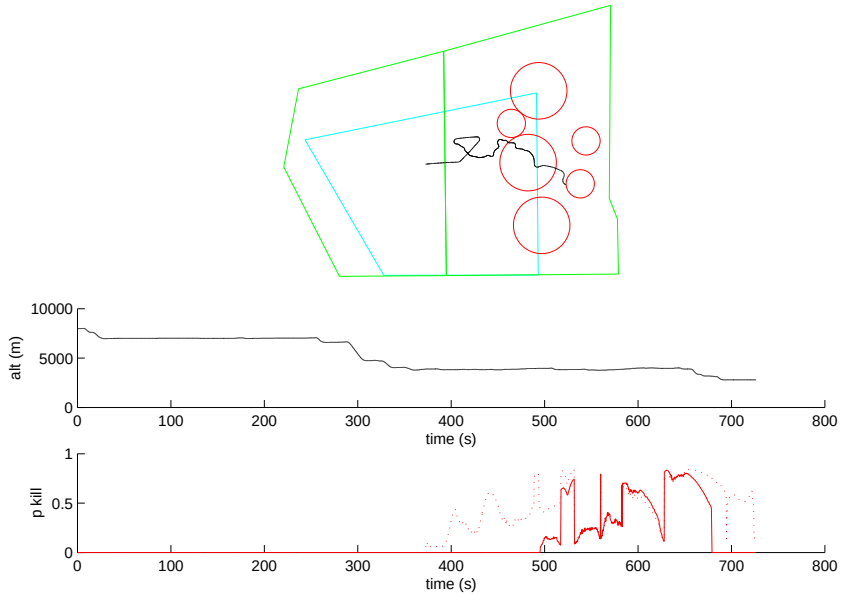


Figure A.17: Test scenario 5B

Appendix B

Background Study

Many articles have been written on how to handle rules in CGFs. One of the earlier attempts is the Soar architecture which originally was created by John Laird, Allen Newell, and Paul Rosenbloom at Carnegie Mellon University. Soar is a cognitive architecture for modeling human behavior and it is short for State, Operator And Result. In the air combat domain, Soar has been used for large scale US Air Force exercises (Jones *et al.*, 1999). Rule based architectures for air combat has also been studied by SAAB and Linköping University (Coradeschi *et al.*, 1996), which has resulted in a simulation tool called TACSI. In a thesis, (Olsson, 2002), TACSI is used as platform for creating intelligent agents.

The first issue that has to be dealt with in path planning is how to model the environment. In mobile robotics dealing with ground vehicles, many authors use occupancy grids to keep track of surrounding obstacles, e.g. (Kolski *et al.*, 2006; Philippsen, 2006). Occupancy grids divides the environments into cells and assigns a belief value to each cell that corresponds to the probability of that cell being occupied. In the field of unmanned aerial vehicles the most constraining factor on the path is rarely solid obstacles but more commonly threats from fix points on the ground or from other aerial vehicles. This type of environment would render very large and changing grids and therefore other methods have been applied. One such method is the Voronoi tessellation (Beard *et al.*, 2002; Ögren and Winstrand, 2005; Qu *et al.*, 2005). A Voronoi graph consists of polygons where each polygon contains only one threat and where each point within the polygon is closer to that threat than any other. Hence, traveling along the edges of the graph assures keeping a maximum distance to the closest threats. Since real-time is an important issue in UAV handling, path planning is often performed in two steps. An offline planner that runs before mission start and an on-line replanner that guarantees real-time behavior.

When a graph has been crated it can be searched, for instance by using a greedy algorithm like A^* or *k shortest paths* (Eppstein, 1998). Since the desired optimal path minimizes length and threat exposure, the cost of traveling along an edge is often a weighted sum of risk and length. Another way of dealing with this is to

apply a Weight Constrained Shortest Path optimization (WCSP). In (Ögren and Winstrand, 2005) the path length act as constraint while the overall risk of the path is minimized.

Also genetic algorithms can be used to find a globally optimal path (Qu *et al.*, 2005). In this case the Delaunay triangulation of the threats is used in the encoding. Nodes are labeled 0 or 1 which constructs a genetic code for the set of threat points. The entry and exit edges are predefined and each triangle that is traversed must be entered and left by crossing a line with a 0 vertex to the left and a 1 to the right. By evolving and evaluating the results, an optimal path can be constructed.

The path generated through a Voronoi graph is often quite rough with long edges that forces the vehicle to take unnecessary long paths around threats. One way to post-process the path is by applying virtual forces (Bortoff, 2000). Break-points along the path is then treated as masses connected by springs. Threats are considered as virtual force fields, pushing away the masses. Equilibrium is reached by solving the ordinary differential equation describing the system and a trimmed path is the result.

Apart from the above proposed methods there is also probabilistic planning (Zengin and Dogan, 2004) that is well suited for an environment containing dynamic threats. Probabilistic methods can be implemented to run on-line, following a previously planned waypoint path while respecting constraints.

Rapidly Exploring Random Trees (RRT) is a technique that also has proven successful in path planning (McConley *et al.*, 2000). The method was originally developed for systems with many degrees of freedom, e.g. robot arms, since the high dimensionality makes grid spaces unreasonably large. Instead of discretizing the search space before searching, nodes are placed randomly in a tree structure emanating from the start position.

In the following sections a few related papers are more thoroughly described.

B.1 Cognitive Modeling

The papers under this section deals mainly with rule-based architectures for controlling behavior in Computer Generated Forces.

B.1.1 Automated Intelligent Pilots for Combat Flight Simulation

In this article (Jones *et al.*, 1999), lessons from using TacAir-Soar, a rule-based system for simulating automated human-like behavior in military airborne missions, are presented from a practical viewpoint. The authors participated with the system in the large scale US military exercise "Synthetic Theater Of War 1997", STOW-97, where they had a good chance of evaluating the system. The STOW-97 48 hour's operational exercise was primarily conducted in order to show the possibilities in using autonomous units in military training. The virtual arena for this exercise was a 500 x 775 square kilometers large area where up to 3700 computer-generated

vehicles participated. The TacAir-Soar system flew all of the military fixed-wing air missions, comprising 722 sorties.

The simulations were run over a distributed network with over 300 computers at different sites in the US and England. Both manned simulators and computer generated forces were plugged in and, in order to minimize load on the network, all connected units stored their own copy of the environment and made use of dead reckoning, which allowed them to send updates only when their state differed from the prediction.

TacAir-Soar is built within the Soar architecture which is a tool for cognitive rule-based simulation. The set of rules, approximately 5,200 during STOW-97, is used to execute different operators. These operators can be either atomic or abstract. Abstract operators, or "goals", are dynamically decomposed by rules to execute more primitive operators. For instance an abstract operator "Execute-Mission", containing goals and sub goals describing the task, would be required to initiate a mission. Alongside with the task specific operators, there are so called "opportunistic operators" that can be independently selected. These operators are mostly affiliated with survival goals of the aircraft such as taking evasive actions when in danger.

The behavioral requirements that were set for TacAir-Soar were the following:

1. Fly all types of missions using appropriate doctrine and tactics.
2. Fly many different types of aircraft.
3. Generate human-like behavior.
4. Coordinate behavior with other entities.
5. Behave with low computational expense.

All missions that were flown during STOW-97 were derived from Air Tasking Orders (ATO) and Air Coordination Orders (ACO) given by the Air Operation Center. The orders were broken down at the different Wing Operation Centers (WOC) and given to the pilots. Due to the high number of missions and parameters associated with each mission, an automated WOC was created that could do the bulk of the work.

As STOW-97 was one of the first large scale simulated military exercise, it raised lots of interest for CGFs and especially the Soar architecture. TacAir-Soar has since then been extended and today it comprises a set of over 10,000 rules.

B.1.2 Participation of TacAir-Soar in Road Runner and Coyote Exercises at Air Force Research Lab, Mesa, AZ

This article, (Nielsen *et al.*, 2000), describes the participation of TacAir-Soar in the exercises Roadrunner 98 and Coyote 98 at the US Air Force Research Laboratory (AFRL). The system had been further developed since cooperation in STOW-97

and now focused more on collaboration and coordination within the strike package. The TacAir-Soar CGF's was to interface each other but also human pilots flying simulators. A large amount of effort was put into communication interface between aircrafts (synthetic and real) and between AWACS and aircrafts. The interface between units was implemented as simulated radio speech and for that reason a commercial speech recognition system was set up alongside a text to speech system. The communication interpretation was simplified since the Air Force makes use of standard brevity terms.

B.1.3 Intelligent Agents for Aircraft Combat Simulation

In relation to the TACSI system developed at SAAB, the authors participated in a project aiming at creating a tool for the generation of intelligent agents for the air combat domain. Unlike similar systems, e.g. TacAir-Soar, the group tried developing a system where the rules, i.e. requested behavior of the agent, could be specified by an expert in the air combat domain without aid from an expert in the computational domain.

The architecture that was chosen was decision-trees and the user was supposed to be able to create his own trees for each participating agent. Each branch in the tree, conjunction of conditions, corresponds to a requested action. The main difference from an ordinary decision-tree is that each branch is associated with a dynamic changeable priority. These priorities are used to determine in what order the actions are to be executed. Iterating down a certain branch of the tree, the priorities are increased or decreased depending on the conditions.

Conditions are validated based on the state of the aircraft and according to the article the state is divided into four types. The first one considers the characteristics of the agent such as aircraft type and weapons loaded. The second type concerns velocities and directions of other aircraft detected with on-board sensory. Thirdly, there is the perception of the agent itself in the environment (velocity direction etc.) and last there is the memory of the agent that includes important past events. The actions are also divided into subclasses: primitive actions, concurrent actions and sequential actions.

B.1.4 Computer Generated Intelligent Companions for Distributed Virtual Environments

(Edwards *et al.*, 1996) developed a system for controlling CGF aircraft with a rule-based system guided by fuzzy logic. The idea was that, in order to imitate human behavior, one needs to consider uncertainty, ambiguity and approximation. The Fuzzy Wingman had a number of databases for different levels of control. In the article, the basic level, flight control, was presented. Fifteen linguistic variables were used in the decision process and in the tests the Fussy Wingman was ordered to fly as wingman to a human pilot.

B.2 Intelligent Control

This section presents articles that mainly deals with aircraft control based on knowledge about the environment. A few systems using Neural Network Controllers are represented that are set up to imitate and augment human control.

B.2.1 Towards A Neural Control Artificial Pilot

(Liang *et al.*, 2004) tried to create a neural controller for basic maneuvering of a simulated aircraft. Their work is part of a project called "The Intelligent Cockpit Environment" and the goal is to ultimately create a realistic, human-like, artificial pilot. The system consisted of a planner module, a neural controller module and a graphical user interface module. The planner was there to plan and execute orders inputted by the user via the GUI.

In the experiments the controller was limited to control elevator and throttle, disregarding rudder and aileron control used for turning. The plane is thereby limited to flying in a vertical plane and the outputs measured were pitch and airspeed, leaving out heading and bank angle.

The controller chosen was the Forward Modeling and Inverse Control which basically consists of two neural networks. The first net, the forward model, is a pre-trained model of the plant (aircraft) and the second, the actual controller, is an adaptive inverse model to the plant. The error signal is propagated back through the forward model and into the inverse controller that uses it in the training. Due to their simplicity and good properties in on-line training, a recurrent network, namely the Jordan Network, was used. In implementation, the C++ program Stuttgart Neural Network Simulator (SNNS) was used to create the neural networks.

B.2.2 Autonomous UCAV Strike Missions using Behavior Control Lyapunov Functions

In 2006 the Swedish Air force conducted a tactical study regarding UCAV capabilities. Simulations were made in FLSC with an operator supervising the autonomous aircraft whose control structure is explained in (Ögren *et al.*, 2006). The aircraft had had five operational goals, rephrased as Lyapunov functions, that it struggled to satisfy. These were: Arrive at the target within the given time window, return to base without running out of fuel, stay on a given waypoint flight path, stay outside of the range of SAM sites and stay away from enemy fighters.

The controller was based on user defined priority levels where each level had different demands on the Lyapunov functions. The higher the level the harder the constraints. At each level the system struggled to reach the next level by taking actions to reach the goals. For instance the aircraft would steer away from SAMs as long as sticking to the path or arriving on time did not have higher priority.

Although being able to react to pop-up threats and executing a mission with desired result, the system requires a predefined and fix flight path. There is also a

high number of parameters, needed to define the behavior. In testing, the operator was able to choose from three different sets of parameters that resulted in the vehicle taking more or less risks.

B.2.3 Hybrid control for aggressive maneuvering of autonomous aerial vehicles

In (McConley *et al.*, 2000) an approach for hybrid control of autonomous aerial vehicles is proposed. The hybrid control constitutes of trim trajectories and a set of pilot inspired maneuvers. Trim trajectories can be seen as equilibrium points where the system is at steady-state, i.e. level flight. A pre defined maneuver could for instance be a roll, in which the aircraft starts and ends in a trim trajectory. With this quantization of system dynamics, the controller chooses amongst predefined maneuvers and trim trajectories, reducing the control problem to a mixed-integer-programming problem.

Furthermore, Rapidly exploring Random Trees (RRT), are proposed for path planning and obstacle avoidance.

B.2.4 Autonomous formation flying of multiple UCAVs under communication failure

(Mehra *et al.*, 2000) introduces a system for handling formation flying of multiple UCAVs. The full system hierarchical control structure is presented but the article concentrates on the lower levels, mission planning and control. The system is capable of deterministic leader reassignment, if the leader would leave the formation in flight. Further, this article proposes ways of handling communication failure within the formation. Another interesting aspect is the guidance law which calculates the followers trajectories based on what the leader does (The leader is assumed to have a preplanned trajectory to follow).

B.2.5 Användning av artificiella neurala nät vid simulering av piloters taktiska beslut under luftstrid

In a master thesis performed at FFA (Rosander and Walther, 1996), Aeronautical Research Institute of Sweden (later merged with FOI), a Neural Network controller was implemented to play the role of a combat pilot engaged in dog fighting in a simulator. The aircraft was controlled with multi layer feed-forward networks that took the the dynamic state of the aircraft as input. Additionally relative angles and a rough distance estimation to the target were inputted. The output was given as centripetal and tangential accelerations together with delta roll angle. After poor results with a single network the authors managed to get good results by using a separate network for each output parameter. The networks were not trained with humans due to shortage of time. Instead, an artificial pilot program, called heuristic pilot, was used in the training process.

B.3 Path Planning

This section present articles concerning path planning using different approaches.

B.3.1 Coordinated target assignment and intercept for unmanned air vehicles

In (Beard *et al.*, 2002) a heuristic method is used to coordinate target assignment and intercept for multiple UAVs. The scenario consists of an area filled with targets and threats where a number of UAVs are to select and simultaneously intercept some of the targets while minimizing their risks. The problem is approached with a Voronoi diagram which generates possible paths that are as far away from the threats as possible. The path planner then takes into account the lengths and threat exposure along each line segment and uses a k-shortest paths algorithm, (Eppstein, 1998), to derive the best possible paths from a UAV to a target.

A target manager uses the evaluated paths to make a team assignment, comprising one target for each UAV, which is considering the two opposing constraints of maximum force and maximum spread. Maximum force expresses the goal to have as many UAVs as possible assigned to each target and maximum spread the goal of intercepting as many targets as possible.

When the assignment is done the intercept manager chooses a "time on target" for the vehicles assigned to the same target. This includes choosing from the available paths and speeds while minimizing the risks.

Finally a trajectory generator stakes out the proposed paths, smoothing the vehicles behavior at the breakpoints. Should a "pop-up"-threat be detected during flight, the entire procedure is updated, taking into account the additional information.

B.3.2 Combining Path Planning and Target Assignment to Minimize Risk in SEAD Mission

(Ögren and Winstrand, 2005) proposed a path planning method similar to the one of (Beard *et al.*, 2002) but concerning autonomous aircraft on SEAD mission (Suppression of Enemy Air Defense). In this scenario the UCAV was capable of carrying a small number of anti radiation missiles (HARM) suppressing some of the threats along its path. As in other studies, the UCAV is to move from position a to b while minimizing path length and risks. A Voronoi graph is used for discretizing and the problem is formulated as how to maximize the possibility of survival while choosing among possible paths. After reformulation, the problem can be projected onto a WCSPP (Weight Constrained Shortest Path Problem) and solved with bisection search and a shortest path algorithm.

The main advantage with this approach is the representation of the risk along the paths. The vehicle then get to make the path decision based on length and total risk of failure.

B.3.3 Probabilistic Trajectory Planning for UAVs in Dynamic Environments

This article, (Zengin and Dogan, 2004), proposes a probabilistic approach for planning trajectories through an environment containing threats. By modeling the environment with a time-variant threat exposure map, the dynamic properties of the threats can be taken into consideration. In each simulation step the vehicle is given a number of possible trajectories by discretizing the turning and velocity spaces. The reference trajectory, which the algorithm is trying to execute in each step, is the one taking the vehicle as close as possible to the direction of the target.

B.3.4 The Delayed D* Algorithm for Efficient Path Replanning

In path planning it is often of interest to use fast algorithms that can render a result on-line. Therefore replanning algorithms are of great interest since they give results faster than if a plan from scratch was to be made. Especially in robotics, where a map of the environment is often known before start but, due to a dynamic world, changes with time. The Delayed D*, (Ferguson and Stentz, 2005), is an extension to A* that uses dynamic programming to "repair" the original path.

B.3.5 Solving robot motion planning problem using Hopfield neural network in a fuzzified environment

The problem of robot motion planning was approached with a Hopfield Neural Network in (Sadati and Taheri, 2002). The environment is modeled with an occupancy grid and fuzzified so that an occupied cell is surrounded by cells with decreasing occupancy value. This is done in order to represent uncertainty within the grid. The robot then has the ability to move from a cell to one of its eight adjacent neighbor cells. A Hopfield net is set up with three neurons being horizontal distance to target, vertical distance to target and uncertainty of the cell. By choosing appropriate weights and setting up a Lyapunov energy function the energies of surrounding cells are calculated and the robot chooses to move to the cell with least energy.

The use of Hopfield Neural Networks in path planning is interesting and the authors propose an addition to the method that prevents it from ending up in local minimums. However, a more detailed description of the work would be necessary in order to evaluate it properly.

B.3.6 Path Planning for UAVs

(Bortoff, 2000) presents a combined approach to path planning using Voronoi graph search and virtual forces. The goal for the vehicle is to avoid detection by enemy radars, hence a stealthy path is required. The initial, but coarse, plan is given by dynamic programming on the Voronoi graph where the weight of each edge is a function of length and probability of detection. In the next step a refined path is generated as the steady-state solution to a Lagrangian mechanical system driven

by virtual forces. The forces are made up by springs that exert a contracting force which act to minimize path length, and force fields at the radar locations that forces the chain of masses away. The force fields acts on the masses with a factor $1/distance^4$ which is also found in the radar equation that determines on what distance an object can be detected.